IMPLEMENTING MEDIATORS WITH CHEAP TALK

A Dissertation

Presented to the Faculty of the Graduate School

of Cornell University

in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

by

Ivan Eduardo Geffner

August 2021

IMPLEMENTING MEDIATORS WITH CHEAP TALK

Ivan Eduardo Geffner, Ph.D.

Cornell University 2021

In this work we study the effects of cheap-talk in games with rational agents. We begin by showing that all asynchronous interactions between n players and a mediator can be t-bisimulated without the mediator if $n > 4t$. Intuitively, t-bisimulation means that for any deviation performed by an adversary controlling at most t players in the scenario with the mediator or in the scenario without the mediator there exists an equivalent deviation in the other scenario (i.e., a deviation that produces the same outcome). We then use this result to show that any (k, t)-robust strategy in a game with n players and a mediator in an asynchronous system can be implemented with cheap talk if $n > 4k + 4t$. A (k, t)-robust strategy is one in which no coalition of t malicious players can decrease the payoff of anyone else and no coalition of k players can increase their payoff even in coalition with the t malicious players. We also prove that the result above can be satisfied if $n > 3k+4t$ whenever honest players can punish players that are caught deviating. A similar result can be shown for synchronous systems, but in this case we only require that $n > 2k + 3t$.

We also show that for every protocol $\vec{\pi}$ for n players and all $k < n$ there exists a belief system that is consistent with $\vec{\pi}$ in which all coalitions K of at most k players believe that, if there is any deviation during the execution of the protocol, then it was someone sending an *incorrect* message to a player in K. We call these beliefs k-paranoid. Intuitively, k-paranoid beliefs are such that all coalitions of size at most K believe that the remaining players are being honest between them.

We use these beliefs to extend the results regarding (k, t)-robustness by showing that all k-resilient sequential equilibria with a mediator can be implemented with cheap-talk if $n > 4k$ in asynchronous systems or $n > 3k$ in synchronous systems.

We finish this work by proving a matching lower bound for most of our results.

BIOGRAPHICAL SKETCH

Ivan was born in New York city but grew up in Caracas, Venezuela, and then in Barcelona, Spain where he finished high school and then obtained a degreee in Mathematics and a Master in Advanced Mathematics at the Polytechnical University of Catalonia (UPC) in 2013 and 2014 respectively. In 2014, Ivan moved to Ithaca to pursue a Ph.D. in Mathematics at Cornell University. After a couple of years taking courses in Math and Computer Science, he decided to work under the supervision of Joe Halpern on themes related to game theory, distributed systems, and cryptography.

Ivan's main areas of interest include game theory (especially applied to algorithms and distributed systems), theoretical computer science, discrete mathematics and cryptography. He also loves math and programming, and before coming to Ithaca he obtained a number of awards and recognitions in both. He obtained a Gold Medal at the Spanish Mathematical Olympiad (2009) a Bronze Medal at the International Mathematical Olympiad (2009), the First Place at the South-Western European ACM Programming Contest (2013), and a First Prize at the International Mathematical Competition (2013).

Ivan is also devoted teacher. He was actively involved in all the preparation courses for the Catalan and Spanish Mathematical Olympiads and Programming Olympiads since 2009. He is also a co-founder and main developer of AI Coliseum, a platform that runs a yearly programming competition in which teams must code their AI for particular strategic games. The aim of this platform is to engage young people into programming with a fun and challenging activity, and the platform is also used to support the organization of programming courses for high and middle school students. Ivan was a TA at the UPC and at Cornell between 2011 and 2016, where he taught linear algebra, algorithms, and several calculus courses.

ACKNOWLEDGEMENTS

First and foremost I want to thank my advisor, Joe Halpern, for his tireless encouragement, support, and guidance, which allowed me to work on a variety of topics that were of great interest to me.

I would also like to thank my thesis committee, Rafael Pass, Éva Tardos, and Lorenzo Alvisi, for taking the time to read and review this work and for their helpful comments during my A exam.

None of this would be possible without the unconditional love and support of my family and friends, especially Hector and Marina, my dad and his wife, Mª Eugenia, my mom, Claudia, my sister, and Yolanda, Bill, Danny, Sophia, Irene, Ariel, Sol, Almi and Tomas.

I am also extremely grateful to Melissa Totman, the ex-Graduate Field Coordinator of the Math department. Without her, I would have been kicked out of Cornell a long time ago because of delays with paperwork :).

CONTENTS

CHAPTER 1

INTRODUCTION

Having a trusted mediator often makes solving a problem much easier. For example, a problem such as reaching consensus becomes trivial with a mediator: agents can just send their initial input to the mediator, and the mediator sends the majority value back to all the agents, which they then output. Not surprisingly, the question of whether a problem in a multiagent system that can be solved with a trusted mediator can be solved by just the agents in the system, without the mediator, has attracted a great deal of attention in both computer science (particularly in the cryptography community) and game theory. In cryptography, the focus has been on *secure multiparty computation* [22, 29]. Here it is assumed that each agent i has some private information x_i. Fix functions $f_1, \ldots, f_n$. The goal is to have agent i learn $f_i(x_1, \ldots, x_n)$ without learning anything about x_j for $j \neq i$ beyond what is revealed by the value of $f_i(x_1, \ldots, x_n)$. With a trusted mediator, this is trivial: each agent i just gives the mediator its private value x_i; the mediator then sends each agent i the value $f_i(x_1, \ldots, x_n)$. Work on multiparty computation provides conditions under which this can be done in a synchronous system [10, 22, 28, 29] and in an asynchronous system [9, 12]. In game theory, the focus has been on whether an equilibrium in a game with a mediator can be implemented using what is called *cheap talk*—that is, just by players communicating among themselves.

In the computer science literature, the interest has been in performing multiparty computation in the presence of possibly malicious adversaries, who do everything they can to subvert the computation. In contrast, in the game theory literature, the assumption is that players have preferences and seek to maximize

their utility; thus, they will subvert the computation iff it is in their best interests to do so. Abraham, Dolev, Gonen and Halpern [1] (ADGH from now on) argued that it is important to consider deviations by both rational players, who have preferences and try to maximize them, and players that we can view as malicious, although it is perhaps better to think of them as rational players whose utilities are not known by the mechanism designer (or other players). ADGH considered equilibria that are (k, t)-*robust*; roughly speaking, this means that the equilibrium tolerates deviations by up to k rational players, whose utilities are presumed known, and up to t players with unknown utilities. Tight bounds were proved on the ability to implement a (k, t)-robust equilibrium in the game with a mediator using cheap talk in synchronous systems. These bounds depend on, among other things, (a) the relationship between k, t and n, the total number of players in the system; (b) whether players know the exact utilities of the rational players; and (c) whether the game has a *punishment strategy*, where an m-punishment strategy is a strategy profile that, if used by all but at most m players, guarantees that every player gets a worse outcome than they do with the equilibrium strategy. The following is a high-level overview of results proved in the synchronous setting that will be of most relevance here. For these results, we assume that the communication with the mediator is bounded, it lasts for at most N rounds, and that the mediator can be represented by an arithmetic circuit of depth c.

R1. If $n > 3k + 3t$, then a mediator can be implemented using cheap talk; no punishment strategy is required, no knowledge of other agents' utilities is required, and the cheap-talk protocol has bounded running time $O(nNc)$, independent of the utilities.

R2. If $n > 2k + 3t$, then a mediator can be implemented using cheap talk if there is a $(k + t)$-punishment strategy and the utilities of the rational players are

known; the cheap-talk protocol has expected running time $O(nNc)$. (In R2, unlike R1, the cheap-talk protocol may be unbounded, although it has finite expected running time.)

Abraham, Dolev and Halpern [3] (ADH from now on) also presented lower bounds that match the upper bounds above. Thus, for example, it is shown that $n > 3k+3t$ is necessary in R1; if $n \leq 3k+3t$, then we cannot implement a mediator in general if we do not have a punishment strategy or if the utilities are unknown. The proofs of R1 and R2 make heavy use of the fact that the setting is synchronous. Here we consider the impact of asynchrony on these results. Once we introduce asynchrony, we must revisit the question of what it even means to implement an equilibrium using cheap talk. Notions like (Bayesian) Nash equilibrium implicitly assume that all uncertainty is described probabilistically. Having a probability is necessary to talk about an agent's expected utility, given that a certain strategy profile is played. If we were willing to put a distribution on how long messages take to arrive and on when agents are scheduled to move, then we could apply notions like Nash equilibrium without difficulty. However, it is notoriously difficult to quantify this uncertainty. The typical approach used to analyze algorithms in the presence of uncertainty that is not quantified probabilistically is to assume that all the non-probabilistic uncertainty is resolved by the environment according to some strategy. Thus, the environment uses some strategy to decide when each agent will be allowed to play and how long each message takes to be delivered. The algorithm is then proved correct no matter what strategy the environment is following in some class of strategies. For example, we might restrict the environment's strategy to being *fair*, so that every agent eventually gets a chance to move. (See [23] for a discussion of this approach and further references.)

We follow this approach in the context of games. Note that once we fix the environment's strategy, we have an ordinary game, where uncertainty is quantified by probability. In this setting, we consider *ex post equilibrium*. A strategy is an ex post equilibrium if it is an equilibrium no matter what strategy the environment uses. Ex post equilibrium is a strong notion, but, as we show by example, it can often be attained with the help of a mediator. It is arguably the closest analogue to Nash equilibrium in an asynchronous setting.

Another issue that plays a major role in an asynchronous setting is what happens if the strategies of players result in some players being *livelocked*, talking indefinitely without making a move in the underlying game, or in some players being *deadlocked*, waiting indefinitely without moving in the underlying game. We consider two approaches for dealing with this problem. One is called the *default-move approach*. In this approach, as part of the description of the game, there is a default move for each player which is imposed if that player fails to explicitly make a move in the cheap-talk phase. Aumann and Hart [8] considered a different approach, which we henceforth call the *AH approach*, where a player's strategy in the underlying game is a function of the (possibly infinite) history of the player in the cheap-talk phase. We can think of this almost as if the player writes a will, describing what he would like to have done (as a function of the history) if the game ends before he has had a chance to move.

Our results show that, in the worst case, the cost of asynchrony is an extra $k + t$ in the bounds on n, but we can sometimes save k or even $k + t$ if there is a punishment strategy or if we are willing to tolerate an ϵ "error". For example, with both the AH approach and the default-move approach, if the utilities are not known, we can implement a mediator using asynchronous cheap talk if $n > 4k+4t$.

Thus, compared to R1, we need an extra $k+t$. However, if we are willing to accept a small probability of error, we can only implement an ϵ-(k,t)-robustness equilibrium (which, roughly speaking, means that players get within ϵ of the best they could get), then we can do this if $n > 3k+3t$, again, using both the AH approach and the default-move approach. We also prove that, in the general case, the $n > 4k + 4t$ upper bound is tight: if $n \leq 4k + 4t$ there exist (k, t)-robust protocols with a mediator such that there is no (k, t)-robust implementation.

Just as in the synchronous case, we can do better if we assume that there is a punishment strategy and utilities are known (as in R2). Specifically, with the AH approach, we can implement a mediator if $n > 3k + 4t$ (compared to $n > 2k + 3t$ in the synchronous case), and can ϵ-implement a mediator if $n > 2k + 3t$. We use the punishment to deal with deadlock. If a good player is waiting for a message that never arrives, then the waiting player instructs his executor to carry out a punishment in his will. Having a punishment does not seem to help in the default-move approach unless the default move is a punishment; if it is, then we can get the same results as with the AH approach.

If there is a punishment strategy, these results significantly improve those of Even, Goldreich, and Lempel [18]. They provide a protocol with similar properties, but the expected number of messages sent is $O(1/\epsilon)$; with a punishment strategy, we show that a bounded number of messages can be sent, with the bound being independent of ϵ.

These results focus on Nash equilibrium ((k, t)-robustness is a generalization of Nash equilibrium). As is well known, Nash equilibrium considers only deviations on the *equilibrium path* (situations that arise with positive probability if the NE is played). It is well-known that in extensive-form games such as interacting with

a mediator, Nash equilibrium does not always describe what intuitively would be reasonable play. For example if A tells B to give her \$1 or she will destroy the world, B giving \$1 to A and A not destroying the world is a NE. However, this equilibrium is based on a non-credible threat. Would A really destroy the world if she doesn't get \$1? For this reason, game theorists have considered solution concepts such as *sequential equilibrium* [24], where players cannot increase their utility by deviating even off the equilibrium path (see Section 5.1 for a formal definition).

Not surprisingly, the question of whether a sequential equilibrium with a mediator can be implemented has been considered before in the game theory literature. Ben-Porath [13] claimed that a sequential equilibrium could be implemented with cheap talk if $n > 3$ provided that there is a *punishment strategy* (a way for players to punish players who are caught cheating—see Section 4.3 for a formal definition);[1] Gerardi [20] showed that sequential equilibrium can be implemented if $n \geq 5$ (even without a punishment strategy).

Here we improve Gerardi's result by showing that we can implement a k-resilient sequential equilibrium (i.e., a strategy in which it is always rational for coalitions of size at most k to follow the protocol) in a game with a mediator if $n > 3k$ in synchronous systems and if $n > 4k$ in asynchronous systems (Gerardi implicitly assumed a synchronous system). In particular, if $k = 1$ we show that a sequential equilibrium with a mediator can always be implemented if $n \geq 4$. These results are the best possible; it is easy to show that the lower bounds of $n > 3k$ in the synchronous case [3] and $n > 4k$ in the asynchronous case on implementing NE apply without change to implementing a sequential equilibrium.

[1] Unfortunately, there is a serious error in Ben-Porath's proof; see [3].

Following Gerardi [20] and Gerardi and Myerson [21], we also relate our results to two other solution concepts in normal-form games: *correlated equilibrium* [7] and *communication equilibrium* [19, 26]. Both of these concepts can be understood in terms of games with mediators. A correlated equilibrium is an equilibrium in a game with a mediator where the players do not talk to the mediator; the mediator simply tells the agent which strategy to play. A *communication equilibrium* in a Bayesian game is an equilibrium in a game with a mediator where the players can tell the mediator their types, and the mediator then tells the players what strategy to play. (See Section 5.1 for formal definitions.)

Our proof for extending $(k, 0)$-robustness to k-resilient sequential equilibria introduces an interesting new technique. Sequential equilibrium involves describing not only what the players do at each point in an extensive-form game, but describing what their beliefs are, even off the equilibrium path. It must be shown that players are always best responding to their beliefs. The key difficulty in our proof involves finding appropriate beliefs. To this end, we define the notion of a *k-paranoid belief system*, where all coalitions K of at most k players always believe that, if other players deviated, they did so by sending inappropriate messages only to players in K. That is, all coalitions of size at most k believe that the remaining players are being truthful between themselves. We show that, given a $(k, 0)$-robust protocol $\vec{\sigma}$, we can extend it to a k-resilient sequential equilibrium by constructing a k-paranoid belief system. The idea is that, given these k-paranoid beliefs, players in a coalition K will not believe that there is anything that they can do to prevent the remaining players from playing their part of the equilibrium.

Our approach for implementing (k, t)-robust strategies with a mediator generalizes Ben-Or, Canetti and Goldreich's (BCG from now on) notion of *secure*

computation for asynchronous systems. BCG show that, in asynchronous systems, if $n > 4t$, a group of n agents can securely compute any function f while a coalition of at most t malicious agents cannot prevent the honest agents from correctly computing the output of f given their inputs, nor can the malicious agents learn anything about the inputs of the honest agents. Ben-Or, Kelmer and Rabin [12] (BKR from now on) then showed if we are willing to tolerate a small probability $\epsilon > 0$ that the agents do not correctly compute f or that the malicious agents learn something, then we can achieve this if $n > 3t$. BCG and BKR also prove matching lower bounds for their results, showing that we really need to have $n > 4t$ (resp., $n > 3t$).

We can view secure function computation as a one-round interaction with a trusted mediator: each agent sends its input to the mediator, the mediator waits until it receives enough inputs, applies f to these inputs (again, replacing missing inputs with a default value), and sends the output back to the agents, who then output it. We generalize BCG and BKR's results for function computation to a more general setting. Specifically, we want to simulate arbitrary interactions with a mediator, not just function computation. Also, unlike previous approaches, we want the simulation to be "bidirectional": the set of possible output distributions that arise with the mediator must be the same as those that arise without the mediator, even in the presence of malicious agents. More precisely, we show that, given a protocol profile $\vec{\pi}$ for n agents and a protocol π_d for a mediator, we can construct a protocol profile $\vec{\pi}'$ such that for all sets T of fewer than $n/4$ malicious agents, the following properties hold:

(a) For all protocols $\vec{\tau}'_T$ for the malicious agents and all schedulers σ'_e in the setting without the mediator, there exists a protocol $\vec{\tau}_T$ for the agents in T

and a scheduler σ_e in the setting with the mediator such that, for all input profiles $\vec{x}$, the output distribution in the computation with $\vec{\pi}'$, $\vec{\tau}'$, and σ'_e with input $\vec{x}$ is the same as the output distribution with $\vec{\pi} + \pi_d$, $\vec{\tau}$, and σ_e with input $\vec{x}$.

(b) For all protocols $\vec{\tau}_T$ for the malicious agents and all schedulers σ_e in the setting with the mediator, there exists a protocol $\vec{\tau}'_T$ for the agents in T and a scheduler σ'_e in the setting without a mediator such that, for all input profiles $\vec{x}$, the output distribution in the computation with $\vec{\pi}'$, $\vec{\tau}'$, and σ'_e with input $\vec{x}$ is the same as the output distribution with $\vec{\pi} + \pi_d$, $\vec{\tau}$, and σ_e with input $\vec{x}$.

We use the notation $\vec{\pi} + \pi_d$ to indicate that the agents use protocol $\vec{\pi}$ and the mediator uses protocol π_d (we use the subscript d to denote the mediator); we view the mediator as just another agent here. This result implies that arbitrary distributed protocols that work in the presence of a trusted mediator can be compiled to protocols that work without a mediator, as long as there are less than $n/4$ malicious agents. And, just as BKR, if we allow a probability ϵ of error, we can get this result while tolerating up to $n/3$ malicious agents. BCG proved the analogue of (a) for secure function computation, which is enough for security purposes: if there is any bad behavior in the protocol without the mediator, this bad behavior must already exist in the protocol with the mediator. However, (b) also seems like a natural requirement; if a protocol satisfies this property, then all behaviors in the protocol with the mediator also occur in the protocol without the mediator.

Property (b) is typically not required in security papers. It plays a critical role in proving our results regarding (k, t)-robustness, but we believe it may be of independent interest. Requiring only (a) may result in protocols where outcomes

that may be likely in the mediator setting do not arise at all. This is especially relevant in asynchronous systems, since by requiring only (a) we are implicitly assuming that the adversary has total control over the scheduler. However, it may be the case that the scheduler acts randomly or that is even influenced by honest agents. For instance, suppose that a group of n agents wants to check who has the fastest internet connection. To do this, each agent pings the server and waits for the server's response. The server (who we are viewing as the mediator) waits until the first ping arrives, then sends a message to each agent saying which agent's ping was received first. In this example, the scheduler determines the lag in the system. If we wanted to simulate this interaction without the mediator but requiring only property (a), even with no malicious agents, a protocol profile in which every honest agent does nothing and outputs 1 would suffice. But, intuitively, this implementation does not capture the behavior of the server. Similar examples exist even in the case of function evaluation. Suppose a group of n congressmen vote remotely (by sending a vote to a trusted third party) to either pass or not a bill that requires support from at least 90% of them. We can view this as a multiparty computation of a function f in which each agent has input 0 (vote against) or 1 (vote for), and the output is either 0 (reject the bill) or 1 (pass the bill) depending on how many agents had input 1 (agents that do not submit input count as 0). In this case, a protocol in which every agent does nothing and outputs 0 securely computes f while tolerating up to $n/4$ malicious agents. To see this, note that regardless of the adversary, the scheduler can delay $n/4$ of the players until everyone else has finished the computation. This is indistinguishable from $n/4$ agents deviating from the protocol and submitting no input. However, again, this protocol does not capture the intended behavior of the voting process. By way of contrast, a protocol that bisimulates f would come closer to capturing the

intended behavior of the voting process.

Clearly, the results of BCG and BKR are special cases of our result. However, in general, our results do not follow from those of BCG/BKR, as is shown in Section 3.1. Specifically, the results of BCG/BKR do not give us property (b), since the outcome can depend on the behavior of the scheduler. For example, consider protocols for two agents and a mediator m in which each agent sends its input to the mediator, the mediator m sends to each agent the first message it receives, and each agent outputs whatever they receive from the mediator. Let σ_e^i be the scheduler that delivers the message from agent i first, for $i = 1, 2$. It is easy to check that if the agents have inputs 0 and 1, respectively, and play with mediator σ_e^1, then they both output 0, while if they play with σ_e^2, then they both output 1. This means that, unlike secure function computation, even if all the agents are honest, the distribution over the agents' outputs can depend on the scheduler's protocol, not just the agents' inputs.

Even though our results do not follow from those of BCG/BKR, our proofs very much follow the lines of those of BCG/BKR. However, there are some new subtleties that arise in our setting. In particular, as the example above shows, when we try to implement the setting with the mediator, the agents must somehow keep track of the scheduler's possible behaviors. Doing this adds nontrivial complexity to our argument.

Besides the main result, we also show that our protocol without the mediator has two additional security properties, which may be of independent interest. Specifically, we show that the following two properties hold for coalitions of malicious agents of size at most $t < n/3$.

(P1) The only way malicious agents can disrupt the computation is by preventing honest agents from terminating; if an honest agent terminates, then its output is correct.

(P2) If $2t + 1$ or more honest agents terminate, then all honest agents terminate. That is, either all the honest agents terminate or a nontrivial number of honest agents (more than $n - 2t$) do not terminate.

If we allow an ϵ probability of error, we get analogous results if we have $n > 2t$ rather than $n > 3t$. We remark that these two properties are in fact also satisfied by BCG's and BKR's implementations, but they do not prove this (or even state the properties explicitly).

From the game-theoretic viewpoint, the second property guarantees that either all the honest agents terminate, or sufficiently many of them do not terminate so as to guarantee that rational agents will not try to prevent honest agents from terminating (due to the threat of punishment). The first property above guarantees that if all the honest agents terminate, their output will be correct. The lower bound for bisimulating arbitrary protocols follows immediately from that of BCG, which we also provide a proof of.

1.1 The Synchronous and Asynchronous Models

The asynchronous model used throughout this document is that of an asynchronous network in which every pair of agents can communicate through a private, authenticated and reliable communication channel. For most of our results, we assume that all messages sent through any of these channels are eventually received, but

they can be delayed arbitrarily. The order in which these messages are received is determined by the *environment* (also called the *scheduler*). The scheduler also chooses the order in which the agents are scheduled. For some of the results of this paper, we drop the condition that all messages must be eventually delivered. We call these more general schedulers *relaxed schedulers*.

Whenever an agent is scheduled, it reads all the messages that it has received since the last time it was scheduled, sends a (possibly empty) sequence of messages, and then performs some internal actions. We assume that the scheduler does not deliver any message or schedule other agents during an agent's turn. Thus, although agent i does not send all its messages simultaneously when it is scheduled, they are sent atomically, in the sense that no other agent is scheduled while i is scheduled, nor are any messages delivered while i is scheduled. Note that the atomicity assumption is really a constraint on the scheduler's protocol.

More precisely, consider the following types of *events*:

- $sch(i)$: Agent i gets scheduled.
- $snd(\mu, j, i)$: Agent i sends a message μ to agent j.
- $rec(\mu, j, i)$: Message μ sent by j is received by i. The message μ must be one sent at an earlier time to i that was not already received.
- $comp(v, i)$: Agent i locally computes value v.
- $out(s, i)$: Agent i outputs string s.
- $done(i)$: i is done sending messages and performing computations (for now).

For simplicity, we assume that agents can output only strings in $\{0, 1\}^*$. Note that all countable sets can be encoded by such strings, and thus we can freely talk about

agents being able to output any element of any countable set (for instance, elements of a finite field $\mathbb{F}_q$) by assuming that they are actually outputting an encoding of these elements. We also assume that at most one event occurs at each time step. Let $h(m)$ denote a *global history* up to time m: a sequence that starts with an input profile $\vec{x}$, followed by the ordered sequence of events that have occurred up to and including time m. We assume that the only events between events of the form $sch(i)$ and $done(i)$ are ones of the form $snd(\mu, j, i)$ and $comp(v, i)$. This captures our atomicity assumption. We do not include explicit events that correspond to reading messages. (Nothing would change if we included them; they would simply clutter the notation.) Message delivery (which is assumed to be under the control of the scheduler) occurs at times between when agents are scheduled. We can also consider the subsequence involving agent i, namely, i's initial state, followed by events of the form $sch(i)$, $snd(\cdot, \cdot, i)$, $comp(\cdot, i)$, $rec(\cdot, \cdot, i)$, and $done(i)$. This subsequence is called i's *local history*. We drop the argument m if it can be deduced from context or if it is not relevant (for instance, when we consider the local history of an agent after a particular event).

Agent i moves only after a $sch(i)$ event. What it does (in particular, the order in which i sends messages) is determined by i's protocol, which is a function of i's local history. The scheduler moves after an action of the form $done(i)$ or $rec(\cdot, \cdot, i)$. It is convenient to assume that the scheduler is also running a protocol, which is also a function of its local history. Since the scheduler does not see the contents of messages, we can take its history to be identical to $h(m)$, except that *comp* events and the contents of the messages in *snd* and *rec* events are removed, although we do track the index of the messages delivered; that is, we replace events of the form $snd(\mu, i, j)$ and $rec(\mu, i, j)$ by $snd(i, j)$ and $rec(i, j, \ell)$, where ℓ is the index of the message sent by i to j in $h(m)$. For instance, $rec(i, j, 2)$ means that the second

message sent by i to j was delivered to j. Note that the scheduler does see events of the form $done(i)$; indeed, these are signals to the scheduler that it can move, since i's turn is over. Since we view the agents (and the mediator) as sending messages atomically, in the sequel, we talk about an agent's (or the mediator's) *turn*. An agent's kth turn takes place the kth time it is scheduled. During its turn, the agent sends a block of messages and performs some local computation.

It is more standard in the literature to assume that agents perform at most one action when they are scheduled. We can view this as a constraint on agents' protocols. A *single-action* protocol for agent i is one where agent i sends at most one message before performing the $done(i)$ action. As we show in Section 3.4, we could have restricted to single-action protocols with no loss of generality as far as our results go; allowing agents to perform a sequence of actions atomically just makes the exposition easier.

The synchronous model is analogous to the asynchronous one, except that computation proceeds in *rounds* of communication and all messages that are sent in a given round are guaranteed to be received by the recipient at the beginning of the next round.

1.2 Canonical and responsive protocols

We deal only with bounded protocols, where there is a bound N on the number of messages that an honest agent sends. Of course, there is nothing to prevent malicious agents from spamming the mediator and sending an arbitrary number of messages. We assume that the mediator reads at most N messages from each agent i, ignoring any further messages sent by i.

For our results involving termination, specifically, (P2), it is critical that agents know when the mediator stops sending messages. For these results, we restrict the honest agents and the mediator to using protocols that have the following *canonical form*: Using a canonical protocol, each honest agent tags its ℓth message with label ℓ and all honest agents are guaranteed to send at most N messages regardless of their inputs or the random bits they use. Whenever the mediator receives a message from an agent i, it checks its tag ℓ; if $\ell > N$ or if the mediator has already received a message from i with tag ℓ, it ignores the message. The mediator is guaranteed to eventually terminate. Whenever this happens, it sends a special "STOP" message to all agents and halts. Whenever an honest agent receives a "STOP" message, it terminates.

Even though canonical protocols have a bound N on the number of messages that honest agents and the mediator can send, the mediator's local history in a canonical protocol in asynchronous systems can be arbitrarily long, since it can be scheduled an arbitrary number of times. We conjecture that, in general, since the message space is finite, the expected number of messages required to simulate the mediator in asynchronous systems is unbounded. However, we can do better if the mediator's protocol satisfies two additional properties. Roughly speaking, the first property says that the mediator can send messages only either at its first turn or in response to an agent's message; the second property says that the mediator ignores *empty turns*, that is, turns where it does not receive or send messages. More precisely, the first property says that whenever the mediator π_d is scheduled with history h_d, then if $h_d \neq (\,)$ (i.e., if h_d is not the initial history) or if the mediator has not received any messages in h_d since the last time it was scheduled, then $\pi_d(h_d) = done(d)$. The second property says that $\pi_d(h_d) = \pi_d(h'_d)$, where h'_d is the result of removing consecutive $(done(d),\ sch(d))$ pairs in h_d (e.g., if $h_d =$

$(sch(d), snd(\mu, j, d), done(d), sch(d), done(d), rec(\mu', i, d), sch(d), done(d), sch(d))$, then $h'_d = (sch(d), snd(\mu, j, d), done(d), rec(\mu', i, d), sch(d)))$. An asynchronous protocol for the mediator that satisfies these two properties is called *responsive*. In Section 3.3.4, we show that if the mediator uses a responsive protocol π_d that can be represented using a circuit with c gates, then we can simulate all protocol profiles $\vec{\pi} + \pi_d$ in such a way that the expected number of messages sent by honest agents during the simulation is polynomial in n and N and linear in c.

1.3 Adversaries in Asynchronous Systems

We define an adversary as a tuple $(T, \vec{\tau}_T, \sigma_e)$ consisting of a subset of malicious agents T, a strategy $\vec{\tau}_T$ for agents in T, and a strategy σ_e for the environment (in synchronous systems we omit σ_e). It may seem *a priori* that malicious agents and the environment act independently, but in fact, we can assume without loss of generality that they are all under the control of a single adversary. Clearly malicious agents can coordinate by sending messages to each other. They can also coordinate with the environment so that, for example, the malicious agents can act knowing who will be scheduled when, and the environment can also schedule malicious agents based on their inputs.

To see that an agent i can communicate with the environment, recall that we have assumed that the message space is finite, say $\{m_0, \ldots, m_M\}$. Immediately after sending m_j, i sends j empty messages to itself. So, even though the environment cannot read the messages, it will know that i sent message m_j. [2] (Clearly the

[2]We can encode m_j using using fewer messages by having i send message to other agents as well as itself. Our goal here is not to minimize the number of messages, but to show that communication with the environment is possible.

environment will also know who sent the message, since the environment delivered the message.) Thus, we can assume without loss of generality that the malicious agents know the environment's protocol (and thus know when a message that is sent will be delivered), hence it suffices for the environment to be able to tell the malicious agents when the kth message is sent, who sent it, and who the intended recipient is. It does that as follows: malicious agents i initially send themselves $(n+1)^2$ empty messages. If the first message was sent by agent j_1 to agent j_2 (treating the mediator as agent 0 in the mediator game), then the environment delivers $(n+1)j_1 + j_2$ of these empty messages. Then agent i sends another $(n+1)^2$ empty messages to itself, allowing the environment to encode the sender and receiver of the next message, if there is one.

As shown by the example above, from now on we will assume without loss of generality that the adversary is controlled by a single entity that knows the state of the scheduler and of all malicious agents in real time.

CHAPTER 2

SECURE COMPUTATION

2.1 Basic Definitions

The first attempt at simulating a mediator comes from Ben-Or, Goldwasser and Widgerson's *secure computation* [11]. Although their definition does not involve a mediator at all, we show in Section 3.1 how it relates to a particular type of mediator game.

For the main definitions in this section, we need the following notation, largely taken from BCG [9] and BGW. Given a finite domain D, let $\vec{x}$ be a vector in D^n. Given a set $C \subseteq [n]$, denote by $\vec{x}_C$ the vector obtained by projecting $\vec{x}$ onto the indices of C. Also, given a $|C|$-vector $\vec{z} \in D^{|C|}$, let $\vec{x}_{(B,\vec{z})}$ be the vector obtained by replacing the entries of $\vec{x}$ indexed by B by $\vec{z}$. To simplify notation, given a function $f : D^n \to D$, we write $f_C(\vec{x})$ rather than $f(\vec{x}_{(\overline{C},\vec{0}_{|C|})})$ to denote the output of evaluating f on $\vec{x}$ with the entries in $\vec{x}$ not indexed by an element of C replaced by 0.

Secure computation is concerned with jointly computing a function f on n variables, where the ith input is known only to agent i. For instance, if we want to compute the average salary of the people from the state of New York, then n would be New York's population, the input x_i is i's salary, and $f(x_1, \ldots, x_n) = \frac{\sum_{i=1}^{n} x_i}{\sum_{x_i \neq 0} 1}$. (For the denominator we count only people who are actually working.) Ideally, a secure computation protocol that computes f would be a protocol in which each agent i outputs $f(x_1, \ldots, x_n)$ and gains no information about the inputs x_j for $j \neq i$. In our example, this amounts to not learning other people's salaries.

Typically, we are interested in performing secure computation in a setting where some of the agents might be malicious and not follow the protocol. In particular, they might not give any information about their input or might just pretend that they have a different input (for instance, they can lie about their salary). What output do we want the secure computation of f to produce in this case? Roughly speaking, the idea is to accept as correct any output of f that can be obtained from an input profile that differs from the actual input profile in at most t coordinates, which are set to some default value, generally 0. (Intuitively, these coordinates are ones corresponding to inputs of malicious agents who did not submit a value or lied about their actual input.) More precisely, a protocol $\vec{\pi}$ t-securely computes f in synchronous systems if for every coalition T of at most t malicious agents, there exist functions $h : D^{|T|} \to D^{|T|}$ and $O : D^{|T|} \times D \times T \to \{0,1\}^*$ such that, for each input $\vec{x}$, (a) each agent $i \notin T$ outputs $f(\vec{x}/_{(T,h(\vec{x}_T)})$, and (b) each agent $i \in T$ outputs $O(\vec{x}_T, f(\vec{x}/_{(T,h(\vec{x}_T)}), i)$. Note that h and O encode how malicious agents might lie about their inputs (if a malicious agents does not participate in the computation, its input is assumed to be a fixed value $x_0 \in D$) and what they output, respectively. We thus consider an output to be correct if only the inputs of agents in T used in the computation of f differ from their actual inputs, and if the output of malicious agents output is just a function of the output of f and their own inputs. Note that this last requirement captures the fact that malicious agents do not learn anything besides the (honest agents') output of the secure computation protocol, since otherwise they could use this extra information to generate outputs that cannot be written as such a function O. Since malicious agents can randomize, we assume that both h and O have an extra input r sampled from a distribution $\mathcal{R}$, and require that agent i's output is distributed identically to $f(\vec{x}/_{(T,h(\vec{x}_T)})$ or $O(\vec{x}_T, f(\vec{x}/_{(T,h(\vec{x}_T)}), i)$, depending on whether i is honest. (See

Definition 1 for the more standard formalization of this property.) BGW proved the following result:

Theorem 1. *[11] If D is a finite domain, $n > 3t$, and $f : D^n \to D$, then there exists a protocol $\vec{\pi}$ that t-securely computes f in synchronous systems.*

Subtleties introduced by asynchrony make the definition of secure computation slightly more involved in asynchronous systems. In asynchronous systems, as is standard, we assume that there is a scheduler that decides the order in which agents act, and how long it takes for a message to be delivered. The existence of adversaries such as those described in Section 1.3 implies that there are deviations that are possible in asynchronous systems that are not possible in synchronous systems; specifically, the scheduler can delay a subset of agents until the other agents terminate the protocol. If the number of agents delayed is less than the number of malicious agents that the protocol tolerates, delayed honest agents are indistinguishable from malicious agents that never engage in the communication, and thus the remaining agents must be able to terminate regardless of the delay. Since the inputs of delayed honest agents are not taken into consideration, the adversary can choose a set $C \subseteq [n]$ of size at least $n - t$ and force the computation to ignore the inputs of agents not in C (they are replaced by a fixed input).

To define asynchronous secure computation, BCG introduced another type of adversary that they called a *trusted-party adversary*. A t-trusted-party adversary is defined as a quadruple $A = (T, h, c, O)$ where

- T is the set of malicious agents;
- $h : D^{|T|} \times \mathcal{R} \to D^{|T|}$ is the input substitution function (again, $\mathcal{R}$ is a distribution of possible random inputs);

- $c : D^{|T|} \times \mathcal{R} \to \{C \subseteq [n] \mid |C| \geq n - t\}$ is a subset of agents (intuitively, the ones whose inputs are taken into consideration);
- $O : D^{|T|} \times \mathcal{R} \times D \times T \to \{0,1\}^*$ is the output function for the malicious agents.

In the sequel, we use "trusted-party adversary" or "ideal adversary" to refer to such a tuple (T, h, c, O), and reserve the term adversary for a tuple of the form $(A, \vec{T}_T, \sigma_e)$, as defined in Section 1.3.

Given a function $f : D^n \to D$, a trusted-party adversary $A = (T, h, c, O)$, and an input vector $\vec{x}$, let $C = c(\vec{x}_T, r)$ and $\vec{y} = \vec{x}/_{(T,h(\vec{x}_T,r))}$. Intuitively, C is the set of agents whose inputs are considered and $\vec{y}$ is the input profile obtained by replacing the actual inputs of agents in T with the output of h. The output of f with trusted-party adversary A and input $\vec{x}$ is an n-vector of random variables $\vec{\rho}(A, \vec{x}; f)$ such that

$$\rho_i(A, \vec{x}; f) = \begin{cases} (C, f_C(\vec{y})) & \text{if } i \notin T \\ O(\vec{x}_B, r, f_C(\vec{y}), i) & \text{if } i \in T. \end{cases}$$

Note that the outputs of trusted-party adversaries are analogous to the outputs of secure computation in the synchronous case, except that here we must take into account the subset C of agents that provide their inputs. In asynchronous systems, secure computation is defined as follows:

Definition 1 (Secure computation)**.** *Let $f : D^n \to D$ be a function of n variables over some finite domain D. The protocol $\vec{\sigma}$* t-securely computes f in an asynchronous setting *if the following hold for all (standard) adversaries $A = (T, \vec{\tau}_T, \sigma_e)$ with $|T| \leq t$:*

SC1. on all inputs, agents not in T terminate the protocol with probability 1;

SC2. there exists a t-trusted-party adversary $A^{tr} = (T, h, c, O)$ *such that, for all inputs* $\vec{x} \in D^n$. *we have* $\vec{\sigma}(\vec{x}, A) \sim \vec{\rho}(A^{tr}, \vec{x}; f)$ *(i.e.,* $\vec{\sigma}(\vec{x}, A)$ *and* $\vec{\rho}(A^{tr}, \vec{x})$ *are identically distributed).*

In other words, a protocol $\vec{\sigma}$ t-securely computes some function f if it terminates with probability 1 and there exists a trusted-party adversary such that that, for all inputs, gives the same distribution over outputs. BCG proved the following result. Note that BCG just require that *some* ideal t-adversary A gives the same distribution over the the outputs of π. This captures the idea that all deviations that malicious agents can effectively make are the ones already modeled by the adversaries. Also note that SC1 follows from SC2 if we consider non-termination as a special kind of output. BCG proved the following:

Theorem 2. *[9] If* D *is a finite domain,* $n > 4t$, *and* $f : D^n \to D$, *then there exists a protocol* $\vec{\pi}$ *that* t*-securely computes* f *in asynchronous systems.*

2.2 Implementation

In this section, we give an overview of BGW and BCG's secure computation implementation for Theorems 1 and 2. We begin by briefly reviewing the properties of a number of well-known primitives used in their constructions, which we also use for our main results.

2.2.1 Tools

Broadcast

A broadcast protocol involves a *sender* who sends a message μ to all agents in such a way that all honest agents receive the same message. (Although we talk about "a broadcast protocol", this is really a joint protocol, that is, a protocol for each agent. Given a joint protocol $\vec{P}$, we use P_i to denote i's part of the protocol. The sender's protocol is different from those of the other agents. The sender has input μ, the message to be shared; the other agents have no input.) Moreover, if the sender is honest, the message received by an agent i must be the message μ that the sender sent. More precisely, a *broadcast* protocol invoked by a sender with input μ must satisfy the following properties in all histories:

- If an honest agent terminates *broadcast* with output μ', then all honest agents eventually terminate *broadcast* with output μ'.
- If the sender is honest, then all honest agents eventually terminate *broadcast* and output μ.

Bracha [14] provides a broadcast protocol that tolerates up to t malicious agents in asynchronous systems if $n > 3t$.

Consensus

In a consensus protocol, each agent i starts with an initial preference $y_i \in \{0, 1\}$ and must output a value $x \in \{0, 1\}$ such that the following properties are satisfied in all histories:

- All honest agents terminate with probability 1.
- If one honest agent terminates and outputs x, then all honest agents terminate and output x.
- If all honest agents have the same initial value y, then if an honest agent i terminates the protocol, i outputs y.

Abraham, Dolev and Halpern [2] provide a consensus protocol that is t-resilient in asynchronous systems if $n > 3t$.

Verifiable secret sharing

In a verifiable secret sharing protocol, a sender starts out with some secret s that it wants to share. VSS consists of a pair of protocols $(\overrightarrow{\text{VSS}}^{sh}, \overrightarrow{\text{VSS}}^{rec})$, commonly referred to as the *sharing protocol* and the *reconstruction protocol*, and a designated agent, the *sender*, such that the following properties hold:

- If the sender is honest, then every honest agent i will eventually complete VSS_i^{sh}.
- If an honest agent i completes VSS_i^{sh}, then all honest agents j eventually complete VSS_j^{sh} and VSS_j^{rec}.
- The output of VSS_i^{sh} is called i's *share* of the secret. There is a unique value s' such that if each honest agent i runs VSS_i^{rec} with input i's share of the secret, then all the honest agents j will complete VSS_j^{rec}, and will output the same value s', no matter what the malicious agents do.
- If the sender is honest, then $s' = s$ (the sender's secret).

- If the sender is honest and no honest agent i has begun executing VSS_i^{rec}, then the malicious agents cannot guess s with probability $> 1/M$ (where M is the cardinality of the space of possible secrets).

With VSS, just as with the broadcast protocol, the sender's protocol is different from that of the other agents; only the sender has the secret s. Whenever a recipient i receives a message μ from the sender, it invokes VSS_i^{sh} with input μ and outputs its share of the secret, which becomes the input to VSS_i^{rec}. Even though we require each agent i to output the same value s' after runing VSS_i^{rec}, a simple modification of $\overrightarrow{\mathrm{VSS}}^{rec}$ allows a single agent to learn the secret, without any other agent getting any additional information: If we want only i to learn the secret, all the agents send their shares to i, and i simulates the computation of $\overrightarrow{\mathrm{VSS}}^{rec}$ locally. (This depends on the assumption that the only input to VSS_j^{rec} is j's share of the secret, and that it suffices for i to learn the shares of the honest agents in order to recover the secret.) However, no other agents learn anything about the secret (since all they have is their share of the secret).

BCG provide a VSS protocol in an asynchronous setting that is t resilient as long as $n > 4t$. BKR showed that if $n > 3t$, then for all $\epsilon > 0$, there exists a t-resilient protocol that achieves the VSS properties in asynchronous systems with probability at least $1 - \epsilon$. More precisely, their protocol has the property that if some honest agent terminates, then all honest agents terminate and all the properties above hold, and some honest agent terminates with probability at least $1 - \epsilon$.

Accumulative sets

Suppose that we have a global clock, initialized to 0. We do not assume that agents have access to the global clock. An *accumulative set* is a function $U(h, m)$ from histories and global time to sets such that $U(h, m) \supseteq U(h, m')$ if $m \geq m'$. (Intuitively, $U(h, m)$ consists of the elements of U at time m in history h.)

Definition 2. *Given $M_1, M_2 \in \mathbb{N}$ with $M_1 \leq M_2$, a tuple $(U_1, \ldots, U_n)$ of accumulative subsets of $\mathbb{N}$ (one for each agent) is* (M_1, M_2)-uniform *in history h if, for every agent i that is honest in h,*

- *$U_i(h, m) \subseteq \{1, \ldots, M_2\}$ for all times $m \geq 0$;*
- *there exists a time m_i^h such that $|U_i(h, m_i^h)| \geq M_1$;*
- *for all agents j that are honest in h, there exists a time $m_{i,j}^h$ such that $U_i(h, m) = U_j(h, m)$ for all $m \geq m_{i,j}^h$.*

To see how (M_1, M_2)-uniform accumulative sets are used, suppose that each agent i in a system of n agents has a secret s_i. The n agents each invoke t-resilient VSS concurrently in a system with t malicious agents and $n > 3t$, with agent i acting as the sender with secret s_i in its invocation of VSS. Let $U_i(h, m)$ consist of those agents j for which i has terminated the sharing phase of the VSS initiated by j by time m in history h. Clearly U_i is an accumulative set. We claim that $(U_1, \ldots, U_n)$ is $(n-t, n)$-uniform. Clearly, $U_i(h, m) \subseteq \{1, \ldots, n\}$ for all times m by construction. Since there at most t malicious agents in each history and the VSS scheme is t-resilient, the properties of VSS guarantee that each honest agent i will eventually complete the VSS initiated by each honest agent j, which means j is included in $U_i(h, m)$ for some m, and thus there must exist a time m_i^h such that

$|U_i(h, m_i^h)| \geq n - t$. Since $U_i(h, m)$ is finite, there must come a time $(m_i^h)^*$ such that $U_i(h, m') = U_i(h, (m_i^h)^*)$ for all $m' \geq (m_i^h)^*$. Let $m_{i,j}^h = \max((m_i^h)^*, (m_j^h)^*)$. The properties of VSS guarantee that $j' \in U_i(h, m_{i,j}^h)$ iff $j' \in U_j(h, m_{i,j}^h)$.

Agreement on a core set

An agreement on a core set (ACS) protocol is given as input natural numbers M_1 and M_2. Each agent i is also assumed to have access to an accumulative set U_i. If the tuple $(U_1, \ldots, U_n)$ is (M_1, M_2)-uniform with respect to the histories of the ACS protocol, then the following properties must hold:

- All honest agents must eventually complete the ACS protocol.
- If an honest agent i completes the protocol at time m, then it output a set $C_i \subseteq U_i(m)$ such that $|C_i| \geq M_1$.
- If i and j are honest, then $C_i = C_j$.

Thus, all honest agents running an ACS protocol must output the same set; this set is called the *core set*. We denote by $ACS_i(U_i, M_1, M_2)$ agent i's invocation of the ACS protocol with inputs M_1 and M_2 relative to accumulative set U_i. Note that although the notation suggests that U_i is the input to ACS_i, the protocol may actually check U_i several times while it is running, and U_i may be different each time it is checked, since U_i may updated in parallel with ACS_i.

BCG provide an ACS protocol that is t-resilient in asynchronous systems if $n > 3t$.

Circuit computation

Another key primitive that we use is circuit computation. Let $(\overrightarrow{\text{VSS}}^{sh}, \overrightarrow{\text{VSS}}^{rec})$ be a VSS scheme, and let $f : \mathbf{F}_p^N \to \mathbf{F}_p$ be a circuit with N inputs consisting only of addition and multiplication gates. Suppose that each agent i has shares $x_1^i, x_2^i, \ldots, x_N^i$ of secrets $x_1, \ldots, x_N \in \mathbf{F}_p$ respectively (where the secrets are computed using $(\overrightarrow{\text{VSS}}^{sh}, \overrightarrow{\text{VSS}}^{rec})$). A *circuit computation of f (relative to $(\overrightarrow{\text{VSS}}^{sh}, \overrightarrow{\text{VSS}}^{rec})$)*, denoted $CC(f)$ (we suppress the dependence on $(\overrightarrow{\text{VSS}}^{sh}, \overrightarrow{\text{VSS}}^{rec})$ from now on) has the following properties. We assume that there is an input $x_1, \ldots, x_N$ such that each agent i has shares $x_1^i, \ldots, x_N^i$ of $x_1, \ldots, x_N$. Agent i's component of the protocol, denoted $CC_i(f)$, is given the inputs $x_1^i, x_2^i, \ldots, x_i^N$ and computes a single output y_i, such that the following properties hold:

- y_i is i's share of $f(x_1, \ldots, x_N)$ (relative to $(\overrightarrow{\text{VSS}}^{sh}, \overrightarrow{\text{VSS}}^{rec})$).
- After running $CC_j(f)$ with inputs $x_1^j, \ldots, x_N^j$ (but before running the reconstruction protocol $\overrightarrow{\text{VSS}}^{rec}$), no malicious agent j has any information about the shares x_l^i of an honest agent i, the values $x_1, \ldots, x_N$, or $f(X_1, \ldots, x_N)$ beyond what it had before running $CC_j(f)$, even if all the malicious agents pool their information.
- Even after honest agents run $\overrightarrow{\text{VSS}}^{rec}$, no malicious agent j can guess the values of the shares x_l^i of an honest agents i or the the secrets $x_1, \ldots, x_N$ any better than it could before running $CC_j(f)$ if it were given $f(x_1, \ldots, x_N)$.

Simply put, a circuit computation protocol CC allows agents to compute their share of the output of an arithmetic circuit given their shares of the circuit's inputs, without revealing any information.

Since it is well known that every function $f : D^N \to D$ can be represented by a circuit $f' : \mathbf{F}_p^N \to \mathbf{F}_p$ for a prime $p \geq |D|$ (viewing the elements of D as the first $|D|$ elements of $\mathbf{F}_p$), if we can define a protocol $CC(f)$ for all arithmetic circuits, then we can define a protocol $CC(f)$ for all functions $f : D^N \to D$. This is especially important in the next section, where we use CC to compute functions whose inputs and outputs are local histories.

BCG provide an implementation of $CC(f)$ for all arithmetic circuits f relative to the VSS protocol that they provide that is t-resilient in asynchronous systems as long as $n > 4t$; given $\epsilon > 0$, BKR provide an implementation of $CC(f)$ for all arithmetic circuits relative to the VSS protocol that they provide that is t-resilient in asynchronous systems and has at most an ϵ probability of error (i.e., there is a probability ϵ that agents remain in deadlock or the output of the computation will not be the appropriate share of the circuit's output) as long as $n > 3t$.

We can assume without loss of generality that CC can handle randomized functions. That is, if there is a protocol $CC(f)$ to securely compute every deterministic function $f : D^N \to D$, then there is a protocol $CC(f)$ to securely compute every randomized function $f : D^N \to D$. A randomized function $f : D^N \to D$ can be viewed as a deterministic function once it is given sufficiently many random bits, that is, it can be identified with a deterministic function $f : D^N \times \{0,1\}^{N'} \to D$ for N' sufficiently large. Using ACS, VSS, and (deterministic) CC, the agents can easily compute shares for N' random bits as follows.

1. Each agent i chooses a random bit b_i and shares it using VSS.
2. Using ACS, the agents agree on a common set C consisting of at least $t + 1$ agents who correctly shared a bit b_i at step 1. Set $b := \oplus_{i \in C} b_i$ (where $\oplus$ denotes sum mod 2).

3. Each agent i computes its share of b using CC.

If $n > 2t$, then there are at least $t+1$ honest agents, so each honest agent will get shares from at least $t+1$ agents. Since the set of $t+1$ agents agreed on using ACS contains at least one honest agent, the bit b must be truly random.

We can also assume without loss of generality that whenever an honest agent terminates a CC computation of some function $f(x_1, \ldots, x_N)$, even in the presence of at most t malicious agents, at least $n-2t$ other honest agents i have computed their share y_i of $f(x_1, \ldots, x_N)$. This can be ensured by having an honest agent i send a *Ready* message to all agents when it finishes the computation of y_i, and terminating the CC procedure when it receives $n-t$ *Ready* messages. If there are at most t malicious agents, if an agent receives $n-t$ *Ready* messages, at least $n-2t$ are from honest agents who genuinely computed their own share. This property will be critical later, since it guarantees that sufficiently many honest agents are running the protocol at roughly the same pace.

2.2.2 BGW's construction of $\vec{\pi}^f$

Using the primitives sketched above, BGW gave a construction of $\vec{\pi}^f$. At the high level, the construction proceeds as follows:

1. Each agent i shares its input using VSS.
2. Let C be the set of agents that shared a value in step 1. Each agent i computes its share of $f_C(\vec{x})$ using CC, where i's input for the jth input gate is i's share of x_j if j shared an input on step 1; otherwise it is 0.

3. Each agent i sends its share of $f_C(\vec{x})$ to each other agent j, then uses the shares received from other agents to reconstruct $f_C(\vec{x})$ using VSS.

4. Each agent i outputs $f_C(\vec{x})$.

2.2.3 BCG's construction of $\vec{\pi}^f$

BCG's construction is similar to BGW's, except that the subset C of agents that didn't share an input on step 1 is not well defined since in asynchronous systems an agent cannot tell the difference between an agent not sending a message or the message being delayed. In this case, players reach a consensus on C using an ACS protocol. More precisely, the construction goes as follows:

1. Each agent i shares its input using VSS.

2. Agents agree on a core set $C \subseteq [n]$ with $|C| \geq n - t$ using an ACS procedure with parameters $M_1 = n - t$ and $M_2 = n$, where the accumulative set U_i of agent i is the set of agents j such that i has terminated the VSS invoked by j at step 1.

3. Each agent i computes its share of $f_C(\vec{x})$ using CC, where i's input for the jth input gate is i's share of x_j if $j \in C$; otherwise it is 0.

4. Each agent i sends its share of $f_C(\vec{x})$ to each other agent j, then uses the shares received from other agents to reconstruct $f_C(\vec{x})$ using VSS.

5. Each agent i outputs $(C, f_C(\vec{x}))$.

CHAPTER 3

SIMULATING MEDIATORS

Over the following sections we will assume that the system we are working on is asynchronous unless explicitly stated otherwise. In most of the cases the equivalent results and properties for the synchronous case are obtained analogously.

3.1 Secure computation and mediators

Secure computation is closely related to mediators. Even though it is not explicitly proven by BCG, their construction of $\vec{\pi}^f$ satisfies an extra property that we call SC3, which is essentially a converse of SC2.

SC3. For all ideal t-adversaries $A = (T, c, L, \vec{O})$, there exists a strategy $\vec{\tau}_T$ for agents in T and a scheduler σ_e such that, for all inputs $\vec{x}$, $\vec{\rho}(\vec{x}, A; f)$ and $\vec{\pi}(\vec{x}, T, \vec{\tau}_T, \sigma_e)$ are identically distributed.

Lemma 1. *Given a function $f : D^n \to D$, protocol $\vec{\pi}^f$ satisfies SC3.*

Proof. Suppose A is deterministic (i.e., it does not depend on r). Given $\vec{x}_T$, let $C := c(\vec{x}_T)$ and $\vec{y} = L(\vec{x}_T)$ (note that we are dropping the r input in both functions since both are independent of r). Consider the strategy $\vec{\tau}_T$ such that, if the agents in T have input $\vec{x}_T$, τ_i consists of i playing π_i with input y_i, where $\vec{y}_T = L(\vec{x}_T)$, except that if i was supposed to output (S, z) (note that all outputs of honest agents are of this form, since the ideal output has this form, and π^f securely computes f) it outputs $O_i(\vec{x}_T, z)$ instead. Finally, consider a scheduler σ_e that delays all messages to and from the agents in $\overline{C}$ until an honest agent terminates. By the properties of Section 2.2.1, it is easy to see that $\vec{\rho}(\vec{x}, A)$ and $\vec{\pi}(\vec{x}, T, \vec{\tau}_T, \sigma_e)$

are identically distributed. If A is randomized, τ_T works the same way except that it chooses C, $\vec{y}_T$, and $\vec{O}_T$ by sampling from the same distribution that r is sampled from. □

We next show how secure computation relates to simulating a mediator. Consider the following mediator protocol $\vec{\pi}_d^f$: Agents send their inputs to the mediator the first time they are scheduled. The mediator waits until it has received a valid input x_i from all i in a subset C of agents with $|C| \geq n - t$. The mediator then computes $y = f_C(\vec{x})$ and sends each agent the pair (C, y). When the agents receive a message from the mediator, they output that message and terminate.

Clearly π_d^f satisfies SC1. It is easy to see that it also satisfies SC2: Given a set T of malicious agents, a deterministic strategy $\vec{\tau}_T$ for the malicious agents, and a deterministic scheduler σ_e, define $L(\vec{x}, r)$ to be whatever the malicious agents send to the mediator with input $\vec{x}$ let $c(\vec{x})$ be the set of agents from whom the mediator has received a message the first time it is scheduled after having received a message from a least $n - t$ agents (given σ_e, $\vec{\tau}_T$, and input $\vec{x}$), and let $O(\vec{x})$ be the output function that malicious agents use in $\vec{\pi}$ (note that they receive a single message with the output of the computation, so their output depends only on $\vec{x}$, $\vec{\tau}$, and σ_e). Arbitrary functions $\vec{\tau}_T$ and σ_e, can be viewed as resulting from sampling random bits r according to some distribution and then running deterministically; the protocols c, h, and O can sample r from the same distribution and then proceed as above with respect to the deterministic $\vec{\tau}(r)$ and $\sigma_e(r)$.

The mediator protocol π_d^f satisfies SC3 as well. Given $A = (T, c, L, O)$, the definition of $\vec{\tau}_T$ and σ_e is straightforward: the agents in T choose a random input $r \in \mathcal{R}$ and then each agent $i \in T$ sends $L(x_i, r)$ to the mediator. The scheduler σ_e delivers all messages from the agents in $c(\vec{x}_T, r)$ first, and then schedules the

mediator. It then delivers all the other messages.

Since both $\vec{\pi}_d^f$ and $\vec{\pi}$ satisfy SC2 and SC3, for each adversary A in $\vec{\pi}^f$ there exists an adversary A' in $\vec{\pi}_d^f$ whose outputs are identically distributed.

Unfortunately, given a protocol $\vec{\pi}_d$ for the mediator, there might not exist a function f such that SC2 and SC3 hold, as the following example shows: Suppose that agents have no input, and just send a fixed arbitrary message to the mediator the first time they are scheduled. If the first message received by the mediator comes from i, the mediator sends the message i to all agents. When an agent receives a message i from the mediator, it outputs the message and terminates. Note that the output of the agents is not a function of their input profile, and thus SC2 and SC3 won't hold for any function f. Nevertheless, we will still be interested in implementing a cheap talk game whose output is distributed identically to that of this mediator game.

We are thus interested in getting analogues to SC2 and SC3 for arbitrary interactive protocols. In particular, we want these definitions not to involve secure computation adversaries. This motivates the following definition:

Definition 3. *Let $O(\vec{\pi}_{-T} + \pi_d, \tau_T, \sigma_e, \vec{x})$ be the output of protocol $\vec{\pi} + \pi_d$ (where $\vec{\pi}$ is the agents' protocol and π_d is the mediator's protocol), given scheduler σ_e and input $\vec{x}$, where the agents in T deviate from $\vec{\pi}$ by using strategy $\vec{\tau}_T$. Protocol $\vec{\pi}'$ in a cheap-talk game* t-bisimulates *$\vec{\pi} + \pi_d$ if the following hold for every subset T of malicious agents with $|T| \leq t$ and every input $\vec{x}$:*

(a) For every strategy $\vec{\tau}_T$ for the agents in T and every scheduler σ_e in the mediator game, there exists a strategy $\vec{\tau}'_T$ and a scheduler σ'_e in the cheap-talk game such that $O(\vec{\pi}_{-T} + \pi_d, \vec{\tau}_T, \sigma_e, \vec{x})$ and $O(\vec{\pi}'_{-T}, \vec{\tau}'_T, \sigma'_e, \vec{x})$ are identically

distributed.

(b) For every strategy $\vec{\tau}'_T$ for the agents in T and every scheduler σ'_e in the cheap-talk game, there exists a strategy $\vec{\tau}_T$ and a scheduler σ_e in the mediator game such that $O(\vec{\pi}_{-T} + \pi_d, \vec{\tau}_T, \sigma_e, \vec{x})$ and $O(\vec{\pi}'_{-T}, \vec{\tau}'_T, \sigma'_e, \vec{x})$ are identically distributed.

We can define bisimulation in the synchronous case analogously (without taking into account the scheduler).

In the rest of the paper, we omit the "$+\pi_d$" term whenever it is clear from its context. Note that the first clause is analogous to SC2, while the second clause is analogous to SC3. Also, note that there is no clause analogous to SC1 since we are interested in agents always terminating, if we consider non-termination as a special type of output, SC2 already guarantees that non-termination happens with the same probability in $\vec{\pi}'$ and $\vec{\pi} + \pi_d$. (In the setting of BGW, since all functions terminate, SC2 implies SC1, a point already made by Canetti [16]).

Our earlier discussion proves the following proposition:

Proposition 1. *$\vec{\pi}^f$ t-bisimulates $\vec{\pi}^f_d$ if $t < n/4$.*

3.2 Simulating arbitrary protocols

Although BCG make claims for their protocol only if $n > 4t$, variants of some of the properties that they are interested in continue to hold even if $n < 4t$. The first of these properties is that if $n > 3t$, then the only way that the adversary can affect $\vec{\pi}^f$ is by preventing some honest agents from terminating. We can capture this notion as follows.

Definition 4. *A scheduler is* relaxed *if it can decide not to deliver some of the messages. A protocol* $\vec{\pi}'$ (t,t')-bisimulates $\vec{\pi}$ *if it* t*-bisimulates* $\vec{\pi}$ *but the schedulers* σ'_e *and* σ_e *of the first and second clause of Definition 3 respectively may be relaxed for* $t \geq |T| > t'$.

Proposition 2. $\vec{\pi}^f$ (t,t')*-bisimulates* $\vec{\tau}^f + \tau_d$ *if* $3t + t' < n$ *and* $t \geq t'$.

This means that if $3t+t' < n$, then adversaries of size between t' and t have the same power to affect the outcome with $\vec{\pi}^f$ as with $\vec{\tau}^f + \tau_d$ as long as schedulers are allowed to discard messages, so that they never reach their recipient. In particular, this means that the adversary cannot influence the outcome in any other way than by preventing some honest players from terminating. However, we can show that the BCG protocol has the property that if at least $2t+1$ honest agents terminate, then all the remaining honest agents terminate. This observation motivates the following definition:

Definition 5. *A protocol* $\vec{\pi}$ (t,k)-coterminates *if, for all adversaries* $A = (T, \vec{\tau}_T, \sigma_e)$ *with* $|T| \leq t$ *and all input profiles* $\vec{x}$*, in all histories of* $\vec{\pi}$ *with adversary* A *and input* $\vec{x}$*, either all the agents not in* T *terminate or strictly fewer than* k *agents not in* T *do.*

Proposition 3. $\vec{\pi}^f$ $(t, 2t+1)$*-coterminates.*

We do not prove Proposition 2 or 3 here, since we prove a generalization of them below (see Theorem 3).

The goal of this paper is to show that we can securely implement any interaction with a mediator, and do so in a way that ensure the two properties discussed in Section 3.1. This is summarized in the following theorem:

Theorem 3. *For every protocol $\vec{\pi} + \pi_d$ for n agents and a mediator, there exists a protocol $\vec{\pi}'$ for n agents such that $\vec{\pi}'$*

(a) (t, t')-bisimulates $\vec{\pi}$ if $n > 3t + t'$ and $t \geq t'$,

(b) $(t, 2t + 1)$-coterminates if $n > 3t$ and $\vec{\pi} + \pi_d$ is in canonical form.

Moreover, if π_d is responsive, the expected number of messages sent in a history of $\vec{\pi}'$ is polynomial in n and N, and linear in c, where N is the expected number of messages sent when running $\vec{\pi} + \pi_d$ and c is the number of gates in an arithmetic circuit that implements the mediator's protocol.

The construction of $\vec{\pi}'$ is given in Section 3.3.2 and, not surprisingly, uses many of the techniques used by BCG. And, like BKR, if we allow an ϵ probability of error we get stronger results. We define ϵ-t-bisimulation just like t-bisimulation (Definition 3), except that, in both clauses, the distance between $(\vec{\pi} + \pi_d)(\vec{x}, A)$ and $\vec{\pi}'(\vec{x}, A')$ is less than ϵ, where the distance d between probability measures ν and ν' on some finite space S is defined as $d(\nu, \nu') = \sum_{s \in S} |\nu(s) - \nu'(s)|$. The definition of ϵ-t-bisimulation and ϵ-(t, t')-bisimulation are analogous. A protocol ϵ-(t, k)-*coterminates* if it (t, k)-coterminates with probability $1 - \epsilon$.

Theorem 4. *For every protocol $\vec{\pi} + \pi_d$ for n agents and a mediator and all $\epsilon > 0$, there exists a protocol $\vec{\pi}'$ for n agents such that $\vec{\pi}'$*

(a) ϵ-(t, t')-bisimulates $\vec{\pi} + \pi_d$ if $n > 2t + t'$ and $t \geq t'$,

(b) ϵ-$(t, t + 1)$-coterminates if $n > 2t$ and $\vec{\pi} + \pi_d$ is in canonical form.

Moreover, if π_d is responsive, $\vec{\pi}'$ can be implemented in such a way that the expected number of messages when running $\vec{\pi}' + \pi_d$ is polynomial in n and N, and linear in c, where N is the expected number of messages sent when running $\vec{\pi} + \pi_d$.

In the synchronous case we get the following result:

Theorem 5. *For every protocol $\vec{\pi} + \pi_d$ for n agents and a mediator, there exists a protocol $\vec{\pi}'$ for n agents such that $\vec{\pi}'$ t-bisimulates $\vec{\pi}$ if $n > 3t$. Moreover, the expected number of messages sent in a history of $\vec{\pi}'$ is polynomial in n and N, and linear in c, where N is the expected number of messages sent when running $\vec{\pi} + \pi_d$ and c is the number of gates in an arithmetic circuit that implements the mediator's protocol.*

3.3 Proofs of Theorems 3 and 4

3.3.1 t-uniform VSS and CC and determinate VSS

BCG's implementation of VSS satisfies some additional properties that they do not make use of, but that we will need in our construction, so we outline them here.

Given a sequence $I = \{i_1, \ldots, i_n\}$ of distinct honest agents, a sequence $S = \{s_1, \ldots, s_n\}$ of values, and a secret s, we say that (I, S) is *s-realizable* by a VSS (resp., CC) implementation if, for that implementation, there exists an agent i such that the event that each agent i_k computes s_k as the output of i's invocation of VSS with secret s has nonzero probability. In other words, (I, S) is s-realizable if S could be the output of the agents in I running VSS with secret s. (I, S) is realizable if it is s-realizable for some s.

We say that (I', S') is an *s-extension* of (I, S) if I is a prefix of I', S is a prefix of S' and (I', S') is s-realizable. (I', S') is a *full* s-extension if I' is the set of all

agents. Again, we say that (I', S') is an extension of (I, S) if it is an s-extension of (I, S) for some s; it is a *full extension* if, in addition, I' is the set of all agents. We omit the I term in each of these definitions if it is clear from context which agent computed each of the shares in S.

BCG's implementation of VSS and CC guarantees that if s_i is the share of an honest agent i after running an invocation of VSS or CC, then there exists a polynomial p of degree t (where t is a bound on the number of malicious agents) such that $p(i) = s_i$ and $p(0)$ is the secret shared through VSS or computed through CC. Moreover, this polynomial p is uniformly sampled from the set of all polynomials p' of degree t with $p'(0) = s$. With BCG's implementation of VSS and CC, a pair (I, S) is realizable iff there exists a polynomial p of degree t such that $p(i_k) = s_k$ for all k.

With this notation, we can state the properties that we need for our VSS and CC implementation. A circuit that computes values in $\mathbb{F}_p$ can be securely computed by n agents if the inputs are shared using VSS, and the addition and multiplication gates are computed using CC. At the end of the computation, each agent has a share of the output. The first property that we require, to simplify our proof, is that for all sets T of size at most t, the output of such a circuit is uniformly distributed over $\mathbf{F}_p^{|T|}$:

Definition 6. *An implementation of VSS and CC is* t-uniform *if, for all circuits C with a single output gate, and all sets I of honest agents with $|I| \leq t$, the output of I after securely computing C is uniformly distributed over $\mathbf{F}_p^{|T|}$.*

We actually seem to need a somewhat stronger property than t-uniformity: *conditional t-uniformity* (i.e., t-uniformity conditional on the outcome of earlier CC instances). In general, a t-uniform implementation may not satisfy conditional

t-uniformity. For example, two empty circuits that take the shares of a single VSS instance as inputs produce identical outputs, which are the the shares of the VSS. Fortunately, it is easy to convert a circuit C to a circuit C' that computes the same secret, and also satisfies conditional t-uniformity conditional on all other CC instances. Suppose for simplicity that C has a single output gate. We construct C' by having each agent i invoke a t-uniform VSS with 0 as the secret. Agent i then computes (using CC) the product of the secrets whose shares it receives, and adds it share of the product to the output of C. Clearly the agents will get the same value with C and C' after they share their shares, no matter what the malicious agents do. Also, since C' takes as inputs instances of VSS that are not used in any other circuit, the output of a subset I with $|I| \leq t$ conditional on the output of all other CC instances is uniformly distributed over $\mathbb{F}_p^{|T|}$. Thus, if the implementation of VSS and CC is t-uniform, we can assume without loss of generality that we are working with conditionally t-uniform circuits.

We also require that the shares of a set of at least $t+1$ honest agents uniquely determine the secret.

Definition 7. *An implementation P of VSS and CC is* t-determinate *if*

(a) P is t-resilient;

(b) P is t-uniform if there are at most t malicious agents; and

(c) for all pairs (I, S) with $|I| \geq t+1$, if (I, S) is realizable, then there exists a unique full extension (I', S') of (I, S).

BCG's implementation of VSS is t-determinate: it is easy to check that it satisfies clauses (b) and (c) of the definition; BCG prove that it is t-resilient.

For all of our constructions, we assume that the secret-sharing scheme used is t-determinate.

3.3.2 Constructing π'

Our construction of $\vec{\pi}'$ is similar in spirit to BCG's constrution of $\vec{\pi}_f$. As we said earlier, what makes our setting more complicated is that the agents send multiple messages to the mediator, and the mediator sends multiple messages back. We will need to keep track of which messages are being sent in response to which other messages. Moreover, to get t-bisimulation, we need to be able to simulate all possible behaviors of the scheduler, both with $\vec{\pi} + \pi_d$ and with $\vec{\pi}'$.

For ease of exposition, we begin by giving a naive construction of $\vec{\pi}'$, which, as we later show, does not quite satisfy all the desired properties. However, it gives the intuition for the actual construction (which requires only a small modification of the naive construction). We now sketch the naive construction, then give a detailed description, and then explain the minor modifications needed to correct the problems in the naive construction. This construction may not satisfy the bound we claimed on the expected number of messages when the mediator is responsive. We show in Section 3.3.4 how to modify the construction so as to satisfy that bound.

When running π'_i, each agent i simulates its counterpart running π_i except that, rather than sending and receiving messages from the mediator, i shares messages it shares and reconstructs messages using VSS. In addition, all agents use CC to compute the mediator's local history given the messages shared by the agents and to compute the messages the mediator sends to the agents according to π_d,

given its local history. Note that, after running CC, each agent has a share of the mediator's message. If this is a message sent by the mediator to agent i, then each agent sends its share to i, so that i can reconstruct the message.

To do the simulation, each agent i computes two sequences, $\{h_{i,k}\}_{k\in\mathbb{N}}$ and $\{h^i_{d,k}\}_{k\in\mathbb{N}}$. Each element $h_{i,k}$ in the first sequence represents i's local history the kth time that i is scheduled in the simulated interaction with the mediator, while each term $h^i_{d,k}$ of the second sequence represents i's share of the mediator's local history the kth time that the mediator is scheduled in the simulated interaction. Of course, these histories depend (in part) on how i does the simulation. In our naive protocol, we assume that all agents get scheduled in the simulation at times corresponding to when they get scheduled in the computation of $\vec{\pi}'$, after getting corresponding messages. That is, whenever i is scheduled in $\vec{\pi}'$, it checks all the messages received from the simulated mediator since the last time it was scheduled in $\vec{\pi}'$, then simulates itself being scheduled in $\vec{\pi} + \pi_d$ after receiving exactly the same messages in the same order as it did in $\vec{\pi}'$. This means that $h_{i,k}$ is constructed by appending to $h_{i,k-1}$ all messages received by i and the results of all local computations of i between the $(k-1)$st and kth time that i is scheduled in $\vec{\pi}'$. Therefore, in the naive construction, $h_{i,k}$, which is i's view the kth time that i is scheduled in the simulation of the computation of $\vec{\pi} + \pi_d$, is also part of i's view of the simulation the kth time that i is scheduled in $\vec{\pi}'$. That is, if i has been scheduled k times in $\vec{\pi}'$, then it is also scheduled exactly k times in the simulation. As we show later by example, this property prevents us from being able to simulate all schedulers in the interaction with the mediator, and is precisely why the naive construction does not quite work. That said, for now we continue to explain the naive construction.

Note that, in $\vec{\pi}'$, i does not receive the mediator's actual messages in its simulation; rather, it receives shares of those messages. Agent i appends a message to $h_{i,k-1}$ only at the point that the message can be reconstructed from the shares of the message that i receives from the other agents. After computing $h_{i,k}$, i computes which messages it sends according to π_i (given the history it has simulated) and, for each such message μ, i shares μ using VSS.

Computing $\{h^i_{d,k}\}_{k\in\mathbb{N}}$ is more subtle. We must ensure that all agents agree on what messages should be appended to $h_{d,k}$ to get $h_{d,k+1}$; otherwise, agents will not have a consistent view of the mediator's history. Since, at any point in the execution of $\vec{\pi}'$, different agents may have terminated different invocations of VSS, this requires a little care. Let $h^i_{d,0}, h^i_{d,1}, h^i_{d,2}, \ldots, h^i_{d,k}$ be the sequence of shares of the mediator's local history in the simulation computed thus far by agent i. We will ensure that, for all k, $(h^1_{d,k}, h^2_{d,k}, \ldots, h^n_{d,k})$ are shares of some local history $h_{d,k}$ of the mediator in the computation of π being simulated, where $h_{d,0}$ is the empty sequence $\langle\,\rangle$ and $h_{d,k}$ is a prefix of $h_{d,k+1}$ (so that the mediator's history get increasingly longer). After computing their shares of $h_{d,k}$, agents can perform a circuit computation to compute the messages the mediator sends to the agents given local history $h_{d,k}$.

We now describe the naive construction of $\vec{\pi}'$ in more detail. As we said, because our naive construction assumes that i is scheduled the same number of times in the simulation of π_i as in the actual computation of π'_i, the kth time i is scheduled when running π'_i, i's history includes simulated histories $h_{i,0}, \ldots, h_{i,k-1}$ and shares of simulated histories $h^i_{d,0}, \ldots, h^i_{d,k'}$ (note that k' might not be equal to k). These simulated histories are the output of local computations, and thus are recorded in the i's history. In addition, i's history keeps track of the status of all

the invocations of protocols like VSS and CC in which i participates (including results of random coin tosses, which we also view as the outcome of computation). Note that there might be several invocations of the same protocol that an agent is involved in at the same time; for example, an agent might invoke VSS several times before any of them complete. To remove ambiguity, we assume that all invocations of a protocol are labeled; for example, the first invocation of VSS invoked by agent i could be labeled (VSS, i, 1), the second one could be labeled (VSS, i, 2)), and so on. Each agent communicates these labels to the scheduler using the scheme presented in Section 1.3. Thus, we can assume without loss of generality that the scheduler knows the labels.

If agent i is scheduled when it is in such a state, it first processes all messages received since the last time it was scheduled. (We assume that all messages received since the last time that i was scheduled are held in some buffer.) "Processing a message" μ consists of i playing its part in the protocol to which μ belongs (which we assume is indicated in the label of μ); if μ is a share of a simulated mediator message, i checks if it can reconstruct a new mediator message and, if so, updates $h_{i,k}$ accordingly. After processing all of its new messages, i will have constructed $h_{i,k+1}$. Agent i checks what action(s) π_i takes given input $h_{i,k+1}$. If π_i outputs a value v, then so does π'_i; if these actions include sending one or more messages to the mediator, then i shares those messages using VSS instead. Finally, if possible, i computes its share of the simulated mediator's local history $h^i_{d,k'+1}$ and computes (along with the other agents) which messages the mediator sends to the agents. We now explain how this is done.

Agent i computes $h^i_{d,k}$ inductively. Clearly, $h_{d,0}$, the mediator's initially local history, is empty (and all agents know this). To compute $h^i_{d,0}$, i simulates a compu-

tation of VSS initiated by agent 1 (there is nothing special about agent 1 here; any other agent would do) with input the empty sequence under the assumption that all agents are honest, and takes $h^i_{d,0}$ to be i's share of the output of the computation. Since VSS is a randomized protocol, to assure consistency, all agents must use the same random bits in this computation of VSS; we can assume that these random bits are hardcoded into $\vec{\pi}'$. Note that this is equivalent to just hardcoding the values of $h^i_{d,0}$ in $\vec{\pi}'$, but viewing $h^i_{d,0}$ as the output of an invocation of VSS will be useful in the future.

Assuming that $h^i_{d,k}$ has been computed, we show how to compute $h^i_{d,k+1}$. The idea is that the agents perform a circuit computation with inputs $h^i_{d,k}$ and all the new messages to be appended to $h_{d,k}$ (note that each agent has a share of each of these inputs). It is critical when running a CC invocation that the inputs of each honest agent are consistent with the inputs of all other honest agents participating in the same invocation. More precisely, for all pairs of agents i and j, if i's ℓth input is i's share of the message being shared in some VSS invocation (VSS, i', j'), then j's ℓth input must be j's share of the same message. It is not straightforward to ensure this, since i and j might have completed different invocations of VSS at the time that they update $h^i_{d,k}$ and $h^j_{d,k}$ respectively.

Since this issue arises in a number of contexts, we formalize this notion of consistency. Suppose that $\vec{\rho}$ is a joint protocol and $\vec{h}$ is a history of $\vec{\rho}$. Let v be an invocation of CC in $\vec{h}$ in which some honest agent i has participated. Invocation v is *well-defined* if the following holds:

(a) All honest agents eventually participate in v.

(b) Suppose that α has m inputs. For each ℓ with $1 \leq \ell \leq m$, there exists an invocation v_l of VSS or CC that occurred earlier in the computation such

that each honest agent i's share of the ℓth input of v is i's share of the output of v_ℓ.

All the invocations of CC to compute $h^i_{d,k+1}$ are well-defined in this sense, since agents must use the shares of the same secret at each gate of the CC. To ensure this, agents first agree on which subset of messages should be appended to $h_{d,k}$, then they agree on the order in which these messages should be appended, and finally they append these messages to $h_{d,k}$ and compute the messages sent by the scheduler to the agents, which are also appended to $h_{d,k}$. The protocol for extending $h^i_{d,k}$ to $h^i_{d,k+1}$ proceeds in four phases, denoted $k1, \ldots, k4$.

Phase $k1$: Let N the maximum number of messages that an honest agent sends when running $\vec{\pi}+\pi_d$. Each agent i participates in nN consensus protocols in phase $k1$, denoted $p_{1,1,k}, \ldots, p_{n,N,k}$, where $p_{i,j,k}$ is intended to achieve consensus on whether i has shared its jth message successfully. More precisely, i's input to consensus protocol $p_{j,\ell,k}$ is 1 iff i has terminated j's ℓth invocation of VSS by the time i starts phase $k1$ and $p_{j,\ell,k'}$ has output 0 for all $k' < k$. Agent i waits until it has terminated all the $p_{j,\ell,k}$ consensus protocols it is involved with in phase $k1$ before starting phase $k2$. If the output of some consensus protocol $p_{j,\ell,k}$ is 1, then i waits until it has also completed j's ℓth VSS invocation in round k before starting phase $k2$.

Phase $k2$: Let $p_{j_1,\ell_1,k}, p_{j_2,\ell_2,k}, \ldots, p_{j_{m_k},\mathit{withoutput}\ell_{m_k},k}$ be the consensus protocols that were used in phase $k1$ and had output 1, ordered in lexicographic order (i.e., $p_{j,\ell,k}$ precedes $p_{j',\ell',k'}$ iff $j < j'$ or $[j = j'$ and $\ell < \ell']$). In this phase, i coordinates with the other agents on the order that they should append the messages shared in (VSS, j_1, ℓ_1), ..., (VSS, j_{m_k}, ℓ_{m_k}) to $h_{d,k}$. We want the agents to agree on the same permutation of $(j_1, \ell_1), \ldots, (j_{m_k}, \ell_{m_k})$. To do this, they

use BCG's secure computation protocol. Each agent i inputs a permutation to the protocol; the output is the permutation that they coordinate on. Agent i's input to the protocol is the unique permutation $\sigma_i : [m_k] \longrightarrow [m_k]$ satisfying $\sigma_i(a) < \sigma_i(b)$ iff i terminated (VSS, j_a, ℓ_a) before (VSS, j_b, ℓ_b). Note that since i completes all the VSS invocations that are in progress in phase $k1$ before it starts phase $k2$, i can compute this permutation. The BCG computation returns the (unique) permutation θ satisfying $\theta(a) < \theta(b)$ iff $\sum_{i \leq n} \sigma_i(a) < \sum_{i \leq n} \sigma_i(b)$ or $\sum_{i \leq n} \sigma_i(a) = \sum_{i \leq n} \sigma_i(b)$ and $a < b$. Thus, roughly speaking, $\theta(a) < \theta(b)$ if, on average, agents terminated (VSS, j_a, ℓ_a) before (VSS, j_b, ℓ_b). Note that because of the asynchrony of the system, we can guarantee only that at most $n-t$ inputs will be available when computing θ. For the remaining inputs σ_i we take σ_i to be the identity permutation.

Phase $k3$: Agent i uses CC to append new messages to $h^i_{d,k}$. More precisely, it updates $h^i_{d,k}$ with (VSS, $j_{\theta(1)}, \ell_{\theta(1)}$), ..., (VSS, $j_{\theta(m_k)}, \ell_{\theta(m_k)}$) in the order determined by the permutation θ computed in phase $k2$. Note that the properties of VSS and the fact that i completes all outstanding VSS invocations in phase $k1$ guarantee that all honest agents have a share of all these messages when they start phase $k3$. This procedure gives agent i a share that we denote $\tilde{h}^i_{d,k+1}$ of the mediator's updated local history $\tilde{h}_{d,k+1}$ after appending these new messages to $h_{d,k}$ in the appropriate order.

Phase $k4$: Agent i computes $h^i_{d,k+1}$ by using CC to append to $\tilde{h}_{d,k+1}$ the message the mediator sends to the according to π_d, given input $\tilde{h}_{d,k+1}$. (This can be done because all agents know the mediator's protocol π_d.) Agent i's input for this invocation of CC is its share $\tilde{h}^i_{d,k+1}$. Note that each agent invokes CC only once, using it to compute all the mediator's messages are computed and appended them to $\tilde{h}_{d,k+1}$. Agent i's output of this invocation of CC is its

share of $h^i_{d,k+1}$.

In phase $k4$ agents never actually compute (their shares, if any, of) the messages sent by the mediator during its $(k+1)$st turn; they compute only the result of appending these message to $h_{d,k}$. Later we will see how agents compute their shares of each of thesee messages individually, using the fact that it is encoded in $h_{d,k+1}$.

This protocol satisfies two important properties if $n > 4t$:

Lemma 2. *All honest agents eventually terminate phase $k1$. Moreover, for all adversaries of size at most t and all histories, the CC protocol invoked in phase $k3$ is well-defined.*

Proof. If the output of some consensus protocol $p_{j,\ell,k}$ is 1, then the properties of VSS guarantee that at least one honest agent had input 1. Thus, at least one honest agent terminated (VSS, j, ℓ). The properties of VSS guarantee that all other honest agents eventually terminate this VSS invocation as well. The properties of consensus and secure computation guarantee that all agents use the outputs of the same VSS invocations in the same order, which means that the CC procedure of phase $k3$ is well-defined for all runs. □

Lemma 3. *If an honest agent shares a message μ using VSS, then μ will be in $h_{d,k}$ for some k, and hence each honest agent i will have a share of μ in $h^i_{d,k}$.*

Proof. If (VSS, j, ℓ) is invoked by an honest agent j, then all honest agents are guaranteed to eventually terminate this invocation of VSS. Thus, the output of consensus protocol $p_{j,\ell,k}$ (at Phase $k1$) is 1 for exactly one value of k. (Note that if the output of $p_{k,\ell,k}$ is 1, then all honest agents take 0 to be the input for all consensus protocols $p_{j,\ell,k'}$ with $k' > k$. This guarantees that the output of all these

protocols is 0.) This ensures that (VSS, j, ℓ) is appended to $\tilde{h}_{d,k+1}$ in Phase $k3$, and thus it is included in $h_{d,k+1}$. □

Since, by the time each agent i finishes computing $h^i_{d,k}$, all the messages that the mediator sends to each agent are already encoded in $h_{d,k}$, it may seem that to compute the shares of these messages individually, i would have to use an instance of CC for each one.

However, this procedure is not so straightforward since i does not know beforehand how many messages the mediator sends or the order in which the mediator sends messages the kth time it is scheduled (although this is also encoded in $h_{d,k}$). To deal with this issue, before computing its share of each of the mediator's messages, i first checks if there is a message that still needs to be sent and, if so, who the recipient is.

More precisely, let $f_{k,\ell}$ be the function that takes as input a mediator's local history and returns the recipient of the ℓth message sent by the mediator the kth time it is scheduled; similarly, let $g_{k,\ell}$ be the function that computes the ℓth message sent by the mediator the kth time it is scheduled, given the mediator's history. If the mediator sends fewer than ℓ messages the kth time it is scheduled, or if the input is not a well-defined local history, both $f_{k,\ell}$ and $g_{k,\ell}$ return 0. After computing $h_{d,k}$, agent i proceeds as follows for $\ell = 1, 2, \ldots$: it performs a circuit computation of $f_{k,\ell}$ with input $h^i_{d,k}$. Then i broadcasts the output of this computation, and uses the values it receives from other agents to reconstruct $f_{k,\ell}(h_{d,k})$. If $f_{k,\ell}(h_{d,k}) \neq 0$, i performs a circuit computation of $g_{k,\ell}$ with input $h^i_{d,k}$ and computes $f_{k,\ell+1}(h_{d,k})$. If $f_{k,\ell}(h_{d,k}) = 0$, then for each $\ell' < \ell$, i sends the output of its circuit computation of $g_{k,\ell'}$ to agent $f_{k,\ell'}(h_{d,k})$.

This completes the description of the naive version of $\vec{\pi}'$. As we have been hinting, this protocol does not quite work. The following example makes the reasons more precise.

Consider a protocol $\vec{\pi} + \pi_d$ in which the mediator sends a STOP message to each agent the first time it (the mediator) is scheduled. If i was scheduled before receiving the STOP message, it outputs 0; otherwise, it outputs 1. Note that any combination of outputs is possible with $\vec{\pi} + \pi_d$, depending on when the scheduler schedules the mediator and the agents. However, this is not true for $\vec{\pi}'$ as we have defined it. Suppose, for example, that all agents are honest, and i is the first agent scheduled in a history of $\vec{\pi}'$. At this point, i is supposed to compute $h_{i,1}$. Since it has not received any messages, it will take $h_{i,1}$ to be empty, and thus output 0. It follows that no history of $\vec{\pi}'$ can end with all agents outputting 1, which means that $\vec{\pi}'$ does not t-bisimulate $\vec{\pi} + \pi_d$.

In our construction of the naive version of $\vec{\pi}'$, each agent i calculates $h_{k,i}$ the kth time that i is scheduled in $\vec{\pi}'$. However, since computing each of $h_{d,1}, h_{d,2}, \ldots$ takes several turns of i, the mediator's history $h_{d,k'}$ being computed by i during its kth turn satisfies that $k' \leq k$. This means that i is simulating that the mediator, at all times, has taken less turns than i, which may not be true in the protocol with the mediator. As our example shows, some scenarios cannot be simulated with our naive construction because of this. We deal with this problem by using the scheduler in the simulation to determine whether an update to $h_{i,k}$ or $h^i_{d,k'}$ should occur when i is scheduled.

We proceed as follows. When an agent i is first scheduled, i sends two special messages, $\mathit{proceed}_i$ and $\mathit{proceed}_{d,0}$, to itself and computes $h_{i,0}$ (which is just an empty history) and $h^i_{d,0}$. What i does when it is scheduled for the ℓth time for $\ell > 1$

depends on whether it has received messages of the form $proceed_i$ and $proceed_{d,r}$ and messages from itself since the last time it was scheduled. Suppose that i has computed the sequences $h_{i,0}, \ldots, h_{i,k}$ and $h^i_{d,0}, \ldots, h^i_{d,k'}$ when it is scheduled for the ℓth time. If i has not received a $proceed_i$ message since the last time it was scheduled, i does not compute $h_{i,k+1}$. If i has received a $proceed_i$ message since the last time it was scheduled, then it sends itself another $proceed_i$ message and computes $h_{i,k+1}$ as described above, using all the messages it received since it was last scheduled and received a $proceed_i$ message. Thus, i computes the next history in the sequence $\{h_{i,k}\}_{k\in\mathbb{N}}$ if and only if i receives a $proceed_i$ message. Similarly, if i has not received a message of the form $proceed_{d,r}$ since the last time it was scheduled, then it does not do any of the steps needed to compute $h^i_{d,k'+1}$. If it has received a message of the form $proceed_{d,r}$ message since the last time it was scheduled, it sends itself a message of the form $proceed_{d,r+1}$. (Thus, the second component of the subscript serves a counter for the number of such messages that have been sent.) If $k' \leq r$, then i plays its part in computing $h^i_{d,k'+1}$. Otherwise, i does not take part in any procedure involved in the computation of $h^i_{d,k'+1}$; that is, i waits until it receives $proceed_{d,r}$ before attempting to compute $h^i_{d,r}$. Thus, when it is scheduled, i may take part in computing both $h_{i,k+1}$ and $h^i_{d,k'+1}$, only one of them, or neither of them. Since the scheduler must eventually deliver all messages, all agents receive all the *proceed* messages that they send themselves, so eventually do update $h_{i,k}$ and $h^i_{d,k}$.

This completes the construction of $\vec{\pi}'$. In the next few subsections, we prove that $\vec{\pi}'$ has the desired properties.

3.3.3 The proof of Theorem 3(a)

We now prove Theorem 3(a). For ease of exposition, we begin by proving this result for the special case that $t' = t$, showing that $\vec{\pi}'$ t-bisimulates $\vec{\pi}$ if $n > 4t$.

Proof that $\vec{\pi}'$ t-bisimulates $\vec{\pi}$ if $n > 4t$: We actually prove a result slightly stronger than Theorem 3(a): while the definition of bisimulation allows σ'_e to depend on both σ_e $\vec{\tau}'_T$ and $\vec{\tau}'_T$ to depend on both $\vec{\tau}_T$ on σ_e, in our construction below, σ'_e depends only on σ_e (and not on $\vec{\tau}'_T$), while $\vec{\tau}'_T$ depends only on $\vec{\tau}_T$ (and not on σ_e).

We begin by showing that $\vec{\pi}'$ satisfies part (a) of the definition of bisimulation assuming that all agents are honest. Later, we show how this proof can also be applied to the case in which a subset T of agents deviate. Given a scheduler σ_e in the mediator setting, we construct a scheduler σ'_e in the setting without the mediator as follows. Initially, σ'_e schedules each agent i exactly once. Recall that if i is honest, the first time it is scheduled it sends only $proceed_i$ and $proceed_{d,0}$ messages to itself. The point of scheduling all the agents initially is simply to get these *proceed* messages into the system. From then, just as the agents do with $\vec{\pi}'$, σ'_e simulates which history h_e the scheduler would have in the interaction with the mediator if the mediator and the agents used $\vec{\pi} + \pi_d$ and the scheduler used σ_e. At the beginning of the game, the scheduler sets h_e to the empty history. How the scheduler updates h_e and what actions the scheduler performs according to σ'_e then depend on the form of $a = \sigma_e(h_e)$ (i.e., the actions that σ_e would perform given history h_e), and on the actions that the agents perform afterwards:

- If a has the form $sch(i)$, then it delivers i's most recent $proceed_i$ message if there is one to deliver, and then schedules i. Suppose that i initiates ℓ VSS

instances during its turn. Then immediately after i's turn, σ'_e appends $sch(i)$ and ℓ $snd(d, i)$ events to h_e, followed by a $done(i)$ event.

- If a has the form $sch(d)$ and it is the kth time that the mediator is scheduled according to σ_e, then the scheduler delivers to each agent i if there is one to deliver and then schedules agents cyclically $(1, 2, \ldots, n, 1, 2, \ldots)$ until all agents i finish computing $h^i_{d,k}$ and their share of each of the messages sent by the mediator during its kth turn. The scheduler also delivers to each agent i all the messages required by i for the computation of $h^i_{d,k}$ and the shares of the mediator's messages immediately after they are Suppose that the agents determine that the mediator sends messages to $j_1, j_2, \ldots, j_\ell$, in that order. Then σ'_e appends $sch(d), snd(j_1, d), \ldots, snd(j_\ell, d), done(d)$ to h_e. Note that the scheduler knows $j_1, \ldots, j_\ell$, and also knows when each agent i terminates the computation of $h^i_{d,k}$ (given our assumption that the scheduler knows the label of each message), since agent i has terminated the computation of $h^i_{d,k}$ if all messages related to this computation have been delivered and no agent sent further messages when it was scheduled. Thus, all agents i are guaranteed to have terminated the computation of $h^i_{d,k}$ after the scheduler has gone through a full cycle of scheduling the agents without any agent sending any message required for the computation of $h^i_{d,k}$, for $i = 1, \ldots, n$. (Recall that we are assuming for now that all agents are honest.)
- If a has the form $rec(i, d, \ell)$, the scheduler delivers to i all the messages that i needs to compute the mediator's ℓth message to i. That is, the scheduler delivers the messages from other agents containing the shares of the ℓth message from the mediator to i. (By our inductive hypothesis, these messages have been sent but not yet delivered.) Then σ'_e appends $rec(i, d, \ell)$ to σ_e.
- If a has the form $rec(d, i, \ell)$, the scheduler schedules the agents cyclically un-

til all the agents finish computing (VSS, i, ℓ). More precisely, the scheduler delivers only the messages involved in protocol (VSS, i, ℓ), and does so immediately after they are sent, all of this while scheduling the agents cyclically until all the agents stop sending messages. Then σ'_e appends $rec(d, i, \ell)$ to σ_e.

Note that σ'_e does not depend on the protocol $\vec{\tau}_T$ used by malicious agents. Suppose that $\vec{\pi} + \pi_d$ and σ_e are deterministic. For each input $\vec{x}$, let $J_i^k(\vec{x})$ denote agent i's local history at the end of its kth turn in the unique history of $(\vec{\pi} + \pi_d, \sigma_e, \vec{x})$. When the agents use $\vec{\pi}'$, they simulate the computation of $\vec{\pi} + \pi_d$. Let $K_i^k(\vec{x})$ denote i's history at the end of i's kth turn in the simulation. Although $\vec{\pi}'$ randomizes, since $\vec{\pi} + \pi_d$ and σ_e are deterministic, as we now show, the value of $K_i^k(\vec{x})$ is independent of this randomization.

Lemma 4. *For all input profiles $\vec{x}$, $K_i^k(\vec{x}) = J_i^k(\vec{x})$.*

We prove this lemma by proving a more general result that establishes a correspondence between histories of $\vec{\pi}$ and histories of $\vec{\pi}'$. In $\vec{\pi}'$, agents attempt to simulate all the events of $\vec{\pi} + \pi_d$, which include being scheduled and sending and receiving messages. By the construction of σ'_e, all shares of a message sent by the mediator are received by its recipient virtually "at the same time" (more precisely, they are received one immediately after the other, with no other action in between). This allows us to define a correspondence between events in a history h of $\vec{\pi} + \pi_d$ when used with scheduler σ_e and events in a history h' of $\vec{\pi}'$ when used with scheduler σ'_e. We start by defining the correspondence between events that are in an agent i's history in h and h'.

- The event that agent i is scheduled for the kth time in h corresponds to the

event that i is scheduled after receving its kth $proceed_i$ message in h'. (Of course, i may not have received k $proceed_i$ messages in h'; in this case, no event in h' corresponds to the event of i being scheduled for the kth time in h. Similar comments hold for all the other correspondences defined below.)

- The event that i sends its ℓth message in h corresponds to the event that i initiates its ℓth invocation of VSS in h'.
- The event that i receives the ℓth message sent by the mediator during its kth turn in h corresponds to the event that i receives a share of $g_{k,\ell}(h_{d,k})$ in h'. Note that $g_{k,\ell}$ encodes the ℓth message sent by the simulated mediator during its kth turn given its local history $h_{d,k}$.

Since the mediator is being simulated by all agents, events in the mediator's history in h do not correspond to single events in h'. Rather, they correspond to exactly n events, one for each agent. Scheduler σ'_e guarantees that these n events occur consecutively in h'.

- The event that the mediator is scheduled for the kth time in h corresponds to the set of events in h' consisting of agent i being scheduled after receiving a $proceed_{d,k}$ message, for each agent i.
- The event that the mediator sends the ℓth message to agent j during its kth turn in h corresponds to the set of events in h' consisting of each agent i computing its share of $g_{k,\ell}(h_{d,k})$ (which encodes the ℓth message sent by the simulated mediator during its kth turn) and sending it to agent j.
- The event that the mediator receives j's ℓth message in h corresponds to the set of events in h' consisting of each agent i terminating the ℓth VSS invocation initiated by j.

Note that we have not included the $done(i)$ events in the correspondence. Even though such events are needed to define then end of agent$i$'s turn, they are redundant, since they come immediately before a $sch(i)$ in i's local history.

An event in a history h' of $\vec{\pi}'$ is a *simulation event* if it could correspond to some event in another history h of $\vec{\pi}$. More precisely, an event e in history h' is a simulation event if there exists a history h of $\vec{\pi}$ and an event e in h such that e' corresponds to e. Two histories h and h' of $\vec{\pi}$ and $\vec{\pi}'$ *correspond* if all non-*done* events in i's history in h correspond to some event in h', all non-*done* events in the mediator's history in h correspond to n events in h, one for each agent, each simulation event in h' corresponds to some event in h, and the order of corresponding events in each agent i's history is preserved; more precisely, if e_1 and e_2 are two non-*done* events in i's (resp., the mediator's) history in h, and e'_1 and e'_2 are the events that correspond to e_1 and e_2 in i's history in h', then e_1 precedes e_2 in h iff e'_1 precedes e'_2 in h'.

Lemma 4 follows from the following lemma, which is almost immediate from the construction of σ'_e and π'. Although there are a number of histories in $\vec{\pi}'$ with scheduler σ'_e and input $\vec{x}$ due to the randomization used in protocols such as VSS and CC, all of them correspond to the unique history in $\vec{\pi}$ when the scheduler plays σ_e and agents have input profile $\vec{x}$.

Lemma 5. *For all input profiles $\vec{x}$, the unique history h of where the agents use $\vec{\pi}$ with scheduler σ_e and input profile $\vec{x}$ corresponds to all the histories h' of where the agents use $\vec{\pi}'$ with scheduler σ'_e and input profile $\vec{x}$.*

Note that Lemma 5 implies Lemma 4, since it states that agents simulate receiving and sending messages in exactly the same order with $\vec{\pi}'$ as they do with $\vec{\pi} + \pi_d$. Moreover, if the protocol is deterministic, the contents of those messages

are uniquely determined.

If $\vec{\pi}+\pi_d$ or σ_e involve randomization, we can assume that the agents and mediator toss all the coins they need at the beginning (before they are first scheduled) and then use the outcomes of these coin tosses for their decisions. Fixing the outcome of such coin tosses makes the protocols deterministic, and an analogous argument to that used for Lemma 4 for each of the possible sequences of coin tosses guarantees that the agents' outputs are identically distributed in $\vec{\pi}+\pi_d$ and in $\vec{\pi}'$. Since $(\vec{\pi}_{-T}, \vec{\tau}_T)$ is just another protocol, it immediately follows from Lemma 4 that for all input profiles $\vec{\pi}(\vec{x}, (T, \vec{\tau}_T, \sigma_e))$ and $\vec{\pi}'(\vec{x}, (T, \vec{\tau}'_T, \sigma'_e))$ are identically distributed for all possible protocols $\vec{\tau}_T$ for the malicious agents. This completes the proof that part (a) of the definition of "bisimulates" holds.

We now prove that part (b) of the definition of "bisimulates" holds. For this proof, we assume without loss of generality that malicious agents output their local history when running $\vec{\tau}'_T$, since any output must be a function of their local history. For ease of exposition, we begin by giving the highlights of the construction of $\vec{\tau}$ and σ_e, given $\vec{\pi}'$, $\vec{\tau}'_T$, and σ'_e; we later present the construction in more detail. The idea for constructing $\vec{\tau}_T$ and σ_e is that the adversary simulates what would occur if honest agents use $\vec{\pi}'_{-T}$, malicious agents use $\vec{\tau}'_T$, and the scheduler uses σ'_e. If the adversary in the protocol with the mediator knew the input $\vec{x}_{-T}$ of honest agents, the adversary could perform the simulation before the protocol starts, and have malicious agents output the local history they have in the simulation, regardless of their history in $\vec{\pi}$. However, the adversary does not know the honest agents' inputs. Thus, the adversary does the simulation assuming honest agents have some fixed input, which we take to be $\vec{0}_{-T}$. The following lemma makes precise the sense in which using $\vec{0}_{-T}$ rather than $\vec{x}_{-T}$ is "safe".

Lemma 6. *Let $J(\vec{\pi}, A, \vec{x})$ be a random a random variable whose values are the values of the malicious agents, when the honest agents use $\vec{\pi}$, given input profile $\vec{x}$, and adversary $A = (T, \tau_T, \sigma_e)$. Let $\vec{\sigma}_{VSS}$ and $\vec{\sigma}_{CC}$ be the implementation of VSS and CC, respectively, in a t-resilient secret-sharing scheme. Then for all adversaries $A = (T, \vec{\tau}_T, \sigma_e)$ and input profiles $\vec{x}$ and $\vec{x}'$, we have*

- *$J^e((\vec{\sigma}_{VSS}), A, \vec{x},)$ and $J^e((\vec{\sigma}_{VSS}), A, (\vec{x}_{-T}, \vec{x}'_{-T}))$ are identically distributed;*
- *$J^e((\vec{\sigma}_{CC}), A, \vec{x},)$ and $J^e((\vec{\sigma}_{CC}), A, (\vec{x}_{-T}, \vec{x}'_{-T}))$ are identically distributed;*

Lemma 6 implies that the adversary's history in an invocation of VSS and CC is independent of the actual inputs of the honest agents. This follows easily from the definition of t-resilience, since otherwise the adversary could deduce information about the honest agents' inputs given its local history. This means that much of the simulation can be performed by the adversary without having to know which inputs honest agents are using. Indeed, there are only three types of actions or decisions of an honest agent i that both depend on the honest agents' inputs and can affect the adversary's local history:

(a) how many time i invokes VSS after receiving a $proceed_i$ message;

(b) what values i broadcasts after computing its share of $f_{k,\ell}(h_{d,k})$;

(c) what values i sends to an agent $j \in T$ after computing $g_{k,\ell}(h_{d,k})$ (if $f_{k,\ell}(h_{d,k}) = j$).

Clearly, the number of times that an honest agent invokes VSS affects the adversary's simulated history. For (b) and (c), if the adversary assumes that the honest agents have arbitrary inputs, the values received by agents in T will also be arbitrary, as opposed to being correlated to the agents' inputs (e.g., the messages

sent by the mediator can depend in the messages received by honest agents, which ultimately depend on their inputs).

We show next that (a), (b) and (c) are the only decisions and actions taken by honest agents that the adversary cannot simulate. Suppose that the adversary had an oracle that could tell the adversary the number of times each honest agent i invokes VSS each time i is scheduled, and the values of each instance of $f_{k,\ell}(h_{d,k})$ and $g_{k,\ell}(h_{d,k})$ (if $f_{k,\ell}(h_{d,k}) \in T$). Then the adversary could perform its simulation even without the honest agents' inputs: it could run its simulation with arbitrary inputs for honest agents. Whenever an honest agent i is scheduled after receiving a $proceed_i$ message, it could ask the oracle how many times i invokes VSS, and could simulate i performing that many invocations of VSS with arbitrary inputs, even without knowing the actual local history of i. Similarly, whenever honest agents have to broadcast or send an agent in T their share of $f_{k,\ell}(h_{d,k})$ or $g_{k,\ell}(h_{d,k})$, the adversary could ask the oracle what value $f_{k,\ell}(h_{d,k})$ (resp., $g_{k,\ell}(h_{d,k})$) takes in the actual history of $\vec{\pi}'$. In its simulation, the adversary takes the set of shares that the honest agents broadcast or sent to agents in T to be $f_{k,\ell}(h_{d,k})$-realizable or $g_{k,\ell}(h_{d,k})$-realizable, respectively, regardless of the local history of the honest agents in the simulation (we will show when we present the more detailed construction how this can be done in such a way that the adversary's local history in its simulation is still consistent, despite the fact that in the simulation, honest agents may send different shares than the ones they computed). It follows from Lemma 6 that the adversary's local histories in the simulation with this oracle and its histories in a real interaction where honest agents play $\vec{\pi}'$ with input $\vec{x}$ are identically distributed.

Unfortunately, the adversary does not have access to such an oracle. However, by the construction of $\vec{\pi}'$, the values given by the oracle can be deduced from the

history of the protocol with the mediator that the adversary is simulating, even without the benefit of an oracle. Specifically, if, for all honest agent i, each of the simulated histories $h_{i,k}$ in $\vec{\pi}'$ is equal to i's local history at the end of i's kth turn in $\vec{\pi}$, and if each of the histories $h_{d,k}$ is equal to the local history of the mediator at the end of its kth turn; the number of times that an honest agent i invokes VSS after receiving its kth $proceed_i$ message but before receiving its $(k+1)$st $proceed_i$ message in $\vec{\pi}'$ is the number of messages sent by i during its kth turn in $\vec{\pi}$; $f_{k,\ell}(h_{d,k})$ is the recipient of the ℓth message sent by the mediator during its kth turn; and $g_{k,\ell}(h_{d,k})$ is the content of this message (which is known by the adversary if its recipient is in T).

Thus, if the adversary could schedule agents and deliver messages in $\vec{\pi}$ in such a way that, for each agent i and all k, the local histories $h_{i,k}$ in the adversary's simulation are the same as the local history of i in $\vec{\pi}$ at the end of i's kth turn, then we could dispense with the oracle. However, because the adversary does not know in the honest agents' input profiles, it cannot in general do this.

Fortunately, we do not need quite this much. Recall that the aim of the simulation is for the adversary to compute what history it would have in $\vec{\pi}'$. Since $|T| < n/4$, Lemma 6 implies that the local histories of the agents in T and of the scheduler have the same distribution, independent of which values are being shared by honest agents. Therefore, to deduce the values given by the oracle, it suffices that the adversary schedules agents and delivers messages in $\vec{\pi}$ in such a way that, for each agent $i \notin T$, the local histories $h_{i,k}$ in the adversary's simulation are the same as the local history of i in $\vec{\pi}$ at the end of i's kth turn, except possibly for i's input and the content of the messages sent and received by i. This means that the local histories $h_{i,k}$ and i's local histories in $\vec{\pi}$ at the end of its kth turn should

consist of exactly the same *sch*, *snd*, *rec*, *comp*, and *done* events, and may differ only in the content of their *rec* and *snd* events. A more detailed construction of the adversary $(T, \vec{\tau}_T, \sigma_e)$ is given next.

For simplicity, we assume the adversary is a single entity that controls both the scheduler and the subset T of malicious agents. Given $\vec{\pi}, \vec{\tau}'_T, \sigma'_e$, and $\vec{x}_T$, the adversary $(T, \vec{\tau}_T, \sigma_e)$. starts by performing a simulation of a history of $(\vec{\pi}'_{-T}, \vec{\tau}'_T, \sigma'_e, \vec{0}/_{(T,\vec{x}_T)})$ using additional variables $\alpha_{i,k,\ell}$ and $\beta_{i,k,\ell}$ with $i, k, \ell \in \mathcal{N}$, all initially set to a special value $\perp$. Whenever one of the following events occur in the simulation, the adversary proceeds as described below.

(1) An agent $i \notin T$ is scheduled after receiving a $proceed_i$ message: In this case, σ_e schedules i in $\vec{\pi}$. If i sends ℓ messages when it is scheduled in $\vec{\pi}$, then in its simulation, i is scheduled and invokes VSS ℓ times with input 0, regardless of i's local history i in the simulation.

(2) An honest agent $i \notin T$ terminates the share phase of (VSS, j, ℓ), with $j \in T$: Since $n > 4t$, the properties of VSS guaranteee that at least $2t + 1$ honest agents in the simulation will also compute their share of (VSS, j, ℓ), and that these shares reconstruct a unique value $mes_{j,\ell}$. The scheduler σ_e then schedules agent j in $\vec{\pi}$, and j sends message $mes_{j,\ell}$ to the mediator, tagged with label ℓ.

(3) An agent $i \notin T$ is the first honest agent to compute the permutation θ_k computed in phase $k2$. Let m_k be the cardinality of the domain of of θ_k, and let $(VSS, j_1, \ell_1), \ldots, (VSS, j_{m_k}, \ell_{m_k})$ be the invocations of VSS that are included in $h_{d,k}$ in Phase $k1$. The scheduler σ_e delivers agent $j_{\theta_k(1)}$'s $\ell_{\theta_k(1)}$th message, ..., and $j_{\tau(n)}$'s $\ell_{\theta_k(n)}$th message to the mediator in $\vec{\pi}$, and then schedules the mediator.

(4) An agent $i \notin T$ terminates the invocation of CC instance for $f_{k,\ell}(h_{d,k})$. If the additional variable $\alpha_{i,k,\ell} \neq \perp$, then the adversary continues its simulation under the assumption that i's output of the CC invocation is $\alpha_{i,k,\ell}$ rather than the actual output. Otherwise, since $n > 4t$, the assumption made at the end of Section 2.2.1 guarantee that there is a set I of at least $2t + 1$ honest agents in the simulation that have already computed their shares of $f_{k,\ell}(h_{d,k})$. Let $\{s_{i,k,\ell}\}_{i\in[n]}$ be the unique full extension of the shares of the agents in I, and let $j_{k,\ell}$ be either the receiver of the mediator's ℓth message during its kth turn or 0 if the mediator didn't send ℓ messages during its kth turn. The adversary samples uniformly at random a full $j_{k,\ell}$-extension $\{s'_{i,k,\ell}\}_{i\in[n]}$ of $\{s_{i,k,\ell}\}_{i\in T}$ and sets $\alpha_{i,k,\ell}$ to $s'_{i,k,\ell}$ for all $i \in [n]$. The adversary continues its simulation by assuming that i's output of the CC invocation is $\alpha_{i,k,\ell}$ rather than $s_{i,k,\ell}$.

(5) An agent $i \notin T$ terminates the CC invocation for $g_{k,\ell}(h_{d,k})$, and k and ℓ are such that the mediator sent at least ℓ messages the kth time it was scheduled in $\vec{\pi}$, and the recipient $j_{k,\ell}$ of the ℓth message sent by the mediator the kth time it is scheduled in $\vec{\pi}$ is in T. If $\beta_{i,k,\ell} \neq \perp$, then the adversary continues its simulation by assuming that i's output of the CC invocation is $\beta_{i,k,\ell}$ rather than the actual output. Otherwise, let $mes_{k,\ell}$ be the content of the ℓth message that the mediator sends the kth time it is scheduled (note that this value is known by $j_{k,\ell}$, and hence by the adversary). Since $n > 4t$, by the properties of CC, there is a set I with at least $2t + 1$ honest agents that have computed their shares of $g_{k,\ell}(h_{d,k})$ in the simulation. Let $\{s''_{i,k,\ell}\}_{i\in[n]}$ be the unique full extension of the shares of agents in I. The adversary samples uniformly at random a full $mes_{k,\ell}$-extension of $\{s''_{i,k,\ell}\}_{i\in T}$ and sets $\beta_{i,k,\ell}$ to $s''_{i,k,\ell}$ for each $i \in [n]$. Then the adversary continues its simulation by

assuming that i's output of the CC invocation is $\beta_{i,k,\ell}$ rather than $s''_{i,k,\ell}$.

(6) An agent $i \notin T$ reconstructs $g_{k,\ell}(h_{d,k})$ using VSS for some k and ℓ. In this case, the scheduler σ_e delivers the ℓth message sent by the mediator to i the kth time the mediator was scheduled.

Note this construction for the adversary is well defined. The first and second clause guarantee that if an agent terminated (VSS, i, ℓ) in the adversary's simulation, then agent i sent a message tagged with label ℓ in the corresponding history of $\vec{\pi}$. Also, whenever an honest agent finishes the computation of $f_{k,\ell}(h_{d,k})$ or $g_{k,\ell}(h_{d,k})$, it must have terminated the computation of θ_k, and (3) guarantees that the mediator has been scheduled at least k times (as needed for (4) and (5)).

The first step in proving that this construction of $\vec{\tau}_T$ and σ_e satisfies clause (b) of the definition of t-bisimulation is to show that the adversary can simulate how many times each honest agent invokes VSS in $\vec{\pi}'$. Recall that honest agents may initiate a new invocation of only after receiving a *proceed* message. Given a history h in $\vec{\pi}'$, let $a_{i,k}(h)$ denote the number of times that agent i invokes VSS after receiving its kth $proceed_i$ message but before receiving the $(k+1)$st $proceed_i$ message. Let $((a_{i,k}(h))_{i \notin T})_{k \in \mathbb{N}}$ be the sequence of all such values, arranged lexicographically first by their k index and then by their i index. The following lemma, which follows immediately from the construction of $\vec{\tau}_T$ and σ_e, shows that the distribution of these random variables is the same in $\vec{\pi}'$ and in the adversary's simulation.

Lemma 7. *Fix $T \subseteq [n]$ with $|T| < n/4$ and an input profile $\vec{x}$. Let H be the distribution over histories when agents use $(\vec{\pi}'_{-T}, \vec{\tau}'_T)$ with scheduler σ'_e and input $\vec{x}$, and let H' be the distribution over histories in the adversary's simulation when agents use $(\vec{\pi}_{-T}, \vec{\tau}_T)$ with scheduler σ_e and input $\vec{x}$. Then $S(H)_T$ and $S(H')_T$ are identically distributed.*

As we pointed out before, one of the only decisions made by honest agents that the adversary cannot simulate without additional information is the number of invocations of VSS that they perform when they are scheduled. Thus, this lemma shows that honest agents behave exactly the same in $\vec{\pi}'$ and in the adversary's simulation except for the values that they share using VSS and the values of $f_{k,\ell}(h_{d,k})$ and $g_{k,\ell}(h_{d,k})$ sent to other agents. However, by Lemma 6, exactly which values are shared using VSS does not affect the adversary's local history in its simulation. Therefore, the only events that might differ between the adversary's local history in $\vec{\pi}'$ and its local history in its simulation in $\vec{\pi}$ are those in which agents in T receive shares of $f_{k,\ell}(h_{d,k})$ and $g_{k,\ell}(h_{d,k})$ from honest agents. In $\vec{\pi}'$, $f_{k,\ell}(h_{d,k})$ and $g_{k,\ell}(h_{d,k})$ are the recipient and the content of the mediator's ℓth message during its kth turn, which are computed using CC with the simulated mediator's local history $h_{d,k}$ as input. Since the adversary assumes in its simulation that the values that honest agents share using VSS are all 0, the local histories $h_{d,k}$ that honest agents compute in the adversary's simulation would not follow the same distribution as their local histories if they used $\vec{\pi}'$ with their actual input. Thus, the shares of $f_{k,\ell}(h_{d,k})$ and $g_{k,\ell}(h_{d,k})$ would also have a different distribution.

However, much as when dealing with VSS invocations, the adversary can use the mediator's actions in $\vec{\pi}$ as feedback for its simulation, and simulates that the shares of $f_{k,\ell}(h_{d,k})$ and $g_{k,\ell}(h_{d,k})$ that honest agents send to agents in T define secrets $j_{k,\ell}$ and $\mu_{k,\ell}$ respectively, regardless of their local history, where $j_{k,\ell}$ and $\mu_{k,\ell}$ are the recipient and the content of the mediator's ℓth message during its kth turn in $\vec{\pi}$ (note that the adversary does this for $g_{k,\ell}(h_{d,k})$ whenever $j_{k,\ell} \in T$, since otherwise it does not know the content of this message). By construction, this simulation proceeds in such a way that the shares that honest agents send are "consistent" with the shares of $f_{k,\ell}(h_{d,k})$ and $g_{k,\ell}(h_{d,k})$ that agents in T could

compute from their local history, in the sense that the shares of $f_{k,\ell}(h_{d,k})$ and $g_{k,\ell}(h_{d,k})$ sent by honest agents together with the shares of those functions that agents in T could compute are $j_{k,\ell}$-realizable and $\mu_{k,\ell}$-realizable respectively. Note that if this weren't the case, the simulated local history of agents in T could not occur if honest agents played $\vec{\pi}'$. The following lemma shows the correctness of this construction.

Given a history h of $\vec{\pi}'$, let $R^T_{k,\ell}(h)$ denote the subsequence of *rec* events in h_T of messages from agents not in T involving the computation of $f_{k,\ell}(h_{d,k})$ and $g_{k,sharesof\ell}(h_{d,k})$, using CC, the broadcast procedures for the shares of $f_{k,\ell}(h_{d,k})$, and the messages in which they send their shares of $g_{k,\ell}(h_{d,k})$.

Lemma 8. *Fix $T \subseteq [n]$ with $n > 4|T|$ and an input profile $\vec{x}$. Let $\vec{\pi}_h(\vec{x}, A)$ be the distribution over histories when agents use $\vec{\pi}$ with adversary A and input $\vec{x}$, and let $A := (T, \vec{\tau}_T, \sigma_e))$ and $A' := (T, \vec{\tau}'_T, \sigma'_e)$. Then $R^T_{k,\ell}(\vec{\pi}'_h(\vec{x}, A')$ and $R^T_{k,\ell}(\vec{\pi}_h(\vec{x}, A))$ are identically distributed for all k, ℓ, where $R^T_{k,\ell}(\vec{\pi}_h(\vec{x}, A))$ is the distribution on sequences of rec events defined naturally by composing $\vec{\pi}_h(\vec{x}, A)$ and $R^T_{k,\ell}$.*

Proof. The proof of this lemma is analogous to that given by Canetti in his proof of Lemma 4.31 [16, p. 91]. □

As we noted in Section 3.3.1, we implement CC in such a way that there is no correlation between the shares of different circuit computations. Thus, Lemma 8 can be easily generalized to show that $(R_{k,\ell}(H))_{k,\ell\in\mathbb{N}}$ and $(R_{k,\ell}(H'))_{k,\ell\in\mathbb{N}}$ are identically distributed. This, together with Lemma 6 and Lemma 7 implies Theorem 3(a) in the case that $t = t'$.

We now prove that $\vec{\pi}'$ (t, t')-bisimulates $\vec{\pi}$ in the general setting, where $3t+t' < n$ and $t \geq t'$. Let $t'' = t - t'$. Given a protocol $\vec{\pi}$ for n agents, consider the protocol

$(\vec{\pi}, \eta)$ for $n + t''$ agents, where the first n agents use $\vec{\pi}$, while the last t'' agents use the null protocol, that is, they never send any messages. Given an adversary $A' = (T, \vec{\tau}'_T, \sigma'_e)$ in the setting with n agents (and no mediator), consider $\vec{\alpha}'$ an adversary $A'' = (T, \vec{\tau}''_T, \sigma''_e)$ in the setting with $n + t''$ agents in which σ''_e is a relaxed scheduler that acts just like σ'_e, except that it might schedule agents in $\{n + 1, \ldots, n + t''\}$, although it never delivers their messages (note that since σ''_e does not deliver the messages sent by the last t'' agents, τ' is well defined even in the setting with $n + t''$ agents). Since $4t < n + t''$, following the same construction as in the case that $t = t'$, there exists an adversary $A = (T, \vec{\tau}_T, \sigma_e)$ such that

$$(\vec{\pi} + \eta)(\vec{x}, A'') = (\vec{\pi} + \eta)(\vec{x}, A)$$

for all input profiles $\vec{x}$. Note that the scheduler σ_e resulting from this construction is in fact a relaxed scheduler: since some of the VSS-share and CC instances in $(\vec{\pi} + \eta)'$ are not guaranteed to terminate if σ''_e is relaxed, some messages might not be delivered by σ_e.

Consider a scheduler σ'''_e in $\vec{\pi}$ that acts like σ_e except that it does not schedule agents $n + 1, \ldots, n + t$. By construction,

$$\vec{\pi}(\vec{x}, (T, \vec{\tau}_T, \sigma'''_e)) = (\vec{\pi} + \eta)(\vec{x}, A)$$

and

$$\vec{\pi}'(\vec{x}, A') = (\vec{\pi} + \eta)(\vec{x}, A'')$$

so

$$\vec{\pi}(\vec{x}, (T, \vec{\tau}_T, \sigma'''_e)) = \vec{\pi}'(\vec{x}, A')$$

as desired.

Finally, it remains to show that if $|T| \leq t'$, σ'''_e is not relaxed. Given the adversaries A' and A'' defined above, consider an adversary $A^* := (\vec{\tau}^*_{T'}, \sigma^*_e)$ for

$(\vec{\pi}+\eta)'$, such that $T' = T \cup \{n+1, \ldots, n+t''\}$, agents $i \in T$ use $\vec{\tau}'_i$, agents in $\{n+1, \ldots, n+t''\}$ send no messages, and σ^*_e acts just like σ''_e. By construction,

$$(\vec{\pi}+\eta)'(\vec{x}, A^*) = (\vec{\pi}+\eta)(\vec{x}, A'')$$

In this case, σ^*_e is not a relaxed scheduler, since agents in $\{n+1, \ldots, n+t''\}$ never send any messages. Moreover, since $|T'| = |T| + t'' < t$, it follows that $4|T'| < n$, and reasoning analogous to the previous case shows that there exists an adversary $A = (T, \vec{\tau}, \sigma'''_e)$ such that

$$\vec{\pi}(\vec{x}, A) = \vec{\pi}'(\vec{x}, A').$$

However, in this case, σ'''_e is not relaxed, since σ''_e was not.

3.3.4 Bounding the number of messages

As mentioned in Section 2.2.2, our construction of $\vec{\pi}'$ does not bound the number of messages sent. To see this, note that agents compute $h_{d,k}$ each time that the mediator is scheduled in the simulation. Since the number of times that the mediator can be scheduled is unbounded, the number of messages sent in $\vec{\pi}'$ can be unbounded as well.

If the mediator π_d is responsive, we show how we can modify the construction of Section 2.2.2 so as to bound the number of messages. The idea is that, since π_d is responsive, agents don't need to simulate all the mediator's histories; it suffices to simulate only the histories in which the mediator receives at least one message at every turn except possibly the first one. Note that this bounds the number of mediator turns that the agents simulate by N, and thus guarantees that the expected number of messages in $\vec{\pi}'$ is polynomial in n and N since all

primitives satisfy this property. To do this, agents run the protocol $\vec{\pi}'$ described in Section 3.3.3 with a simple modification in the computation of $h_{d,k}$ for $k > 1$. Instead of running a consensus protocol $p_{j,\ell,k}$ for each VSS invocation (VSS, j, ℓ), agents use an ACS computation C_k with parameter $m = 1$, in which the accumulative set U_i of agent i consists of the pairs (j, ℓ) such that i has terminated (VSS, j, ℓ) but (j, ℓ) was not in any core set $C_{k'}$ with $k' < k$. Agents then continue Phases $k2$ to $k4$ as usual, but take $p_{j,\ell,k} = 1$ iff $(j, \ell) \in C_k$. Since the agents take the parameter m to be 1, $|C_k| \geq 1$; thus, it is guaranteed that, in the simulation, the mediator has received at least one message at its kth turn (note that, in this construction, phases $k1, k2, k3$ and $k4$ are run $O(N)$ times).

The proof of correctness of this modified construction is identical to that given in Section 3.3.3 for the original construction.

3.3.5 The proof of Theorem 3(b)

If $\vec{\pi} + \pi_d$ is in canonical form, the construction of $\vec{\pi}'$ is as in Section 2.2.2, except that if an honest agent reconstructs a message containing "STOP", it terminates.

Suppose that a set I of at least $2t+1$ honest agents terminate. This means that all agents in I have computed their share of each of the mediator's messages. Thus, for each message μ sent by the mediator, each honest agent i will eventually receive a subset S^I_μ of shares such that (I, S^I_μ) is μ-realizable. Recall that we assumed (at the end of Section 2.2.1) that the secret-sharing scheme used in π' is t-determinate. Thus, this subset of shares suffices for each honest agent to uniquely reconstruct μ, even with an adversary of size t: if a pair (I, S) with $|I| \leq 2t+1$ is μ-realizable, at least $t + 1$ agents from I are honest, and their shares uniquely define μ (and

each of the other agents' shares). Thus, receiving a realizable set of at least $2t+1$ shares uniquely determines the secret being shared.

3.3.6 The proof of Theorems 4 and 5

The protocol $\vec{\pi}'$ for Theorem 4 is analogous to that for Theorem 3, except that we use the VSS and CC implementations of BKR instead of those of BCG. The proof of Theorem 4(a) is then identical to that of Theorem 3(a). Since it can be easily shown that the VSS and CC implementations constructed by BKR ϵ-$(t, t+1)$-coterminate, Theorem 4(b) follows.

The construction $\vec{\pi}'$ for Theorem 5 is also analogous to that for Theorem 3, except that we use the VSS and CC implementations of BGW and players don't have to send *Ready* messages to themselves.

3.4 Other Asynchronous Models

In this section, we show that the choices made in our formal model in Section 1.1 are essentially being made without loss of generality. We start by considering our assumption that agents perform a sequence of actions atomically when they are scheduled. We next show that we would get theorems essentially equivalent to Theorems 3 and 4 if we had instead assumed that agents perform just a single action when they are scheduled. To prove this, we first need the following notion:

Definition 8. *A protocol $\vec{\pi}$ is* N-message bounded *if for all inputs and all histories, no agent ever sends more than N messages in a single turn. A protocol is* message bounded *if it is N-message bounded for some N.*

Proposition 4. *There exist a function H from message-bounded protocols to single-action protocols such that for all protocols $\vec{\pi}$, the following holds:*

(a) For all schedulers (resp., relaxed schedulers) σ_e there exists a scheduler (resp., relaxed scheduler) σ'_e such that, for all input profiles $\vec{x}$, $\vec{\pi}(\vec{x}, \sigma_e)$ and $H(\vec{\pi})(\vec{x}, \sigma'_e)$ are identically distributed, where we take $H(\vec{\pi}) = (H(\pi_1), \ldots, H(\pi_n))$ and we view σ_e and σ'_e, respectively, as the adversaries (i.e., we take $T = \emptyset$).

(b) For all schedulers (resp., relaxed schedulers) σ'_e there exists a scheduler (resp., relaxed scheduler) σ_e such that, for all input profiles $\vec{x}$, $\vec{\pi}(\vec{x}, \sigma_e)$ and $H(\vec{\pi})(\vec{x}, \sigma'_e)$ are identically distributed.

The converse of Proposition 4 is trivial, since single-action protocols are protocols. It follows from Proposition 4 that Theorem 3 holds even if we restrict agents to using single-action protocols (note that canonical protocols are message bounded).

Proof. Intuitively, $H(\pi_i)$ is identical to π_i, except that rather than sending a sequence of messages when it is scheduled, i sends the messages one at a time. The scheduler σ'_e is then chosen to ensure that i is scheduled so that it sends all of its messages as if they were sent atomically. In addition to keeping track of the messages it has sent and received, i uses the variable U_i whose value is a sequence of messages (intuitively, the ones that i would have sent at this point in the simulation of π_i that it has not yet sent), initially set to the empty sequence, and a binary variable *next*, originally set to 1. When i is scheduled by σ'_e, $H(\pi_i)$ proceeds as follows: If $next = 1$, then i sets U_i to the sequence of messages that it would send with π_i given its current history. (If π_i randomizes, then $H(\pi_i)$ does the same

randomization. If U_i is the empty sequence (so π_i would not send any messages at that point), i performs the action $done(i)$, and outputs whatever it does with π; otherwise, i sets *next* to 0, sends the first message in U_i to its intended recipient, and removes this message from U_i. If $next = 0$, then if U_i is empty, i sets *next* to 1, sends $done(i)$, and outputs whatever it does with π; otherwise, i sends the first message in U_i to its intended recipient and removes it from U_i.

Since $\vec{\pi}$ is message bounded, there exists an N such that $\vec{\pi}$ is N-message bounded. For part (a), given σ_e, we construct σ'_e so that it simulates σ_e, except that if σ_e schedules i, σ'_e schedules i repeatedly until either it observes $done(i)$ or until i sends messages in $N+1$ consecutive turns. Since $\vec{\pi}$ is N-message bounded, it is clear that $\vec{\pi}(\vec{x}, \sigma_e)$ and $H(\vec{\pi})(\vec{x}, \sigma'_e)$ are identically distributed. Note that it is necessary for $\vec{\pi}$ to be N-message bounded, since if the scheduler schedules each agent i repeatedly until it stops sending a message during its turn, an agent that keeps sending messages would be scheduled indefinitely, and so would prevent other agents from being scheduled.

For part (b), given σ'_e, we construct σ_e so that it simulates σ_e. There is one issue that we have to deal with. Whereas with σ_e, an agent i can send k messages each time it is scheduled, with σ'_e, it can send only one message when it is scheduled. The scheduler σ'_e constructed from σ_e in part (a) scheduled i repeatedly until it sent all the messages it did with σ_e. But we cannot assume that the scheduler σ'_e that we are given for part (b) does this. Thus, σ_e must keep track of how many of the messages that each agent i was supposed to send the last time it was scheduled by σ_e have been sent so far. To do this, σ_e uses variables mes_i, one for each agent i, initially set to 0, such that mes_i keeps track of how many of the messages that agent i sent with σ_e still need to be sent by σ'_e. As we observed above, given a local

history h of the scheduler where the agents use $\vec{\pi}$ and the scheduler uses σ_e, there is a corresponding local history h' of the scheduler where the agents use $\vec{\pi}'$ and the scheduler uses σ'_e. If, given h', σ'_e schedules agent i with probability α_i, then with the same probability α_i, σ_e proceeds as follows: if $mes_i = 0$ (which means that all the messages that i sent the last time it was scheduled have been delivered in h'), then σ_e schedules i, sees how many messages i delivers according π_i, and sets mes_i to this number; if $mes_i \neq 0$, then mes_i is decremented by 1 but no agent is scheduled. Again, it is clear that that $\vec{\pi}(\vec{x}, \sigma_e)$ and $H(\vec{\pi})(\vec{x}, \sigma'_e)$ are identically distributed. □

BCG put further constraints on the scheduler. Specifically, they assume that, except possibly for the first time that agent i is scheduled, i is scheduled immediately after receiving a message and only then. That is, in our terminology, BCG assume that a $rec(\cdot, \cdot, i)$ event must be followed by a $sch(i)$ event, and all $sch(i)$ events except possibly the first one occur after a $rec(\cdot, \cdot, i)$ event. We call the schedulers that satisfy this constraint *BCG schedulers*.

We now prove a result analogous to Proposition 4, from which it follows that we could have obtained our results using a BCG scheduler.

Proposition 5. *There exist a function H from protocols to protocols such that for all protocols $\vec{\pi}$ the following holds:*

(a) For all schedulers (resp., relaxed schedulers) σ_e there exists a BCG scheduler (resp., relaxed BCG scheduler) σ'_e such that, for all input profiles $\vec{x}$, $\vec{\pi}(\vec{x}, \sigma_e)$ and $H(\vec{\pi})(\vec{x}, \sigma'_e)$ are identically distributed.

(b) For all BCG schedulers (resp., relaxed schedulers) σ'_e there exists a scheduler (resp., relaxed scheduler) σ_e such that, for all input profiles $\vec{x}$, $\vec{\pi}(\vec{x}, \sigma_e)$ and

$H(\vec{\pi})(\vec{x}, \sigma'_e)$ *are identically distributed.*

Proof. As in Proposition 4, the idea is that σ'_e simulates σ_e, but since σ_e can schedule an agent only when it delivers a message, we have each agent i send itself special messages, denoted $proceed_i$, to ensure that there are always enough messages in the system. In more detail, $H(\pi_i)$ works as follows. When it is first scheduled, agent i sends itself a $proceed_i$ message. Since we are considering BCG schedulers, agent i is scheduled subsequently only when it receives a message. If it receives a message other than $proceed_i$, it does nothing (although the message is added to its history). If it receives a $proceed_i$ message, then it does whatever it would do with π_i given its current history with the $proceed_i$ messages and the $sch(i)$ events not preceded by a $proceed_i$ message removed, and sends itself another $proceed_i$ message.

For part (a), given σ_e, σ'_e first schedules each agent once (in some arbitrary order), to ensure that that each of them has sent a $proceed_i$ message that is available to be delivered. Given a history h', σ'_e considers what σ_e would do in the history h that results from h' by removing the initial $sch(i)$ event for each agent i, the last message that each agent i sends when it is scheduled if it sends a message at all, and the receipt of these messages. If h' is a history that results where the agents are running $H(\vec{\pi})$, then the send and receive events removed are precisely those that involve $proceed_i$. If σ_e delivers a message with some probability, then σ'_e delivers the corresponding message with the same probability; if σ_e schedules an agent i with some probability, σ'_e delivers the last $proceed_i$ that i sent and schedules agent i with the same probability. If there is no $proceed_i$ message to deliver, then σ'_e does nothing, but our construction of $H(\pi_i)$ guarantees that if h' is a history that results from running $H(\vec{\pi})$, then there will be such a message that can be delivered.

Again, it is clear that $\vec{\pi}(\vec{x}, \sigma_e)$ and $H(\vec{\pi})(\vec{x}, \sigma'_e)$ are identically distributed.

For part (b), given σ'_e, the construction of σ_e is similar to that of Proposition 4. Again, given a local history h of σ_e where the agents use $\vec{\pi}$, there is a corresponding history h' of σ'_e where the agents use $H(\vec{\pi})$. If, given input h', σ'_e delivers a message with some probability p and the messages is not a $proceed_i$ message, then σ_e delivers the corresponding message with probability p. If the message is a $proceed_i$ message, then σ_e also schedules agent i. If σ'_e schedules an agent i with probability p, and in h' this is the first time that i is scheduled, then σ_e schedules i with probability p and otherwise does nothing with probability p. Yet again, it is straightforward to show that $\vec{\pi}(\vec{x}, \sigma_e)$ and $H(\vec{\pi})(\vec{x}, \sigma'_e)$ are identically distributed. □

CHAPTER 4

IMPLEMENTING MEDIATORS WITH CHEAP TALK

4.1 Mediator and Cheap Talk Games

Up to this point, we assumed that the adversary could act arbitrarily, which is what standard in the computer science literature. In contrast, in the game theory literature, the assumption is that players have preferences and seek to maximize their utility; thus, they will subvert the computation if and only if it is in their best interest to do so. As we have seen in chapter 3, we can simulate any interaction with a trusted mediator if the number of deviating players is less than a fourth of the total. In this chapter we are interested on what are the necessary conditions such that if an equilibrium can be reached with a mediator, the same equilibrium could also be reached without the mediator by having the players communicate between them. The following definitions - most of them taken from ADGH [1] - are necessary to illustrate these concepts.

We consider three games: an *underlying game* Γ, an extension Γ_d of Γ with a mediator, and an extension Γ_{CT} of Γ with (asynchronous) cheap talk. We assume that Γ is a *normal-form Bayesian game*: each player has a type t taken from some type space $\mathcal{T}_i$, such that there is a commonly known distribution on $\mathcal{T} \subseteq \mathcal{T}_1 \times \cdots \times \mathcal{T}_n$, the set of types; each player i chooses an action $a \in A_i$, the set of actions of agent i; player i's utility u_i is determined by the type profile of the players and the actions they take. A strategy for player i in the Bayesian game is just a function T_i to A_i, which tells player i what to do, given his type. If

$A = A_1 \times \cdots \times A_n$, then a strategy profile $\vec{\sigma} = (\sigma_1, \ldots, \sigma_n)$ can be viewed as a function $\vec{\sigma} : \mathcal{T} \to \Delta(A)$ (where, as usual, $\Delta(X)$ denotes the set of probability distributions on X).

A game Γ' extends Γ if the players have initial types from the same type space as Γ, with the same distribution over types; moreover, in each path of the game tree for Γ', the players send and receive messages, and perform at most one action from Γ. In a history where each player makes a move from Γ, each player gets the same utility as in Γ (where the utility is a function of the moves made and the types). That leaves open the question of what happens in a complete history of Γ' where some players do not make a move in Γ. As we suggested in the introduction, we consider two approaches to dealing with this. In the first approach, we assume that the description of Γ' includes a function M_i for each player i that maps player i's type to a move in Γ. In an infinite history h where i has type t and does not make a move in Γ, i is viewed as having made move $M_i(t)$. We can then define each player's utility in h as above. This is the *default-move approach*. In the Aumann and Hart approach [8], we extend the notion of strategy so that i's strategy in Γ' also describes what move i makes in the underlying game Γ in any infinite history h where i has not made a move in Γ. In the AH approach, i's move in h is under i's control; in the default-move approach, it is not. We believe that both the AH approach and the default-move approach are both reasonable in different contexts. The AH approach makes sense if the agent can leave a "will"; i.e., instructions as to what to do in the event of an infinite history, that will be carried out somehow. But if we consider a game-theoretic variant of Byzantine agreement, it seems more reasonable to say that if a "malicious" agent can prevent an agent from making a move in finite time, the agent should not get a chance to make a move after the cheap-talk phase has ended.

Given an underlying Bayesian game Γ (which we assume is synchronous—the players move simultaneously), we will be interested in two types of extensions. A *mediator game* extending Γ is an asynchronous game where players can send messages to and receive messages from a mediator (who can be viewed as a trusted third party) as well as making a move in Γ; "good" or "honest" players do not send messages to each other, but "bad" players (i.e., one of the k rational deviating players or one of the t "malicious" players with unknown utilities) may send messages to each other as well as to the mediator. We assume that the space of possible messages that can be sent in a mediator game is fixed and finite.

In a cheap-talk game extending Γ, there is no mediator. Players send messages to each other as well as making a move in Γ. We assume that each pair of agents communicates over an *authenticated private channel*, so the adversary cannot eavesdrop on conversations between the players, and players can identify the sender of each message.

4.2 Solution concepts

In this section, we review the solution concepts introduced in ADGH and extend them to asynchronous settings. The definitions in the synchronous case are equivalent, except that there is no environment.

Note that in an asynchronous game Γ, the utility of a player i can depend not only on the strategies of the agents, but on what the environment does. Since we consider an underlying game, a mediator game, and a cheap-talk game, it is useful to include explicitly in the utility function which game is being considered. Thus, we write $u_i(\Gamma, \vec{\sigma}, \sigma_d, \sigma_e, \vec{x})$ to denote the expected utility of player i in game Γ

when players play strategy profile $\vec{\sigma}$, the mediator plays σ_d, the environment plays σ_e, and the type profile is $\vec{x}$. We typically say "input profile" rather than "type profile", since in our setting, the type of player i is just i's initial input. Note that if Γ is the underlying game, the σ_e component is unnecessary, since the underlying game is assumed to be synchronous. We occasionally omit the mediator strategy σ_d when it is clear from context.

Given a type space $\mathcal{T}$, a set K of players, and $\vec{x} \in \mathcal{T}$, let $\mathcal{T}(\vec{x}_K) = \{\vec{x}' : \vec{x}'_K = \vec{x}_K\}$. If Γ is a Bayesian game over type space $\mathcal{T}$, $\vec{\sigma}$ is a strategy profile in Γ, and Pr is the probability on the type space $\mathcal{T}$, let

$$u_i(\Gamma, \vec{\sigma}, \sigma_e, \vec{x}_K) = \sum_{\vec{x}' \in \mathcal{T}(\vec{x}_K)} \Pr(\vec{x}' \mid \mathcal{T}(\vec{x}_K))\, u_i(\Gamma, \vec{\sigma}, \sigma_e, \vec{x}').$$

Thus, $u_i(\Gamma, \vec{\sigma}, \sigma_e, \vec{x}_K)$ is i's expected payoff if everyone uses strategy $\vec{\sigma}$ and type profiles are in $\mathcal{T}(\vec{x}_K)$.

k-resilient equilibrium: In a standard game, a strategy profile is a Nash equilibrium if no player can gain any advantage by using a different strategy, given that all the other players do not change their strategies. The notion of k-resilient equilibrium extends Nash equilibrium to allow for coalitions.

Definition 9. *$\vec{\sigma}$ is a* k-resilient equilibrium *(resp.,* strongly k-resilient equilibrium*) in an asynchronous game Γ if, for all subsets K of players with $1 \leq |K| \leq k$, all strategy profiles $\vec{\tau}_K$ for the players in K, all type profiles $\vec{x} \in \mathcal{T}$, and all strategies σ_e of the environment, $u_i(\Gamma, (\vec{\sigma}_{-K}, \vec{\tau}_K), \sigma_e, forall\vec{x}_K) \leq u_i(\Gamma, \vec{\sigma}, \sigma_e, \vec{x}_K)$ for some (resp., all) $i \in K$.* [1]

Thus, $\vec{\sigma}$ is k-resilient if, no matter what the environment does, no subset K of at most k players can all do better by deviating, even if they share their type

[1] As usual, in the strategy profile $(\vec{\sigma}_{-K}, \vec{\tau}_K)$, each player $i \in K$ plays τ_i and each player $i \notin K$ plays σ_i.

information (so that if the true type is $\vec{x}$, the players in K know $\vec{x}_K$). It is strongly k-resilient if not even one of the players in K can do better if all the players in K deviate.

For some of our results we will be interested in equilibria that are "almost" k-resilient, in the sense that no player in a coalition can do more than ϵ better if the coalition deviates from the protocol, for some small ϵ.

Definition 10. *For $\epsilon > 0$, $\vec{\sigma}$ is an ϵ-k-*resilient equilibrium *(resp.,* strongly ϵ-k-resilient equilibrium*) if, for all subsets K of players, all strategy profiles $\vec{\tau}_K$ for the players in K, all type profiles $\vec{x} \in \mathcal{T}$, and all strategies σ_e of the environment, we have $u_i(\Gamma, (\vec{\sigma}_{-K}, \vec{\tau}_K), \sigma_e, \vec{x}_K) < u_i(\Gamma, \vec{\sigma}, \sigma_e, \vec{x}_K) + \epsilon$ for some (resp., for all) $i \in K$.*

Note that we have "$< u_i(\Gamma, \vec{\sigma}, \sigma_e, \vec{x}_K) + \epsilon$" here, not "$\leq$"; this means that a 0-$k$-resilient equilibrium is not a k-resilient equilibrium. However, an equilibrium is k-resilient iff it is ϵ-k-resilient for all $\epsilon > 0$. We have used this slightly nonstandard definition to make the statements of our theorems cleaner.

Robustness: A standard assumption in game theory is that utilities are (commonly) known; when we are given a game we are also given each player's utility. When players make decision, they can take other players' utilities into account. However, in large systems, it seems almost invariably the case that there will be some fraction of users who do not respond to incentives the way we expect. For example, in a peer-to-peer network like Kazaa or Gnutella, it would seem that no rational agent should share files. Whether or not you can get a file depends only on whether other people share files; on the other hand, it seems that there are disincentives for sharing (the possibility of lawsuits, use of bandwidth, etc.). Nevertheless, people do share files. However, studies of the Gnutella network have shown almost 70 percent of users share no files and nearly 50 percent of responses

are from the top 1 percent of sharing hosts [6].

It seems important to design protocols that tolerate such unanticipated behaviors, so that the payoffs of the users who follow the recommended strategy are not affected by players who deviate, provided that not too many deviate.

Definition 11. *A strategy profile $\vec{\sigma}$ is* t-immune *in a game Γ if, for all subsets T of players with $|T| \leq t$, all strategy profiles $\vec{\tau}$, all $i \notin T$, all type profiles $\vec{x} \in \mathcal{T}$, and all strategies σ_e of the environment, we have $u_i(\Gamma, (\vec{\sigma}_{-T}, \vec{\tau}_T), \sigma_e, \vec{x}_T) \geq u_i(\Gamma, \vec{\sigma}, \sigma_e, \vec{x}_T)$.*

Intuitively, $\vec{\sigma}$ is t-immune if there is nothing that players in a set T of size at most t can do to give the players not in T a worse payoff, even if the players in T share their type information.

The notion of t-immunity and k-resilience address different concerns. For t-immunity, we consider the payoffs of the players not in K; for k-resilience, we consider the payoffs of players in K. It is natural to combine both notions. Given a strategy profile $\vec{\tau}$, let $\Gamma^T_{\vec{\tau}}$ be the game which is identical to Γ except that the players in T are fixed to playing strategy $\vec{\tau}_T$.

Definition 12. *$\vec{\sigma}$ is a* (strongly) (k,t)-robust *equilibrium in a game Γ if $\vec{\sigma}$ is t-immune and, for all subsets T of players with $|T| \leq t$ and all strategy profiles $\vec{\tau}$, $(\vec{\sigma}_{-T}, \vec{\tau}_T)$ is a (strongly) k-resilient equilibrium of $\Gamma^T_{\vec{\tau}}$.*

Note that a (strongly) k-resilient equilibrium is a (strongly) $(k,0)$-robust equilibrium. The notion of $(0,t)$ resilience is somewhat in the spirit of Eliaz's [17] notion of t fault-tolerant implementation, although Eliaz does not require t-immunity.

We can define "approximate" notions of t-immunity and (k,t)-robustness analogous to Definition 10:

Definition 13. *For $\epsilon > 0$, a strategy profile $\vec{\sigma}$ is* ϵ-t-immune *in Γ if, for all subsets T of players with $|T| \leq t$, all strategy profiles $\vec{\tau}$, all $i \notin T$, all type profiles $\vec{x} \in \mathcal{T}$, and all strategies σ_e of the environment, we have $u_i(\Gamma, (\vec{\sigma}_{-T}, \vec{\tau}_T), \sigma_e, \vec{x}_T) > u_i(\Gamma, \vec{\sigma}, \sigma_e, \vec{x}_T) - \epsilon$.*

For ϵ-(k,t)-robustness, we replace t-immunity and (strong) k-resilience for ϵ-t-immunity and (strong) ϵ-k-resilience, respectively:

Definition 14. *For $\epsilon \geq 0$, $\vec{\sigma}$ is a* (strongly) ϵ-(k,t)-robust *equilibrium in Γ if $\vec{\sigma}$ is ϵ-t-immune and, for all subsets T of players with $|T| \leq t$ and strategy profiles $\vec{\tau}_T$, $(\vec{\sigma}_{-T}, \vec{\tau}_T)$ is a (strongly) ϵ-k-resilient equilibrium of $\Gamma^T_{\vec{\tau}}$.*

4.3 Main Results

In this section, we state our results formally. We begin with the upper bounds for implementing mediator games in asynchronous setting, which are analogues of the upper bounds given by ADGH [1] in the synchronous setting. We later also state and give an alternative proof to some of their results, since some of the techniques used in the asynchronous case can be generalized deal with the synchronous case as well.

We say that a game Γ' is a utility variant of a game Γ if Γ' and Γ have the same game tree, but the utilities of the players may be different in Γ and Γ'. We use the notation $\Gamma(\vec{u})$ if we want to emphasize that $\vec{u}$ is the utility function in game Γ. We then take $\Gamma(\vec{u}')$ to be the utility variant of Γ with utility function $\vec{u}'$. We also say that that a strategy $\vec{\sigma}$ implements another strategy $\vec{\sigma}'$ if $\vec{\sigma}$ 0-bisimulates $\vec{\sigma}'$, which means that, for all inputs, the possible distribution of outcomes in $\vec{\sigma}$

and $\vec{\sigma}'$ are identical if no player deviates (note that the distributions depend on the scheduler). If $\vec{\sigma}$ only satisfies part (b) of Definition 3 we say that $\vec{\sigma}$ *weakly implements* $\vec{\sigma}'$.

One more technical comment before stating the theorems: in the mediator game we can also view the mediator as a player (albeit one without a utility function) that is following a strategy. Thus, when we talk about a strategy profile that is a (k,t)-robust equilibrium in the mediator game, we must give the mediator's strategy as well as the players' strategies. We sometimes write $\vec{\sigma} + \sigma_d$ if we want to distinguish the players' strategy profile $\vec{\sigma}$ from the mediator's strategy σ_d. We occasionally abuse notation and drop the σ_d if it is clear from context, and just talk about $\vec{\sigma}$ being a (k,t)-robust equilibrium.

Theorem 6. *If Γ is a normal-form Bayesian game with n players, $\vec{\sigma} + \sigma_d$ is a strategy profile for the players and the mediator in an asynchronous mediator game Γ_d that extends Γ, and $n > 4k + 4t$, then with both the default-move approach and the AH approach, there exists a strategy profile $\vec{\sigma}_{CT}$ that implements $\vec{\sigma} + \sigma_d$ in the asynchronous cheap-talk game Γ_{CT} such that for all utility variants $\Gamma_d(\vec{u}')$ of Γ_d, if $\vec{\sigma} + \sigma_d$ is a (strongly) (k,t)-robust equilibrium in $\Gamma_d(\vec{u}')$, then $\vec{\sigma}_{CT}$ is a (strongly) (k,t)-robust equilibrium in $\Gamma_{CT}(\vec{u}')$. If σ_d is responsive, the number of messages sent in a history of $\vec{\sigma}_{CT}$ is polynomial in n and N, linear in c, and independent of $\vec{u}'$.*

Theorem 7. *If Γ is a normal-form Bayesian game with n players, $\vec{\sigma} + \sigma_d$ is a strategy profile for the players and mediator in an asynchronous mediator game Γ_d that extends Γ, $M > 0$, and $n > 3k + 3t$, then with both the default-move approach and the AH approach, for all $\epsilon > 0$, there exists a strategy profile $\vec{\sigma}_{CT}$ in the asynchronous cheap-talk game Γ_{CT} that ϵ-implements $\vec{\sigma}$ such that for all utility variants $\Gamma_d(\vec{u}')$ of Γ_d bounded by $M/2$ (i.e., where the range of u_i' is contained in*

$[-M/2, M/2]$*), if* $\vec{\sigma} + \sigma_d$ *is a (strongly)* (k,t)*-robust equilibrium in* $\Gamma_d(\vec{u}')$*, then* $\vec{\sigma}_{CT}$ *is a (strongly)* ϵ*-*(k,t)*-robust equilibrium in* $\Gamma_{CT}(\vec{u}')$*. If* σ_d *is responsive, the number of messages sent in a history of* $\vec{\sigma}_{CT}$ *is polynomial in* n *and* N*, linear in* c*, and independent of* $\vec{u}'$*.*

If we have a punishment strategy and utilities are known, we can do better with the AH approach. To make this precise, we need the definition of an m-punishment strategy [1] (which generalizes the notion of punishment strategy defined by Ben Porath [13]). Before defining this carefully, note that in an asynchronous setting (i.e., in the mediator game and the cheap-talk game, but not in the underlying game), the utility of players depends on the environment's strategy as well as the players' strategy profile and the players' type profile.

Definition 15. *If* Γ' *is an extension of an underlying game* Γ*, a strategy profile* $\vec{\rho}$ *in* Γ *is a* k-punishment strategy with respect to a strategy profile $\vec{\sigma}'$ in Γ' *if for all subsets* K *of players with* $1 \leq |K| \leq k$*, all strategy profiles* $\vec{\sigma}$ *in* Γ*, all strategies* σ_e *for the environment, all type profiles* $\vec{x} \in \mathcal{T}$*, and all players* $i \in K$*, we have*

$$u_i(\Gamma', \vec{\sigma}', \sigma_e, \vec{x}_K) > u_i(\Gamma, (\vec{\sigma}_K, \vec{\rho}_{-K}), \vec{x}_K).$$

Thus, if $\vec{\rho}$ is a k-punishment strategy with respect to $\vec{\sigma}'$ and all but k players play their part of $\vec{\rho}$ in the underlying tame, then all of the remaining players will be worse off than they would be in Γ' if everyone had played $\vec{\sigma}'$, no matter what they do in the underlying game. For the following results we need an additional definition, we say that a protocol has bounded randomization if there exists a uniform bound N on the number of random bits that each of the players can use during the execution of the protocol.

Theorem 8. *If* Γ *is a normal-form Bayesian game with* n *players,* $\vec{\sigma} + \sigma_d$ *is a strategy profile with bounded randomization and in canonical form for the players*

and mediator in an asynchronous mediator game Γ_d that extends Γ, there is a $(k+t)$-punishment strategy with respect to $\vec{\sigma}+\sigma_d$, and $n > 3k+4t$, then with the AH approach, there exists a strategy profile $\vec{\sigma}_{CT}$ that implements $\vec{\sigma}+\sigma_d$ in the asynchronous cheap-talk game Γ_{CT}, and if $\vec{\sigma}+\sigma_d$ is a (strongly) (k,t)-robust equilibrium in Γ_d, then $\vec{\sigma}_{CT}$ is a (strongly) (k,t)-robust equilibrium in Γ_{CT}. If we require only that $\vec{\sigma}_{CT}$ is a weak implementation, then the number of messages in a history of $\vec{\sigma}_{CT}$ is polynomial in n and linear in c.

Note that in Theorem 8, the running time of the algorithm is significantly affected by whether we want $\vec{\sigma}_{CT}$ to implement $\vec{\sigma}$ or whether a weak implementation suffices.

If we assume both that there is a $(2k+2t)$-punishment strategy and that utilities are known, we can get an analogue to R2, but with an ϵ error.

Theorem 9. *If Γ is a normal-form Bayesian game with n players, $\vec{\sigma}+\sigma_d$ is a strategy profile with bounded randomization and in canonical form for the players and mediator in an asynchronous mediator game Γ_d that extends Γ, there is a $(2k+2t)$-punishment strategy with respect to $\vec{\sigma}+\sigma_d$, and $n > 2k+3t$, then with the AH approach, for all $\epsilon > 0$ there is a strategy $\vec{\sigma}_{CT}$ that ϵ-implements $\vec{\sigma}$ in the asynchronous cheap-talk game Γ_{CT} such that if $\vec{\sigma}+\sigma_d$ is a (strongly) (k,t)-robust equilibrium in Γ_d, then $\vec{\sigma}_{CT}$ is a (strongly) ϵ-(k,t)-robust equilibrium in Γ_{CT}. If we require only that $\vec{\sigma}_{CT}$ is a weak implementation, then the number of messages in a history of $\vec{\sigma}_{CT}$ is polynomial in n and linear in c.*

In the synchronous case we get analogous results to Theorems 6 and 8, except that the number of required players necessary to implement (k,t)-robust strategies with a mediator decreases by a factor of $(k+t)$.

Theorem 10. *If Γ is a normal-form Bayesian game with n players, $\vec{\sigma} + \sigma_d$ is a strategy profile for the players and the mediator in a synchronous mediator game Γ_d that extends Γ, and $n > 3k + 3t$, then there exists a strategy profile $\vec{\sigma}_{CT}$ that implements $\vec{\sigma} + \sigma_d$ in the synchronous cheap-talk game Γ_{CT} such that for all utility variants $\Gamma_d(\vec{u}')$ of Γ_d, if $\vec{\sigma} + \sigma_d$ is a (strongly) (k,t)-robust equilibrium in $\Gamma_d(\vec{u}')$, then $\vec{\sigma}_{CT}$ is a (strongly) (k,t)-robust equilibrium in $\Gamma_{CT}(\vec{u}')$.*

Theorem 11. *If Γ is a normal-form Bayesian game with n players, $\vec{\sigma} + \sigma_d$ is a strategy profile in canonical form for the players and mediator in a synchronous mediator game Γ_d that extends Γ, there is a $(k+t)$-punishment strategy with respect to $\vec{\sigma}+\sigma_d$, and $n > 2k+3t$, then with the AH approach, there exists a strategy profile $\vec{\sigma}_{CT}$ that implements $\vec{\sigma}+\sigma_d$ in the synchronous cheap-talk game Γ_{CT}, and if $\vec{\sigma}+\sigma_d$ is a (strongly) (k,t)-robust equilibrium in Γ_d, then $\vec{\sigma}_{CT}$ is a (strongly) (k,t)-robust equilibrium in Γ_{CT}.*

4.4 Proofs (Asynchronous Case)

4.4.1 Coordination between the environment and malicious players

Before proving the main results, it is useful to understand some of the implication of (k,t)-robustness, particularly when it comes to the interactions between the environment and the malicious and rational players. The definition of (k,t)-robustness requires that rational players have no profitable deviation, no matter what the malicious players and the environment do. It may seem *a priori* that the malicious players, the rational players, and the environment all act independently,

but in fact, as shown in Section 1.3, we can assume without loss of generality that they are all under the control of a single adversary. The fact that the environment and the malicious players can communicate allows us to prove a tighter correspondence between deviations in the cheap-talk game and deviations in the mediator game than the one given by t-bisimulation.

Proposition 6. *Given two protocols $\vec{\pi}$ and $\vec{\pi}'$ and a scheduler σ_e, if $\vec{\pi}'$ t-bisimulates $\vec{\pi}$, there exists a function H_{σ_e} from strategies to strategies such that $H_{\sigma_e}(\vec{\pi}_i) = \vec{\pi}'_i$ for all i, and for all adversaries $A = (\vec{\tau}_T, \sigma_e)$ with $|T| \leq t$, there exists a scheduler σ'_e such that for all inputs $\vec{x}$, $\vec{\pi}(\vec{x}, A)$ and $\vec{\pi}'(\vec{x}, (H_{\sigma_e}(\vec{\tau}_T), \sigma'_e))$ are identically distributed (where we extend H_{σ_e} to strategy profiles by taking $H_{\sigma_e}(\tau_1, \ldots, \tau_m) = (H_{\sigma_e}(\tau_1), \ldots, H_{\sigma_e}(\tau_m))$).*

Proof. Since $\vec{\pi}'$ t-bisimulates $\vec{\pi} + \pi_d$, for each adversary $A = (\vec{\tau}_T, \sigma_e)$, there exists an adversary $A' = (\vec{\tau}'_T, \sigma'_e)$ such that $\vec{\pi}(\vec{x}, A)$ and $\vec{\pi}'(\vec{x}, A')$ are identically distributed for all inputs $\vec{x}$. This means that, fixing σ_e, there exists a well-defined function $H^{adv}_{\sigma_e}$ from strategy profiles to adversaries such that for all subsets T with $|T| \leq t$ and all strategies $\vec{\tau}_T$ for players in T, there exists a scheduler σ'_e such that $(\vec{\pi} + \pi_d)(\vec{x}, (\vec{\tau}_T, \sigma_e))$ and $\vec{\pi}'(\vec{x}, H^{adv}_{\sigma_e}(\vec{\tau}_T))$ for all inputs $\vec{x}$. Given an adversary $A = (\vec{\tau}_T, \sigma_e)$, consider the following adversary $A' = (\vec{\tau}'_T, \sigma'_e)$:

1. The scheduler begins the game by scheduling all players in T once.
2. Each player $i \in T$ sends the description of the strategy $\vec{\tau}_i$ to the scheduler σ'_e the first time it is scheduled.
3. After receiving the strategies of the players in T, the scheduler computes $H^{adv}_{\sigma_e}(\vec{\tau}_T)$ and sends each player $i \in T$ the (description of) strategy $H^{adv}_{\sigma_e}(\vec{\tau}_T)_i$, and then switches to using $H^{adv}_{\sigma_e}(\vec{\tau}_T)_e$.

4. Each player $i \in T$ switches to the strategy sent by the scheduler after receiving it.

Consider the function H_{σ_e} that maps τ_i to τ'_i if $\tau_i \neq \pi_i$ and maps π_i to π'_i (note that H_{σ_e} is well defined, since τ'_i does not depend on the strategy of other players in T) and a scheduler σ''_e that acts like σ'_e except that at step 1, it schedules only the players i in T such that $\tau_i \neq \pi_i$. We have by construction that $\vec{\pi}(\vec{x}, A) = \vec{\pi}'(\vec{x}, H^{adv}_{\sigma_e}(\vec{\tau}_T))$ and therefore that $\vec{\pi}(\vec{x}, A) = \vec{\pi}'(\vec{x}, (H_{\sigma_e}(\vec{\tau}_T), \sigma''_e))$, as desired. □

Proposition 6 implies that we can assume without loss of generality that individual deviations in the mediator game correspond to individual deviations in the cheap-talk game and vice-versa. An analogous result holds for ϵ-t-bisimulation:

Proposition 7. *Given two protocols $\vec{\pi}$ and $\vec{\pi}'$ and a scheduler σ_e, if $\vec{\pi}'$ ϵ-t-bisimulates $\vec{\pi}$, there exists a function H_{σ_e} from strategies to strategies such that $H_{\sigma_e}(\vec{\pi}_i) = \vec{\pi}'_i$ for all i, and for all adversaries $A = (\vec{\tau}_T, \sigma_e)$ with $|T| \leq t$, there exists a scheduler σ'_e such that for all inputs A, $\vec{x}$, the distance between the distributions $\vec{\pi}(\vec{x}, A)$ and $\vec{\pi}'(\vec{x}, (H_{\sigma_e}(\vec{\tau}_T), \sigma'_e))$ is at most ϵ (where the notion of distance is that used in the definition of ϵ-t-bisimulation in Section 3.1).*

As we show next, since the scheduler can collude with malicious players, t-immune strategy profiles satisfy an even stronger condition: deviations by players in a set T with $|T| \leq t$ do not make things worse for the non-deviating players even if the environment colludes with the players in T.

Proposition 8. *If $\vec{\sigma}$ is t-immune, then for all sets T of players with $|T| \leq t$, strategies σ_e and σ'_e for the environment, strategy profiles $\vec{\tau}_T$ for the players in T, input profiles $\vec{x}$ and $\vec{x}'$, and players $i \notin T$, we have*

$$u_i(\Gamma_d, (\vec{\sigma}_{-T}, \vec{\tau}_T), \sigma'_e, \vec{x}'_T) \geq u_i(\Gamma_d, \vec{\sigma}, \sigma_e, \vec{x}_T). \tag{4.1}$$

Proof. Clearly, if (4.1) holds for all σ_e, σ'_e, $\vec{x}$, and $\vec{x}'$, then $\vec{\sigma}$ is t-immune. For the converse, suppose by way of contradiction that $\vec{\sigma}$ is t-immunte but for some T, $\vec{\tau}$, σ_e, σ'_e, and $i \notin T$, (4.1) does not hold. Consider a scheduler σ''_e that acts just like σ_e, except that if some player i sends a message to itself it acts like σ'_e. Then players in T can effectively decrease i's payoff with scheduler σ''_e by sending a message to themselves and playing as if they had input $\vec{x}'_T$; that is, there is a strategy $\vec{\tau}''_T$ such that

$$\begin{aligned} & u_i(\Gamma_d, (\vec{\sigma}_{-T}, \vec{\tau}''_T), \sigma''_e, \vec{x}_T) \\ = \; & u_i(\Gamma_d, (\vec{\sigma}_{-T}, \vec{\tau}_T), \sigma'_e, \vec{x}'_T) \\ < \; & u_i(\Gamma_d, \vec{\sigma}, \sigma_e, \vec{x}_T) \\ = \; & u_i(\Gamma_d, \vec{\sigma}, \sigma''_e, \vec{x}_T), \end{aligned}$$

contradicting the assumption that $\vec{\sigma}$ is t-immune. □

A similar argument shows that (k,t)-robust strategy profiles satisfy a correspondingly stronger condition, made precise in the following proposition:

Proposition 9. *A strategy profile $\vec{\sigma}$ is (k,t)-robust (resp., strongly (k,t)-robust) if and only if it is t-immune and for all disjoint sets K and T with $1 \leq |K| \leq k$ and $|T| \leq t$, all strategy profiles $\vec{\tau}_K$, $\vec{\tau}_T$, and $\vec{\tau}'_T$ for the players in K and T, respectively, all environment strategies σ_e and σ'_e, and all input profiles $\vec{x}$ and $\vec{x}'$, we have that*

$$\begin{aligned} & u_i(\Gamma_d, (\vec{\sigma}_{-(K \cup T)}, \vec{\tau}_K, \vec{\tau}''_T), \sigma'_e, \vec{x}'_{(K \cup T)}) \\ \leq \; & u_i(\Gamma_d, (\vec{\sigma}_{-(K \cup T)}, \vec{\tau}_K, \vec{\tau}'_T), \sigma'_e, \vec{x}'_{(K \cup T)}) \end{aligned} \tag{4.2}$$

for some $i \in K$ (resp., for all $i \in K$).

Proof. Again, it is clear that if (4.2) holds for all K and T with $1 \leq |K| \leq k$ and $|T| \leq t$, all $\vec{\tau}_K$, $\vec{\tau}_T$, $\vec{\tau}'_T$, $\vec{x}$, and $\vec{x}'$, and some (resp., all) $i \in K$, then $\vec{\sigma}$ is (k,t)-robust (resp., strongly (k,t)-robust).

For the converse, assume by way of contradiction that $\vec{\sigma}$ is (k,t)-robust, but for some disjoint sets K and T with $1 \leq |K| \leq k$ and $|T| \leq t$, $\vec{\tau}_K$, $\vec{\tau}_T$, $\vec{\tau}'_T$, $\vec{x}$, and $\vec{x}'$, and all $i \in K$, (4.2) does not hold. Again, we use the fact that the rational players can effectively communicate with malicious players and with the scheduler. Consider a scheduler σ''_e that acts like σ_e unless some player sends a message to itself, in which case it acts like σ'_e, and a strategy profile $\vec{\tau}''_T$ in which each player $i \in T$ acts as if it was using strategy $(\tau_T)_i$, except that it switches to $(\tau'_T)_i$ and acts as if it has input x'_i if it receives a message from a rational player (i.e., a player in K) asking it to do so. Then, given input profile $\vec{x}$, strategy profile $\vec{\tau}''_T$ for T, and scheduler σ''_e, player i can gain by sending a message to itself and sending a message to players in T asking them to follow $\vec{\tau}'_T$ and to act is if they have input $\vec{x}'_T$, and by having players in K play $\vec{\tau}_K$ as if they had input $\vec{x}'_K$, rather than playing $\vec{\sigma}$. This contradicts the assumption that $\vec{\sigma}$ is (k,t)-robust. The argument for strong (k,t)-robustness is analogous. □

Another property interesting in its own right that follows from this argument is that (k,t)-robust strategies must be *scheduler-proof*: the expected payoff for all players is the same regardless of the scheduler:

Corollary 1. *If $\vec{\sigma}$ is (k,t)-robust for some $k \geq 1$, then for all sets T with $|T| \leq t$, strategy profiles $\vec{\tau}_T$ for the players in T, environment strategies σ_e and σ'_e, input profiles $\vec{x}$, and players $i \notin T$, we have $u_i(\Gamma, (\vec{\sigma}_{-T}, \vec{\tau}_T), \sigma_e, \vec{x}_T) = u_i(\Gamma, (\vec{\sigma}_{-T}, \vec{\tau}_T), \sigma'_e, \vec{x}_T)$.*

We have analogous strengthenings of ϵ-t-immunity and ϵ-(k,t)-robustness, which are stated next. The proofs are essentially identical to that of Proposition 8 and 9 respectively, so we omit them here.

Proposition 10. *If $\epsilon > 0$ and $\vec{\sigma}$ is ϵ-t-immune in game Γ, then for all sets T of players with $|T| \leq t$, strategy profiles τ_T for the players in T, environment*

strategies σ_e and σ'_e, input profiles $\vec{x}$ and $\vec{x}'$, and players $i \notin T$, we have that

$$u_i(\Gamma, (\vec{\sigma}_{-T}, \vec{\tau}_T), \sigma'_e, \vec{x}'_T) > u_i(\Gamma, \vec{\sigma}, \sigma_e, \vec{x}_T) - \epsilon.$$

Proposition 11. *A strategy profile $\vec{\sigma}$ is ϵ-(k,t)-robust (resp., strongly ϵ-(k,t)-robust) in game Γ if and only if it is ϵ-t-immune and, for all disjoint sets K and T of players with $1 \leq |K| \leq k$ and $|T| \leq t$, all strategy profiles $\vec{\tau}_K$, $\vec{\tau}'_T$, and $\vec{\tau}_T$ for players in K and T, respectively, all environment strategies σ_e and σ'_e, and all input profiles $\vec{x}$ and $\vec{x}'$, we have that*

$$u_i(\Gamma, (\vec{\sigma}_{-(K\cup T)}, \vec{\tau}_K, \vec{\tau}'_T), \sigma'_e, \vec{x}'_{(K\cup T)}) < u_i(\Gamma, (\vec{\sigma}_{-T}, \vec{\tau}_T), \sigma_e, \vec{x}_T) + \epsilon$$

for some $i \in K$ (resp., for all $i \in K$).

It will be useful for our later results that we can actually improve on the bound of ϵ in Propositions 10 and 11.

Proposition 12. *If $\vec{\sigma}$ is an ϵ-t-immune strategy in a finite game Γ, then there exists ϵ_0 with $0 < \epsilon_0 < \epsilon$ such that for all sets of players T with $|T| \leq t$, strategy profiles $\vec{\tau}_T$ for the players in T, environment strategies σ_e and σ'_e, input profiles $\vec{x}$ and $\vec{x}'$, and players $i \notin T$, we have that*

$$u_i(\Gamma, (\vec{\sigma}_{-T}, \vec{\tau}_T), \sigma'_e, \vec{x}'_T) > u_i(\Gamma, \vec{\sigma}, \sigma_e, \vec{x}_T) - \epsilon_0.$$

Proof. Since, by Proposition 10, for each choice of $\vec{\tau}_T$, σ_e, and σ'_e, we have

$$u_i(\Gamma, \vec{\sigma}, \sigma_e, \vec{x}_T) - u_i(\Gamma, (\vec{\sigma}_{-T}, \vec{\tau}_T), \sigma'_e, \vec{x}'_T) < \epsilon,$$

and the space of player strategy profiles, environment strategies, and input value profiles is compact, if we take the sup of the left-hand side over all choices of strategy profiles $\vec{\tau}_T$, environment strategies σ_e and σ'_e, and input profiles $\vec{x}$ and $\vec{x}'$, it takes on some maximum value $\epsilon_1 < \epsilon_0$. We can then take $\epsilon_0 = (\epsilon + \epsilon_1)/2$. □

Using Proposition 11, we get a similar result for ϵ-(k,t)-robustness. The proof is analogous to that of Proposition 12.

Proposition 13. *If Γ is a finite game and $\vec{\sigma}$ is a ϵ-(k,t)-robust strategy (resp., strongly ϵ- (k,t)-robust strategy) in Γ_d, then there exists ϵ_0 with $0 < \epsilon_0 < \epsilon$ such that for all disjoint sets K and T of players with $1 \leq |K| \leq k$ and $|T| \leq t$, all strategy profiles $\vec{\tau}_K$, $\vec{\tau}_T$ and $\vec{\tau}'_T$ for players in K and T, respectively, all environment strategies σ_e and σ'_e, and all input profiles $\vec{x}$ and $\vec{x}'$, we have that*

$$u_i(\Gamma, (\vec{\sigma}_{-(K\cup T)}, \vec{\tau}_K, \vec{\tau}_T), \sigma_e, \vec{x}'_{(K\cup T)}) < u_i(\Gamma, (\vec{\sigma}_{-T}, \vec{\tau}'_T), \sigma'_e, \vec{x}_T) + \epsilon_0$$

for some $i \in K$ (resp., all $i \in K$).

4.4.2 Constructing a protocol that t-coterminates

If $t < n/3$ we can get an an analogue of Theorems 3 and 4 by replacing $(t, 2t+1)$-cotermination by t-cotermination and ϵ-$(t, t+1)$-cotermination by ϵ-t-cotermination respectively. We sketch the construction for t-cotermination; an analogous construction achieves ϵ-t-cotermination.

Consider a protocol $\vec{\sigma}'_{CT}$ in which players play just as in $\vec{\sigma}_{CT}$ except that, whenever an honest player i terminates in $\vec{\sigma}_{CT}$, it instead broadcasts an 'OK' message to all players and waits until it receives $3t+1$ 'OK' messages before it terminates in $\vec{\sigma}'_{CT}$. Note that if an honest player terminates in $\vec{\sigma}'_{CT}$, then at least $3t+1$ players must have broadcast an 'OK' message in this history of $\vec{\sigma}'_{CT}$, of which at least $2t+1$ are honest. Thus, at least $2t+1$ players terminate in the corresponding history of $\vec{\sigma}_{CT}$. Since $\vec{\sigma}_{CT}$ $(t, 2t+1)$-coterminates, it follows that all players not in T must terminate with $\vec{\sigma}_{CT}$, and hence all players not in T send an 'OK' message (and terminate) with $\vec{\sigma}'_{CT}$.

It remains to show that $\vec{\sigma}'_{CT}$ still (t,t')-bisimulates $\vec{\sigma}+\sigma_d$ if $3t+t'<n$. Clearly, it still relaxed t-bisimulates $\vec{\sigma}+\sigma_d$, so we just have to show that all players are guaranteed to terminate in the presence of at most t' malicious players. However, in this case, by assumption, all honest players are guaranteed to terminate in $\vec{\sigma}_{CT}$, and thus all honest players are guaranteed to eventually send an 'OK' broadcast in $\vec{\sigma}'_{CT}$. Since $n-t'>3t+1$, this guarantees that there will be at least $3t+1$ 'OK' broadcasts and all honest players will eventually terminate, as desired.

Note that this construction requires players a reliable broadcast protocol, and thus can be done only if $n>3(t+k)$. To prove Theorem 9 we require different techniques.

4.4.3 Proof of Theorem 6

By Theorem 3, if $n>4k+4t$, there exists a strategy profile $\vec{\sigma}_{CT}$ that $(k+t)$-bisimulates $\vec{\sigma}+\sigma_d$. It is immediate from the definition of $(k+t)$-bisimulation that $\vec{\sigma}_{CT}$ implements $\vec{\sigma}+\sigma_d$. Since the probability of deadlock is 0, what the players do in case of deadlock is irrelevant, so this approach works both in the case of the AH approach and the default-move approach. It remains to show that, for each utility variant $\Gamma_d(\vec{u}')$ of Γ_d, if $\vec{\sigma}+\sigma_d$ is a (strongly) (k,t)-robust equilibrium in $\Gamma_d(\vec{u}')$, then $\vec{\sigma}_{CT}$ is a (strongly) (k,t)-robust equilibrium in $\Gamma_{CT}(\vec{u}')$. We start by showing that $\vec{\sigma}_{CT}$ is t-resilient in $\Gamma_{CT}(\vec{u}')$.

Given T with $|T|\leq t$, $\vec{\tau}_T$, and σ_e, by Theorem 3 and Proposition 6, there exists a function H_{σ_e} from strategies to strategies and a scheduler σ'_e such that for all

input profiles $\vec{x}$,

$$\begin{aligned} & u_i'(\Gamma_{CT}(\vec{u}'), ((\vec{\sigma}_{CT})_{-T}, \tau_T), \sigma_e, \vec{x}) \\ = \; & u_i'(\Gamma_d(\vec{u}'), (\vec{\sigma}_{-T}, H_{\sigma_e}(\vec{\tau}_T)), \sigma_e', \vec{x}) \end{aligned}$$

for all players i. There also exists a scheduler σ_e'' such that

$$u_i'(\Gamma_{CT}(\vec{u}'), \vec{\sigma}_{CT}, \sigma_e', \vec{x}) = u_i'(\Gamma_d(\vec{u}'), \vec{\sigma}, \sigma_e'', \vec{x}).$$

Since $\vec{\sigma}$ is t-immune, for all $i \notin T$ we have that

$$\begin{aligned} & u_i'(\Gamma_{CT}(\vec{u}'), ((\vec{\sigma}_{CT})_{-T}, \vec{\tau}_T), \sigma_e, \vec{x}_T) \\ = \; & u_i'(\Gamma_d(\vec{u}'), (\vec{\sigma}_{-T}, H_{\sigma_e}(\vec{\tau}_T)), \sigma_e', \vec{x}_T) \\ \geq \; & u_i'(\Gamma_d(\vec{u}'), \vec{\sigma}, \sigma_e'', \vec{x}_T) && \text{[by Lemma 8]} \\ = \; & u_i'(\Gamma_{CT}(\vec{u}'), \vec{\sigma}_{CT}, \sigma_e', \vec{x}_T). \end{aligned}$$

Therefore, $\vec{\sigma}_{CT}$ is t-immune.

To show (strong) (k,t)-robustness, taking $\vec{\tau}_T$, σ_e, and σ_e' as above, suppose that K is a set of players disjoint from T such that $|K| \leq k$, and the players in K play $\vec{\tau}_K$. By Theorem 3 and Proposition 6, there exists σ_e^* and H_{σ_e} such that

$$\begin{aligned} & u_i'(\Gamma_{CT}(\vec{u}'), ((\vec{\sigma}_{CT})_{-(K \cup T)}, \vec{\tau}_K, \vec{\tau}_T), \sigma_e, \vec{x}) \\ = \; & u_i'(\Gamma_d(\vec{u}'), (\vec{\sigma}_{-(K \cup T)}, H_{\sigma_e}(\vec{\tau}_K), H_{\sigma_e}(\vec{\tau}_T)), \sigma_e^*, \vec{x}_{(K \cup T)}) \end{aligned}$$

for all players i.

By Corollary 9, if $\vec{\sigma} + \sigma_d$ is (strongly) (k,t)-robust in $\Gamma_d(\vec{u}')$, then

$$\begin{aligned} & u_i'(\Gamma_d(\vec{u}'), (\vec{\sigma}_{-(K \cup T)}, H_{\sigma_e}(\vec{\tau}_K), H_{\sigma_e}(\vec{\tau}_T)), \sigma_e^*, \vec{x}_{(K \cup T)}) \\ \leq \; & u_i'(\Gamma_d(\vec{u}'), (\vec{\sigma}_{-T}, H_{\sigma_e}(\vec{\tau}_T)), \sigma_e', \vec{x}_T) \end{aligned}$$

for some (resp., all) $i \in K$. For those $i \in K$ for which this inequality holds, we

have

$$\begin{aligned} & u_i'(\Gamma_{CT}(\vec{u}'), ((\vec{\sigma}_{CT})_{-(K\cup T)}, \tau_K, \tau_T), \sigma_e, \vec{x}_{(K\cup T)}) \\ = \; & u_i'(\Gamma_d(\vec{u}'), (\vec{\sigma}_{-(K\cup T)}, H_{\sigma_e}(\vec{\tau}_K), H_{\sigma_e}(\vec{\tau}_T)), \sigma_e^*, \vec{x}_{(K\cup T)}) \\ \leq \; & u_i'(\Gamma_d(\vec{u}'), (\vec{\sigma}_{-T}, H_{\sigma_e}(\vec{\tau}_T)), \sigma_e', \vec{x}_T) \\ = \; & u_i'(\Gamma_{CT}(\vec{u}'), ((\vec{\sigma}_{CT})_{-T}, \sigma_e, \vec{x}_T). \end{aligned}$$

It follows that $\vec{\sigma}_{CT}$ is (strongly) (k,t)-robust in $\Gamma_{CT}(\vec{u}')$.

4.4.4 Proof of Theorem 7

The proof of Theorem 7 is essentially the same as that of Theorem 6, except that we now use Theorem 4 instead of Theorem 3. By Theorem 4, for all $\epsilon' \in (0,1]$, there exists a protocol $\vec{\sigma}_{CT}$ that ϵ-$(t+k)$-bisimulates $\vec{\sigma}$ and the expected number of messages sent is polynomial in n and N, and linear in c. It follows that $\vec{\sigma}_{CT}$ ϵ'-implements $\vec{\sigma}$ and has at most if probability ϵ' of deadlock. We next show that the can make ϵ' sufficiently small so that the question of whether we use AH approach or the default-move approach becomes irrelevant.

We now prove ϵ-(k,t)-robustness. Suppose that $\vec{\sigma} + \sigma_d$ is a (strongly) ϵ-(k,t)-robust equilibrium in the utility variant $\Gamma_d(\vec{u}')$ of Γ_d. We show that $\vec{\sigma}_{CT}$ is ϵ-t-immune in $\Gamma_{CT}(\vec{u}')$.

Since $\vec{\sigma}_{CT}$ ϵ-$(t+k)$-bisimulates $\vec{\sigma}$, for all inputs $\vec{x}$ we can associate histories in the mediator game and histories in the cheap-talk game in such a way that the set of histories where the outcomes differ has probability at most ϵ'. Since all utilities are in the range $[-M/2, M/2]$, by assumption, the maximum difference in utility between two outcomes in the underlying game is M. Thus, by Proposition 7, there exists an environment strategy σ_e' and a function H_{σ_e} from strategies to strategies

such that for all input profiles $\vec{x}$, we have

$$\begin{aligned} & u_i'(\Gamma_{CT}(\vec{u}'), ((\vec{\sigma}_{CT})_{-T}, \vec{\tau}_T), \sigma_e', \vec{x}) \\ > \; & u_i'(\Gamma_d(\vec{u}'), (\vec{\sigma}_{-T}, H_{\sigma_e}(\vec{\tau}_T)), \sigma_e, \vec{x}_T) - \epsilon' M \end{aligned}$$

for all $i \notin T$. Theorem 4 also guarantees that there exists an environment strategy σ_e'' such that

$$u_i'(\Gamma_{CT}(\vec{u}'), \vec{\sigma}_{CT}, \sigma_e', \vec{x}_T) < u_i'(\Gamma_d(\vec{u}'), \vec{\sigma}, \sigma_e'', \vec{x}_T) + \epsilon' M.$$

Since $\vec{\sigma}$ is ϵ-t immune in $\Gamma_d(\vec{u}')$, by Proposition 12, there exists a value ϵ_0 with $0 < \epsilon_0 < \epsilon$ such that

$$\begin{aligned} \vec{\tau}_T'), \quad & u_i'(\Gamma_{CT}(\vec{u}'), ((\vec{\sigma}_{CT})_{-T}, \vec{\tau}_T'), \sigma_e', \vec{x}_T) \\ > \quad & u_i'(\Gamma_d(\vec{u}'), (\vec{\sigma}_{-T}, H_{\sigma_e}(\vec{\tau}_T)), \sigma_e, \vec{x}_T) - \epsilon' M \\ > \quad & u_i'(\Gamma_d(\vec{u}'), \vec{\sigma}, \sigma_e'', \vec{x}_T) - \epsilon_0 - \epsilon' M \\ > \quad & u_i'(\Gamma_{CT}(\vec{u}'), \vec{\sigma}_{CT}, \sigma_e', \vec{x}_T) - \epsilon_0 - 2\epsilon' M. \end{aligned}$$

If we take $\epsilon' = (\epsilon - \epsilon_0)/2M$, this shows that $\vec{\sigma}_{CT}$ is (ϵ, t)-immune with both the AH approach and the default-move approach.

To show (strong) ϵ-(k,t)-robustness, keeping T, $\vec{\tau}_T$, H_{σ_e}, σ_e, and σ_e' as above, for all sets K of players disjoint from T with $1 \leq |K| < k$ and strategy profiles $\vec{\tau}_K$, there exists an environment strategy σ_e^* and a value ϵ_0 with $0 < \epsilon_0 < \epsilon$ such that for all input profiles $\vec{x}$, if $\vec{\sigma} + \sigma_d$ is (k,t)-robust (resp. strongly (k,t)-robust), then

$$\begin{aligned} & u_i'(\Gamma_{CT}(\vec{u}'), ((\vec{\sigma}_{CT})_{-(K \cup T)}), \vec{\tau}_K, \vec{\tau}_T), \sigma_e', \vec{x}_{(K \cup T)}) \\ < \; & u_i'(\Gamma_d(\vec{u}'), (\vec{\sigma}_{-(K \cup T)}, H_{\sigma_e}(\vec{\tau}_K), H_{\sigma_e}(\vec{\tau}_T)), \sigma_e, \vec{x}_{(K \cup T)}) + \epsilon' M \\ < \; & u_i'(\Gamma_d(\vec{u}'), (\vec{\sigma}_{-T}, H_{\sigma_e}(\vec{\tau}_T)), \sigma_e'', \vec{x}_T) + \epsilon_0 + \epsilon' M \qquad \text{[by Proposition 13]} \\ < \; & u_i'(\Gamma_{CT}(\vec{u}'), (\vec{\sigma}_{CT})_{-T}, \vec{\tau}), \sigma_e, \vec{x}_T) + \epsilon_0 + 2\epsilon' M \qquad \text{[by Theorem 4].} \end{aligned}$$

for some (resp., for all) $i \in K$. This shows that if we take $\epsilon' := (\epsilon - \epsilon_0)/2M$, then $\vec{\sigma}_{CT}$ is ϵ-(k,t)-robust (resp., strongly (k,t)-robust). Note that this argument works for both the AH approach and the default-move approach since it does not depend on the actions played in deadlock.

4.4.5 Proof of Theorem 8

The proof of Theorem 8 is similar in spirit to that of Theorem 6. The main problem we have to deal with is that of ensuring that rational players participate. To force participation, we have the honest players put the punishment strategy in their "wills", so that if $\vec{\sigma}_{CT}$ ends in deadlock, the rational players will be punished. By the arguments given in Section 4.4.2, we can assume without loss of generality in this proof that the implementation given by Theorems 4 $(t+k)$-coterminates, and thus either all honest players terminate or they all play the punishment strategy. Unfortunately, a naive implementation of this approach does not work, as the following example shows.

Consider an underlying game Γ for $n > 3k$ players where the set of actions is $A := \{0, 1, \perp\}$. If at least $k+1$ players play $\perp$, all players get a payoff of 1.1; if k or fewer players play $\perp$ and all players play either 0 or $\perp$, then all players get a payoff of 1; if k or fewer players play $\perp$ and all players play either $\perp$ or 1, then all players get a payoff of 2; otherwise, all players get 0. Let Γ_d be an extension of Γ with a mediator. Suppose that the mediator d uses the following strategy: The mediator d chooses $a, b \in \{0, 1\}$ uniformly at random. Then d sends the message $a + bi$ (mod 2) to player i (the same a and b are used in all these messages). Finally, d sends the message "output b; STOP" to all players (so the strategy is in canonical form).

Let σ_i be the strategy where player i ignores the message $a + bi$ and plays b after receiving the message "output b". It is easy to check that $\vec{\sigma}$ is a k-resilient equilibrium in the mediator game, and gives players an expected payoff of 1.5. Moreover, playing $\perp$ is a k-punishment strategy with respect to $\vec{\sigma}$, since if all but k players play $\perp$, then everyone gets a payoff of 1.1 (since at least $k + 1$ players

play $\perp$), which is less than 1.5.

The naive approach to implementing the mediator does not work for this game, at least with the punishment strategy $\perp$. For example, suppose that after receiving the messages $a + bi \pmod 2$, the rational players communicate with each other. Moreover, suppose that the set K of rational players includes i and j such that $i-j$ is odd. Then the rational players can compute b. If $b = 0$, they actually prefer their payoff with the punishment strategy to their payoff with $\vec{\sigma}_{CT}$. Thus, they will stop sending messages. The simulation will not terminate, so the punishment strategy in the players' wills will be applied, making the rational players better off. Thus, we cannot simulate the mediator with this approach. Of course, there are punishment strategies in this game that would lead to cooperation (e.g., randomizing between 0 and 1). Nevertheless, this example shows that using an arbitrary punishment strategy may not suffice to force the rational players to cooperate.

The problem here is that the mediator tells each player i what $a+bi$ is. We do not want the mediator to send such unnecessary information. But what counts as unnecessary? We make "unnecessary" precise by showing that, for each strategy profile $\vec{\sigma}+\sigma_d$ of a mediator game, we can construct a strategy $\vec{\sigma}^m+\sigma_d^m$ that implements $\vec{\sigma}+\sigma_d$ and leaks no information. More precisely, there exists a function f from strategy profiles to strategy profiles such that, for all strategy profiles $\vec{\sigma}+\sigma_d$, $f(\vec{\sigma}+\sigma_d)$ implements $\vec{\sigma}+\sigma_d$ and essentially all the mediator sends each player when playing $f(\vec{\sigma}+\sigma_d)$ is the action to play in the underlying game. (If we require only weak implementation, then this is exactly the case; for implementation, the messages can also include a round number.) Moreover, if $\vec{\sigma}+\sigma_d$ is (k,t)-robust (resp., strongly (k,t)-robust, ϵ-(k,t)-robust, strongly ϵ-(k,t)-robust), then so is $f(\vec{\sigma}+\sigma_d)$.

The intuition behind the construction of $\vec{\sigma}^m+\sigma_d^m := f(\vec{\sigma}+\sigma_d)$ is that all players

send their input to the mediator, the mediator waits until it receives messages from at least $n - k - t$ players, then simulates the game using the inputs sent by the players, and sends back to each player what they would have played in the simulation. However, to get an implementation of $\vec{\sigma} + \sigma_d$, the scheduler that the mediator uses in its simulation must depend somehow on the actual scheduler and must be chosen in such a way that, for a given input profile $\vec{x}$, all distributions in $(\vec{\sigma} + \sigma_d)(\vec{x}, \sigma_e)$ are possible.

More precisely, the construction proceeds as follows: player i uses strategy σ_i^m, according to which i sends input x_i to the mediator, waits for the mediator's message msg_i (which we take to be an action for player i), and then plays action msg_i. The mediator's strategy σ_d^m consists of waiting until the first turn ℓ at which there exists a subset S of at least $n - k - t$ players such that the mediator has received exactly one message s_i from each player $i \in S$, and s_i is a possible input of player i. What the mediator does next depends on whether $|S| = n$ or $|S| < n$.

If $|S| < n$, the mediator simulates $\vec{\sigma} + \sigma_d$ assuming that each player i in S has input s_i, and that the scheduler schedules players in S and the mediator in round-robin fashion and delivers all messages immediately after they are sent (note that such a scheduler exists even with the constraint that all players must be eventually scheduled, since the scheduler can schedule players not in S after the mediator terminates). The mediator then sends to each player i they action that i plays in its simulation. For future reference, we denote the scheduler used in this simulation by σ_e^S.

If $|S| = n$, the mediator proceeds as follows: Let Ω be the set of deterministic schedulers, and let $\mathcal{D}^{\vec{x}} := \bigcup_{\sigma_e \in \Omega} \{(\vec{\sigma} + \sigma_d)(\vec{x}, \sigma_e)\}$ Since each player uses a finite amount of randomization in $\vec{\sigma} + \sigma_d$, for all inputs $\vec{x}$, deterministic schedulers σ_e,

and action profiles $\vec{a}$, the probability that players play action profile $\vec{a}$ in the underlying game when playing $\vec{\sigma} + \sigma_d$ with input $\vec{x}$ and scheduler σ_e is a rational number. Since there are finitely many possible action profiles in the underlying game, $\mathcal{D}^{\vec{x}}$ is countable. Thus, there exists a set $\{\sigma_e^{(\vec{x},1)}, \sigma_e^{(\vec{x},2)}, \ldots\}$ of schedulers such that $\mathcal{D}^{\vec{x}} = \{(\vec{\sigma}+\sigma_d)(\vec{x}, \sigma_e^{(\vec{x},1)}), (\vec{\sigma}+\sigma_d)(\vec{x}, \sigma_e^{(\vec{x},2)}), \ldots\}$. The mediator simulates $\vec{\sigma} + \sigma_d$ assuming that each player i has input s_i and that the scheduler is $\sigma_e^{(\vec{s},\ell)}$ (recall that ℓ is the turn at which the mediator receives the required number of messages). If $\mathcal{D}^{\vec{x}}$ is finite, it performs the simulation with scheduler $\sigma_e^{(\vec{s},\ell(\bmod |\mathcal{D}^{\vec{x}})|)}$ instead. $\sigma_e^{(\vec{s},(\ell \bmod |\mathcal{D}^{\vec{x}}|))}$ instead.

Lemma 9. *$\vec{\sigma}^m + \sigma_d^m$ implements $\vec{\sigma} + \sigma_d$ and is (k,t)-robust.*

Proof. First note that it suffices to prove this result for deterministic schedulers. This follows from the fact that all randomized schedulers can be written as a (possibly infinite) linear combination of deterministic schedulers. By construction, for all inputs $\vec{x}$ and all deterministic schedulers $\vec{x}$ (under the assumption that no agents deviate). To prove the converse, given a deterministic scheduler σ_e for $\vec{\sigma} + \sigma_d$, we have that $(\vec{\sigma} + \sigma_d)(\vec{x}, \sigma_e) \in \mathcal{D}^{\vec{x}}$, and thus there exists $k \in \mathbb{N}$ such that $(\vec{\sigma} + \sigma_d)(\vec{x}, \sigma_e) = (\vec{\sigma} + \sigma_d)(\vec{x}, \sigma_e^{(\vec{x},k)})$ Consider a scheduler σ'_e in $\vec{\sigma}^m + \sigma_d^m$ that schedules all honest players consecutively, then schedules the mediator $k-1$ times, then delivers all messages sent by the players to the mediator, and then schedules the mediator again. By construction, in this scenario, the mediator simulates $\vec{\sigma} + \sigma_d$ with input profile $\vec{x}$ and scheduler $\sigma_e^{(\vec{x},k)}$. Therefore $(\vec{\sigma} + \sigma_d)(\vec{x}, \sigma_e)$ and $(\vec{\sigma}^m + \sigma_d^m)(\vec{x}, \sigma'_e)$ are identically distributed.

To see that $\vec{\sigma}^m + \sigma_d^m$ is t-immune, suppose, by way of contradiction, that it is not. Thus, there must exist an adversary $A = (T, \vec{\tau}_T, \sigma_e)$ with $|T| \leq t$ and an input profile $\vec{x}$ such that $u_i(\vec{\sigma}^m + \sigma_d^m, A, \vec{x}) < u_i(\vec{\sigma}_m + \sigma_d^m, \vec{x}, \sigma_e)$ for some $i \notin T$.

We can assume without loss of generality that A is deterministic. When playing $\vec{\sigma}^m + \sigma_d^m$ with adversary A and input profile $\vec{x}_T$, the first round m by which the mediator receives messages of the right form from a subset S of at least $n-k-t$ players, the set S and the values s_i used in the mediator's simulation are uniquely determined. Consider an adversary $A' = (T, \vec{\tau}'_T, \sigma'_e)$ in $\vec{\sigma} + \sigma_d$ where each player $i \in T$ acts as if it was an honest player with input s_i. while the scheduler acts like σ_e^S if $|S| < n$, and like σ_e^m otherwise. However, if a_i is the action that i would have played if it was honest and had input s_i, instead of playing a_i, i plays what it would have played in $\vec{\sigma}^m + \sigma_d^m$ if it had received message a_i from the mediator. By construction, for all inputs $\vec{x}$, $(\vec{\sigma} + \sigma_d)(\vec{x}, A)$ and $(\vec{\sigma}^m + \sigma_d^m)(\vec{x}, A')$ are identically distributed. This implies that

$$u_i(\vec{\sigma} + \sigma_d, A', \vec{x}) < u_i(\vec{\sigma} + \sigma_d, \vec{x}, \sigma''_e)$$

for some scheduler σ''_e, which contradicts the assumption that $\vec{\sigma} + \sigma_d$ is t-immune.

The argument that $\vec{\sigma}^m + \sigma_d^m$ is (k,t)-resilient is analogous, and left to the reader. □

Thus, without loss of generality, we can assume that the players and mediator use such a strategy profile. We call $f(\vec{\sigma} + \sigma_d)$ the *minimally-informative* strategy corresponding to $\vec{\sigma} + \sigma_d$. More generally, we say that $\vec{\sigma}^m + \sigma_d^m$ is a minimally-informative strategy if $\vec{\sigma}^m + \sigma_d^m = f(\vec{\sigma} + \sigma_d)$ for some strategy profile $\vec{\sigma} + \sigma_d$.

Since we consider only mediator games in canonical form, this guarantees termination for all honest players regardless of what the rational and malicious players do, provided that the scheduler is standard (i.e., not relaxed). However, once we allow relaxed schedulers, there is a possibility of deadlock. We assume for the purposes of the proof that we use the AH approach in the mediator game, and have the

players play the punishment punishment strategy in their wills. Since $\vec{\sigma}_{CT}$ guarantees t-cotermination for $t < n/3$, it follows that in the cheap-talk game, either all honest players terminate or all honest players play the punishment strategy. This guarantees that the players get the same payoff in corresponding histories in the mediator game and the cheap-talk game.

The next step in proving Theorem 8 is to show that rational players playing with a relaxed scheduler cannot get an expected payoff that is higher than their expected payoff when they play such a minimally-informative (k, t)-robust equilibrium strategy with a standard scheduler. Although this property does not hold in general, it does hold when a certain degree of cotermination (which is provided by Theorems 3 and 4) is guaranteed and there exists a punishment strategy. Under these conditions, the rational players do not want too many honest players to fail to terminate, because the honest players that do not terminate will play the punishment strategy.

To state this more precisely, we need to introduce a little more notation. Bisimulation guarantees that for each strategy τ_A that the adversary can play in the cheap-talk game, there exists a corresponding strategy τ'_A in the mediator game that leads to the same outcome for all players, regardless of the input. Since the strategy $\vec{\sigma}_{CT}$ provided by Theorems 3 and 4 coterminates (with the parameters of cotermination depending on the theorem), this imposes a constraint on τ'_A that is captured in the following definition:

Definition 16. *Given a strategy profile $\vec{\sigma}$, a scheduler σ_e, and a subset T of players, $(\vec{\sigma}, \sigma_e)$ T-t-*coterminates *if, for all input profiles $\vec{x}$, in every history of $(\vec{\sigma}, \sigma_e, \vec{x})$, either all players not in T terminate or less than t do. We say that $(\vec{\sigma}, \sigma_e)$ ϵ-T-t-*coterminates *if this property holds with probability $1 - \epsilon$.*

Proposition 14. *If $\epsilon > 0$, $\vec{\sigma} + \sigma_d$ is a minimally-informative ϵ-(k,t)-robust (resp., strongly ϵ-(k,t)-robust) equilibrium in a mediator game Γ_d for which a $(2k+2t)$-punishment strategy exists, σ_E is a relaxed scheduler, K and T are disjoint sets of players with $1 \leq |K| \leq k$ and $|T| \leq t$, and $\vec{\tau}_{(K \cup T)}$ is a strategy profile for the players in $K \cup T$ such that $(\vec{\sigma}_{-(K \cup T)}, \vec{\tau}_{(K \cup T)}, \sigma_E)$ ϵ-$(K \cup T)$-$(t+k+1)$-coterminates, then there exists a value $\epsilon_0 < \epsilon$ such that for all standard schedulers σ_e and all input profiles $\vec{x}$, we have that*

$$\begin{aligned} & u'_i(\Gamma_d, (\vec{\sigma}_{-(K \cup T)}, \vec{\tau}_{(K \cup T)}), \sigma_E, \vec{x}_{(K \cup T)}) \\ < \quad & u'_i(\Gamma_d, (\vec{\sigma}_{-T}, \vec{\tau}_T), \sigma_e, \vec{x}_T) + \epsilon_0 + \epsilon M \end{aligned}$$

for some (resp., for all) $i \notin T$.

To prove Proposition 14, we first show that all strategy profiles can be approximated by a profile by where there is a uniform bound on the amount of randomness used by an adversary.

Definition 17. *Given a strategy profile $\vec{\sigma}$, an adversary $A = (\vec{\tau}_T, \sigma_e)$ is* N-bounded with respect to $\vec{\sigma}$ *if, for all inputs and all histories, the number of random coin tosses performed by a player in T or the scheduler σ_e when players in T play $\vec{\tau}_T$, the remaining players play $\vec{\sigma}$, and the scheduler plays σ_e is bounded by N.*

Lemma 10. *For all strategy profiles $\vec{\sigma}$, adversaries $A = (\vec{\tau}_T, \sigma_E)$, and $\epsilon > 0$, there exists an adversary $A' = (\vec{\tau}'_T, \sigma'_E)$ and an $N > 0$ such that A' is N-bounded with respect to $\vec{\sigma}$ and, for all input profiles $\vec{x}$, the distance between the distributions $O(\vec{\sigma}, A, \vec{x})$ and $O(\vec{\sigma}, A', \vec{x})$ is at most ϵ.*

Proof. Fix $\epsilon > 0$. Let $A^N = (\vec{\tau}^N_T, \sigma^N_E)$ be the adversary that plays $(\vec{\tau}_T, \sigma_E)$, except that all players $i \in T$ and the scheduler act as if all the coin tosses after the Nth

coin toss are tails. By construction, for all input profiles $\vec{x}$, we have

$$\lim_{N\to\infty} d(O(\vec{\sigma}, A, \vec{x}), O(\vec{\sigma}, A^N, \vec{x})) = 0,$$

where $d(\cdot,\cdot)$ denotes the distance between distributions. Thus, there exists an integer $N_{\vec{x}}$ such that $d(O(\vec{\sigma}, A, \vec{x}), O(\vec{\sigma}, A^{N_{\vec{x}}}, \vec{x})) < \epsilon$. Since there are only finitely many input profiles, we can take $N = \max_{\vec{x}}(N^{\vec{x}})$ to get the desired result. □

Proof of Proposition 14. First assume that
$A := (\vec{\tau}_{(K\cup T)}, \sigma_E)$ is N-bounded with respect to $\vec{\sigma}$ for some $N > 0$. Let τ'_i be the strategy where $i \in K \cup T$ begins by tossing N random coins, it then communicates the outcome of the coin tosses and its input to the adversary (using the communication scheme described in Section 1.3). Player i then plays τ_i using the outcome of the coin tosses whenever it needs to randomize. The scheduler σ'_E acts like σ_E except that it does not deliver any message that a player $j \in (K \cup T)$ sends with τ'_j that is not also sent with τ_j. (Note that for j to commuicate its initial state and randomness to the scheduler does not actually require j to send messages to the scheduler. It just sends messages to itself, which do not have to be delivered.) By construction, we have that

$$\begin{aligned} & u'_i(\Gamma_d, (\vec{\sigma}_{-(K\cup T)}, \vec{\tau}_{(K\cup T)}), \sigma_E, \vec{x}_{(K\cup T)}) \\ = \;& u'_i(\Gamma_d, (\vec{\sigma}_{-(K\cup T)}, \vec{\tau}'_{(K\cup T)}), \sigma'_E, \vec{x}_{(K\cup T)}). \end{aligned}$$

We can view the adversary's strategy $(\vec{\tau}'_{(K\cup T)}, \sigma'_E)$ as a convex combination of (possibly infinitely many) deterministic strategies $(\vec{\tau}^*_{(K\cup T)}, \sigma^*_E)$. The construction of minimally-informative strategies guarantees that the number of honest players that terminate when running $(\Gamma_d, (\vec{\sigma}_{-(K\cup T)}, \vec{\tau}^*_{(K\cup T)}),\ \sigma^*_E, \vec{x}_{(K\cup T)})$ depends only on the scheduler and the inputs and randomization performed by players in $K \cup T$. Thus, given the input profile $\vec{x}_{(K\cup T)}$ of players in $K \cup T$, we can classify

each of the deterministic strategies $(\vec{\tau}^*_{(K\cup T)}, \sigma^*_E)$ into the following three categories depending on how many honest players terminate (which, by our constraints on relaxed schedulers, must be the same in every history of $(\vec{\sigma}_{-(K\cup T)}, \vec{\tau}^*_{K\cup T}, \sigma^*_E)$):

A. all honest players terminate;

B. $n - 2t - 2k$ or more honest players do not terminate;

C. at least one but fewer than $n - 2t - 2k$ honest players do not terminate.

Consider a scheduler σ''_e that acts just like σ'_E as long as the history is consistent with a history of $(\vec{\sigma}_{-(K\cup T)}, \vec{\tau}^*_{K\cup T}, \sigma'_E)$, until there comes a point when it is clear that some honest players will not terminate. If such a point comes, or if the history is inconsistent with a history of $(\vec{\sigma}_{-(K\cup T)}, \vec{\tau}^*_{K\cup T}$, then σ''_e delivers all undelivered messages and from then on delivers all messages immediately. in more detail, σ''_e just like σ'_E until one of the following conditions holds:

- A player in $K \cup T$ does not communicate its initial state and the outcome of N coin tosses to the scheduler.
- The scheduler σ'_e can tell that what has happened thus far is inconsistent with the information sent by the players in $K \cup T$, assuming that they are using $\vec{\tau}'_{K\cup T}$ and the remaining players are using $\vec{\sigma}_{-(K\cup T)}$.
- It follows from the information sent by the players in $K \cup T$ that, with $(\sigma_{-(K\cup T)}, \tau_{K\cup T}, \sigma^*_E)$, some honest player will not terminate.
- All honest players terminate.

Note that, by construction, one of these one of these conditions must hold in every history. For if the mediator and honest players play a minimally-informative

strategy, then the honest players do not randomize, and the mediator randomizes only with regard to the message it sends along with a STOP message. Moreover, an honest player terminates iff it gets a STOP message. Whether it gets one is determined by the scheduler's strategy, and the input of and randomization used by the players in $K \cup T$. If the players in $K \cup T$ use $\vec{\tau}_{K \cup T}$, then the scheduler can determine as soon as it has received the input and the coin tosses of the players in $K \cup T$ which honest players will terminate. Similarly, the scheduler can determine exactly when each honest player that terminates does so. In any case, once one of these conditions holds, the scheduler σ'_e delivers all of the messages not yet delivered, and from then on delivers messages immediately after they are sent. Thus, σ'_e is guaranteed to be standard.

Suppose that $(\vec{\tau}^*_{(K \cup T)}, \sigma^*_E)$ is a deterministic strategy in the support of $(\vec{\tau}'_{(K \cup T)}, \sigma'_E)$ that is in category A. Then, σ'_E and σ'_e are indistinguishable when the players in $K \cup T$ play $\vec{\tau}^*_{(K \cup T)}$. Thus, if $\vec{\sigma} + \sigma_d$ is a minimally-informative ϵ-(k,t)-robust (resp., strong ϵ-(k,t)-robust) equilibrium, then there exists a value $\epsilon' < \epsilon$ such that

$$\begin{aligned} & u'_i(\Gamma_d, (\vec{\sigma}_{-(K \cup T)}, \vec{\tau}^*_{(K \cup T)}), \sigma^*_E, \vec{x}_{(K \cup T)}) \\ = \; & u'_i(\Gamma_d, (\vec{\sigma}_{-(K \cup T)}, \vec{\tau}^*_{(K \cup T)}), \sigma'_e, \vec{x}_{(K \cup T)}) \\ < \; & u'_i(\Gamma_d, (\vec{\sigma}_{-T}, \vec{\tau}_T), \sigma_e, \vec{x}_T) + \epsilon' \qquad \text{[by Proposition 13]} \end{aligned}$$

for some (resp., for all) $i \in K$.

If $(\vec{\tau}^*_{(K \cup T)}, \sigma^*_E)$ is in category B, then again we have that if $\vec{\sigma} + \sigma_d$ is a minimally-informative ϵ-(k,t)-robust (resp., strong ϵ-(k,t)-robust) equilibrium, there exists a value $\epsilon' < \epsilon$ such that

$$\begin{aligned} & u'_i(\Gamma_d, (\vec{\sigma}_{-(K \cup T)}, \vec{\tau}^*_{(K \cup T)}), \sigma^*_E, \vec{x}_{(K \cup T)}) \\ < \; & u'_i(\Gamma_d, \vec{\sigma}, \sigma'_e, \vec{x}_{(K \cup T)}) \qquad \text{[by definition of } (2t+2k)\text{-punishment strategy]} \\ < \; & u'_i(\Gamma_d, (\vec{\sigma}_{-T}, \vec{\tau}_T), \sigma_e, \vec{x}_T) + \epsilon' \qquad \text{[by Proposition 12]} \end{aligned}$$

for some (resp., for all) $i \in K$.

Since $(\vec{\sigma}_{-(K\cup T)}, \vec{\tau}_{(K\cup T)}, \sigma_E)$ ϵ-$(K \cup T)$-$(t+k+1)$-coterminates, the probability that the strategy played by players in $K \cup T$ is in category C is at most ϵ. In this case, the payoff for each player is bounded by M. Using a compactness argument analogous to that of Proposition 12, there exists a value $\epsilon_0 < \epsilon$ such that $\epsilon' \leq \epsilon_0$ for all deterministic relaxed adversaries $(\vec{\tau}^*_{(K\cup T)}, \sigma^*_E)$ in the support of $(\vec{\tau}'_{(K\cup T)}, \sigma'_E)$ in categories A and B, and all inputs $\vec{x}_{(K\cup T)}$. So

$$\begin{aligned} & u'_i(\Gamma_d, (\vec{\sigma}_{-(K\cup T)}, \vec{\tau}^*_{(K\cup T)}), \sigma^*_E, \vec{x}_{(K\cup T)}) \\ < \;& u'_i(\Gamma_d, (\vec{\sigma}_{-T}, \vec{\tau}_T), \sigma_e, \vec{x}_T) + \epsilon_0 + \epsilon M, \end{aligned}$$

and therefore, if $\vec{\sigma} + \sigma_d$ is a minimally-informative ϵ-(k,t)-robust (resp., strong ϵ-(k,t)-robust) equilibrium, then

$$\begin{aligned} & u'_i(\Gamma_d, (\vec{\sigma}_{-(K\cup T)}, \vec{\tau}_{(K\cup T)}), \sigma_E, \vec{x}_{(K\cup T)}) \\ < \;& u'_i(\Gamma_d, (\vec{\sigma}_{-T}, \vec{\tau}_T), \sigma_e, \vec{x}_T) + \epsilon_0 + \epsilon M \end{aligned}$$

for all inputs $\vec{x}_{(K\cup T)}$, all standard schedulers σ_e, all strategies $\vec{\tau}_{(K\cup T)}$, and for some (resp., for all) $i \notin T$.

It remains to prove the result in the case that A is not N-bounded with respect to $\vec{\sigma}$ for some N. Fix $\epsilon' > 0$ and let A'' be an N-bounded ϵ'-approximation of A with respect to $\vec{\sigma}$ given by Lemma 10. Then, by the previous argument

$$\begin{aligned} & u'_i(\Gamma_d, (\vec{\sigma}_{-(K\cup T)}, \vec{\tau}''_{(K\cup T)}), \sigma''_E, \vec{x}_{(K\cup T)}) \\ < \;& u'_i(\Gamma_d, (\vec{\sigma}_{-T}, \vec{\tau}_T), \sigma_e, \vec{x}_T) + \epsilon_0 + \epsilon M, \end{aligned}$$

and by Lemma 10 it follows that

$$\begin{aligned} & u'_i(\Gamma_d, (\vec{\sigma}_{-(K\cup T)}, \vec{\tau}_{(K\cup T)}), \sigma_E, \vec{x}_{(K\cup T)}) \\ < \;& u'_i(\Gamma_d, (\vec{\sigma}_{-T}, \vec{\tau}_T), \sigma_e, \vec{x}_T) + \epsilon_0 + \epsilon M + \epsilon' M \end{aligned}$$

for all input profiles $\vec{x}_{(K\cup T)}$, standard schedulers σ_e, and strategy profiles $\vec{\tau}_{(K\cup T)}$, and some (resp., all) $i \notin T$. Since this inequality holds for all $\epsilon' > 0$, the result follows. □

An analogous argument can be used if we have an (k,t)-robust equilibrium (not just an ϵ-(k,t)-robust equilibrium):

Proposition 15. *If $\vec{\sigma}+\sigma_d$ is a minimally-informative (k,t)-robust equilibrium in a mediator game $\Gamma_d(u_i')$ for which a $(k+t)$-punishment strategy exists, σ_E is a relaxed scheduler, T and K are disjoint sets of players with $|T| \leq t$ and $1 \leq |K| \leq k$, and $\vec{\tau}_{(K \cup T)}$ is a strategy profile for the players in $K \cup T$ such that $(\vec{\sigma}_{-(K \cup T)}, \vec{\tau}_{(K \cup T)}, \sigma_E)$ $(K \cup T)$-coterminates, then there exists a (standard) scheduler σ_e such that for all input profiles $\vec{x}$ and all $i \notin T$,*

$$\begin{aligned} & u_i'(\Gamma_d, (\vec{\sigma}_{-(K \cup T)}, \vec{\tau}_{(K \cup T)}), \sigma_d, \sigma_E, \vec{x}_{(K \cup T)}) \\ \leq \; & u_i'(\Gamma_d, (\vec{\sigma}_{-T}, \vec{\tau}_T), \sigma_d, \sigma_e, \vec{x}_T). \end{aligned}$$

Returning to the proof of Theorem 8, we can now prove (strong) (k,t)-robustness. Let $\Gamma_d(\vec{u}')$ be a utility variant of Γ_d such that $\vec{\sigma} + \vec{\sigma}_d$ is a (k,t)-robust equilibrium of $\Gamma_d(\vec{u}')$, let σ_e be a scheduler in $\Gamma_{CT}(\vec{u})$, and let K and T be disjoint subsets of players with $1 \leq |K| \leq k$ and $|T| \leq t$, respectively, such that $3k+4t < n$. Let $\vec{\tau}_K$ and $\vec{\tau}_T$ be strategy profiles for players in K and T, respectively. By Theorem 3 and Proposition 6, there exist a function H_{σ_e} and a relaxed scheduler σ_E in the mediator game such that

$$\begin{aligned} & u_i'(\Gamma_{CT}(\vec{u}'), (\vec{\sigma}_{CT})_{-(K \cup T)}, \vec{\tau}_K, \vec{\tau}_T), \sigma_e, \vec{x}_{(K \cup T)}) \\ = \; & u_i'(\Gamma_d(\vec{u}'), (\vec{\sigma}_{-(K \cup T)}, H_{\sigma_e}(\vec{\tau}_K), H_{\sigma_e}(\vec{\tau}_T)), \sigma_d, \sigma_E, \vec{x}_{(K \cup T)}) \end{aligned}$$

for all $i \notin T$ and all input profiles $\vec{x}$. We can assume without loss of generality that $(\vec{\sigma}, \sigma_d)$ is minimally informative. Thus, by Proposition 15, there exists a standard scheduler σ_e' such that

$$\begin{aligned} & u_i'(\Gamma_d(\vec{u}'), (\vec{\sigma}_{-(K \cup T)}, H_{\sigma_e}(\vec{\tau}_T), H_{\sigma_e}(\vec{\tau}_K)), \sigma_d, \sigma_E, \vec{x}_{(K \cup T)}) \\ \leq \; & u_i'(\Gamma_d(\vec{u}'), (\vec{\sigma}_{-T}, H_{\sigma_e}(\vec{\tau}_T)), \sigma_d, \sigma_e', \vec{x}_{(K \cup T)}). \end{aligned}$$

Finally, by Theorem 3, if $\vec{\sigma}$ is (k,t)-robust (resp., strongly (k,t)-robust), then there exists a standard scheduler σ_e'' such that

$$\begin{aligned} & u_i'(\Gamma_{CT}(\vec{u}'), ((\vec{\sigma}_{CT})_{-T}, \vec{\tau}_T), \sigma_e, \vec{x}_T) \\ = \; & u_i'(\Gamma_d(\vec{u}'), (\vec{\sigma}_{-T}, H_{\sigma_e}(\vec{\tau}_T)), \sigma_d, \sigma_e'', \vec{x}_T) \quad \text{[by Theorem 3]} \\ = \; & u_i'(\Gamma_d(\vec{u}'), (\vec{\sigma}_{-T}, H_{\sigma_e}(\vec{\tau}_T)), \sigma_d, \sigma_e', \vec{x}_T) \quad \text{[by Corollary 1]} \end{aligned}$$

for some $i \in K$ (resp., for all $i \in K$). Therefore,

$$\begin{aligned} & u_i'(\Gamma_{CT}(\vec{u}'), ((\vec{\sigma}_{CT})_{-(K \cup T)}, \vec{\tau}_K, \vec{\tau}_T), \sigma_e, \vec{x}_{(K \cup T)}) \\ \leq \; & u_i'(\Gamma_{CT}(\vec{u}'), ((\vec{\sigma}_{CT})_{-T}, \vec{\tau}_T), \sigma_e, \vec{x}_T), \end{aligned}$$

as desired. □

We remark that, with a little more effort, we can show that the minimally-informative strategy profile $f(\vec{\sigma} + \sigma_d)$ that implements $\vec{\sigma} + \sigma_d$ is actually a $(t+k)$-bisimulation of $\vec{\sigma} + \sigma_d$. Moreover, the strategy profile that implements $f(\vec{\sigma} + \sigma_d)$ in the cheap-talk game preserves all the properties of the cheap-talk equilibrium in Theorem 8. Thus, under the conditions of Theorem 8, we can get a strategy profile in the cheap-talk game that $(t+k, t)$-bisimulates a strategy profile in the mediator game.

4.4.6 Proof of Theorem 9

To prove Theorem 9, we use an analogous strategy to that used to prove Theorem 8, using Theorem 4 instead of Theorem 3. The same argument as that used in the proof of Theorem 7 shows that for all $\epsilon' \in (0, 1]$ there exists a protocol $\vec{\sigma}_{CT}$ that ϵ'-implements $\vec{\sigma} + \sigma_d$ and that $\vec{\sigma}_{CT}$ is (ϵ, t)-immune.

To prove (strong) ϵ-(k,t)-robustness, fix an adversary $A = (\vec{\tau}_K, \vec{\tau}_T, \sigma_e)$ for subsets K, T such that $1 \leq |K| \leq k$, $|T| \leq t$ and $K \cap T = \emptyset$. By Theorem 4 and

Proposition 7, there exists a function H_{σ_e} from strategies to strategies and a relaxed scheduler σ_E such that, for all input profiles $\vec{x}$,

$$\begin{aligned} & u_i(\Gamma_{CT}, ((\vec{\sigma}_{CT})_{-(K\cup T)}, \vec{\tau}_K, \vec{\tau}_T), \sigma_e, \vec{x}_{(K\cup T)}) \\ < \; & u_i(\Gamma_d, (\vec{\sigma}_{-(K\cup T)}, H_{\sigma_e}(\vec{\tau}_K), H_{\sigma_e}(\vec{\tau}_T)), \sigma_d, \sigma_E, \vec{x}_{(K\cup T)}) + \epsilon' M \end{aligned}$$

for all $i \in K$.

By Theorem 4, there exists a standard scheduler σ'_e such that

$$\begin{aligned} & u_i(\Gamma_{CT}, ((\vec{\sigma}_{CT})_{-T}, \vec{\tau}_T), \sigma_e, \vec{x}_T) \\ > \; & u_i(\Gamma_d, (\vec{\sigma}_{-T}, H_{\sigma_e}(\vec{\tau}_T)), \sigma_d, \sigma'_e, \vec{x}_T) - \epsilon' M. \end{aligned}$$

Thus, there exists some $\epsilon_0 < \epsilon$ such that if $\vec{\sigma}_d$ is (k,t)-robust (resp., strongly (k,t)-robust), then

$$\begin{aligned} & u_i(\Gamma_{CT}, ((\vec{\sigma}_{CT})_{-(K\cup T)}, \vec{\tau}_K, \vec{\tau}_T), \sigma_e, \vec{x}_{(K\cup T)}) \\ < \; & u_i(\Gamma_d, (\vec{\sigma}_{-(K\cup T)}, H_{\sigma_e}(\vec{\tau}_K), H_{\sigma_e}(\vec{\tau}_T)), \sigma_d, \sigma_E, \vec{x}_{(K\cup T)}) + \epsilon' M \\ < \; & u_i(\Gamma_d, (\vec{\sigma}_{-T}, H_{\sigma_e}(\vec{\tau}_T)), \sigma_d, \sigma'_e, \vec{x}_T) + \epsilon_0 + 2\epsilon' M \qquad \text{[by Proposition 14]} \\ < \; & u_i(\Gamma_{CT}, ((\vec{\sigma}_{CT})-T, \vec{\tau}_T), \sigma_e, \vec{x}_T) + \epsilon_0 + 3\epsilon' M \end{aligned}$$

for some $i \in K$ (resp., for all $i \in K$). Therefore, taking $\epsilon' = (\epsilon - \epsilon_0)/3M$, we have that $\vec{\sigma}_{CT}$ is a ϵ-(k,t)-robust equilibrium (resp., strongly (k,t)-robust equilibrium) for Γ_{CT}.

Again, as was the case for Theorem 8, with a little more effort we can show that under the conditions of Theorem 9, we can get a strategy profile in the cheap-talk game that ϵ-$(t+k,t)$-bisimulates a strategy profile in the mediator game.

4.5 Proofs (Synchronous Case)

The proof of Theorem 10 is analogous to that of Theorem 6, except that we use the protocol of Theorem 5 except of that of Theorem 3. Unfortunately, the proof

of Theorem 11 cannot be generalized from that of Theorem 8. The reason for this is that the broadcast properties described in Section 2.2.1 are guaranteed only if $n > 3(t + k)$. This means that players may not necessarily be able to agree on punishing if some of them do not terminate. The proof of Theorem 8 is presented next.

4.5.1 Reducing the game

During the following section, we denote by D the message space (which is assumed to be a finite field), by I the input (or type) space of the players, and by E_i the set of actions of player i. We also denote $E := E_1 \times \ldots \times E_n$.

Define a one-round protocol $\vec{\sigma} + \sigma_d$ as a protocol in which all players send a message to the mediator at round 1, the mediator computes some value v with the messages received in round 1, and sends v to all players in round 2.

Proposition 16. *Given a mediator game Γ_d and a (k,t)-robust strategy $\vec{\sigma}+\sigma_d$ for Γ_d, there exists a one-round (k,t)-robust protocol $\vec{\sigma}' + \sigma'_d$ that implements[2] $\vec{\sigma} + \sigma_d$.*

Proof. Let $f^{\vec{\sigma}+\sigma_d} : (I \times F)^n \to F^n$ be the (randomized) function that takes as input a tuple of pairs $((x_1, p_1), \ldots, (x_n, p_n))$ with $x_i \in I$ and $p_i \in F$, and outputs $(a_1 \oplus p_1, \ldots, a_n \oplus p_n)$, where a_i is the action that player i would play with $\vec{\sigma} + \sigma_d$ given input $\vec{x}$. Consider the protocol $\vec{\sigma}' + \sigma'_d$ in which each player i, in round 1, computes a one-time pad $p_i \in F$ uniformly at random, and sends (x_i, p_i) to the mediator, where x_i is i's input. Then the mediator computes $(z_1, \ldots, z_n) := f^{\vec{\sigma}+\sigma_d}((x_1, p_1), \ldots, (x_n, p_n))$, taking both x_i and p_i to be 0 if i sent a message with

[2]Since there is no scheduler, this means they have the same output distribution if all players are honest.

an inappropriate format or no message at all at round 1. The mediator sends $\vec{z}$ to all players in round 2. After receiving this message, each player i plays $z_i \oplus p_i$.

It is straightforward to check that $\vec{\sigma}' + \sigma'_d$ is (k,t)-robust and implements $\vec{\sigma} + \sigma_d$. □

Note that one-round protocols can be reduced to secure computation, since the value v that the mediator sends is just a function of the inputs received. Thus, constructing $\vec{\sigma}_{ACT}$ reduces to implementing secure computation. Implementing secure computation can be reduced to implementing broadcast channels, implementing VSS, and implementing CC. We consider these three protocols in turn.

4.5.2 Implementing Consensus and Broadcast

Given a history $\vec{h}$ of a protocol for n players and a subset $S \subseteq [n]$, let $\vec{h}^{-S}$ be the history obtained by removing all messages sent and received by players in S (so $\vec{h}^{-S}$ is just like $\vec{h}$, except that players in S do not send messages). If S has the form $S = [n+1, m]$, where $m > n$, let $\vec{h}^{+S}$ be the extension of $\vec{h}$, viewed as a history for m players, in which the last $m - n$ players are not scheduled.

Definition 18. *Fix $m < n$.*

- *A protocol $\vec{\pi}'$ for m players is an* m-compression *of a protocol $\vec{\pi}$ for n players if $\vec{\pi}'$ is identical to π_i for $1 \leq i \leq m$, except that if i sends a message to a player in $\{m+1, \ldots, n\}$ with π_i, it does not do so with π'_i.*
- *A protocol $\vec{\pi}'$ for n players is an n-extension of a protocol $\vec{\pi}$ for m players if, given a history h_i of $\vec{\pi}'$, player i using π'_i acts as it does with π_i given history*

$h_i^{-[m+1,n]}$, *which is identical to* h_i *except that all messages to and from players in* $\{m+1,\ldots,n\}$ *are omitted.*

Intuitively, these two definitions capture the idea of expanding or shrinking the number of players in a protocol while the original players continue to do essentially the same thing.

Definition 19. *Let* $\vec{\pi}$ *be any protocol for* n *players and* A *be any adversary. We say that a finite history* $\vec{h}$ *is* possible *in* $(\vec{\pi}, A, \vec{x})$ *if there exists a history* $\vec{h}'$ *that extends* $\vec{h}$ *in which players play* $\vec{\pi}$ *with adversary* A *and input profile* $\vec{x}$*.*

Basically, a finite history is possible in $(\vec{\pi}, A)$ if it could be the history of $(\vec{\pi}, A, \vec{x})$ at some point in time with some choice of randomization.

Lemma 11. *If* $\vec{\pi}$ *is a protocol for* n *players in an asynchronous system,* $\vec{\pi}'$ *is an* m*-compression of* $\vec{\pi}$*,* $\vec{x}$ *is an input profile for* n *players,* σ_e' *is a scheduler for* $\vec{\pi}'$*,* $T \subseteq [m]$*,* $\vec{\tau}_T'$ *is a strategy for the players in* T*,* $\vec{\tau}_T$ *is an* n*-extension of* $\vec{\tau}_T'$*, and* $\vec{h}$ *is a possible history in* $(\vec{\pi}', (T, \tau_T', \sigma_e'), \vec{x}_{[m]})$*, then there exists a scheduler* σ_e *such that* $\vec{h}^{+[m+1,n]}$ *is possible in* $(\vec{\pi}, (T, \tau_T, \sigma_e), \vec{x})$*.*

Proof. Let N be the number of scheduler actions taken in $\vec{h}$. Consider a scheduler σ_e in the system with n players that acts like σ_e' but does not schedule any player in $\{m+1,\ldots,n\}$ until it has taken N actions. After that, it acts arbitrarily (for instance, it may schedule players in round-robin and deliver all messages immediately after they are sent). It is easy to check that, by construction, $\vec{h}^{+[m+1,n]}$ is possible in $(\vec{\pi}, (T, \tau_T, \sigma_e))$. □

This lemma allows us to prove the following:

Theorem 12. *If $n > 2k + 3t$ and the input space of the players is $\{0, 1\}$, then there exists a protocol $\vec{\pi}$ with the following properties:*

(a) For all histories with adversaries of size at most $t + k$, if two honest players i and j terminate with output o_i and o_j respectively, then $o_i, o_j \in \{0, 1\}$ and $o_i = o_j$. Moreover, if all honest players have a common input v, then $o_i = o_j = v$.

(b) For all histories with adversaries of size at most t, all honest players terminate.

Proof. Let $\vec{\pi}^{n+k}$ be Bracha's implementation of consensus for $n + k$ players that tolerates up to $t + k$ malicious players (note that there is such a protocol since $n + k > 3(t + k)$), and let $\vec{\pi}$ be the n-compression of $\vec{\pi}^{n+k}$. If $\vec{\pi}$ does not satisfy (a), then there exists an adversary $A = (T, \vec{\tau}'_T, \sigma'_e)$ of size at most $t + k$, an input profile $\vec{x}$, and a finite history $\vec{h}$ possible in $(\vec{\pi}, A, \vec{x})$ such that (a) is not satisfied in $\vec{h}$. Let $\vec{x}'$ be an extension of $\vec{x}$ in which $x_{n+1}, \ldots, x_{n+k}$ are set to the majority of the remaining inputs (and set to 1 if the number of players with eachinput is the same), and let $\vec{\tau}_T$ be an $(n + k)$-extension of $\vec{\tau}'_T$. By Lemma 11, there exists a scheduler σ_e such that $\vec{h}^{+[n+1,n+k]}$ is possible in $(\vec{\pi}^{n+k}, (T, \vec{\tau}_T, \sigma_e), \vec{x}')$, which would contradict the assumption that $\vec{\pi}^{n+k}$ satisfies (a).

If $\vec{\pi}$ does not satisfy (b), then there exists an adversary $A = (T, \vec{\tau}'_T, \sigma'_e)$ and an input profile $\vec{x}$ such that $(\vec{\pi}, A, \vec{x})$ has a non-zero probability of deadlock. Let $K = \{n + 1, \ldots, n + k\}$ and consider an adversary $A' = (K \cup T, \vec{\tau}_T + \vec{\tau}''_K, \sigma_e)$, in which $\vec{\tau}_T$ is an $(n + k)$-extension of $\vec{\tau}'_T$, τ''_i consists of sending no messages at all, and σ_e acts like σ'_e except that between each pair of consecutive actions it schedules a player in K using round-robin. This is made so that σ_e satisfies the constraint

that all players must be scheduled eventually. By construction, for any input profile $\vec{x}'$ such that $\vec{x}'_{[n]} = \vec{x}$, the probability of deadlock of each player in $[n] - T$ when playing $(\vec{\pi}, A, \sigma'_e)$ is identical to its probability of deadlock when playing $(\vec{\pi}^{n+k}, A', \sigma_e)$. This would mean that there is an honest player in $(\vec{\pi}^{n+k}, A', \sigma_e)$ with a non-zero probability of deadlock. This contradicts the assumption that $\vec{\pi}^{n+k}$ tolerates $t + k$ malicious players. □

What this theorem shows is that we can construct a protocol such that all honest players that terminate agree on the same value, and if in addition all honest players have the same input, then those that terminate do so with that value. We next show that we can guarantee that if at least $n-2k-2t$ honest players terminate, then all of them do.

Consider a protocol $\vec{\pi}'$, which consists of running the protocol $\vec{\pi}$ given by Theorem 12, but just before a player outputs a value v, it sends a message of the form $(decide, v)$ to all the other players. Also, if a player receives at least $t + k + 1$ messages of the form $(decide, v)$ from different senders, it outputs v and sends a $(decide, v)$ message to each other player as well (if it hasn't done this before).

Theorem 13. *Protocol $\vec{\pi}'$ satisfies properties (a) and (b) in Theorem 12 and*

(c) for all histories with adversaries of size at most $t + k$, if an honest player does not terminate, at least $n - 2k - 2t$ honest players do not terminate.

Proof. It is easy to check that $\vec{\pi}'$ still satisfies (a) and (b). To see that it satisfies (c), if the number of honest players that do not terminate is less than $n - 2k - 2t$, then the number of honest players that terminate is at least $t+k+1$. By (a), each of these honest players sends a message of the form $(decide, v)$ at some point, all

with the same value v. Thus, all other honest players must terminate and output v as well. □

Note that although $\vec{\pi}'$ was designed for asynchronous systems, since synchronous systems are just a special case of asynchronous systems, it still has properties (a), (b), and (c) in synchronous systems.

We say that a player *broadcasts* a message m if it runs a protocol that satisfies the three properties in Theorem 13 with input m. For some of our applications, it is necessary that property (a) is satisfied, and thus we require players to broadcast their messages. However, in situations where $n \leq 2k + 3t$, we cannot guarantee a broadcast. In these situations, we just require that the sender sends message m to all players. But note that in this case malicious players may send different messages to different players.

4.5.3 VSS

Proposition 17. *If $n > 2k + 3t$, then there exists a protocol $\vec{\pi}_r$ with the following properties:*

(a) For all histories where the adversary controls at most $t + k$ players, if the honest players $a_1, \ldots, a_m$ terminate with outputs $o_1, \ldots, o_m$, respectively, then there exists a polynomial p of degree $t + k$ such that $p(a_i) = o_i$. Moreover, if the sender r is honest, then $p(0) = x_r$, where x_r is r's input.

(b) In all histories where the adversary has size t, all the honest players terminate.

(c) *Let $\vec{H}_r^x(A)$ be the distribution of local histories when running $\vec{\pi}_r$ with an adversary A, and in which r's input is x. If the sender r is honest, then for all adversaries A of size at most t and all inputs $x, y \in I$, $\vec{H}_r^x(A)$ and $\vec{H}_r^y(A)$ are identically distributed.*

Proof. Ben-Or, Goldwasser and Widgerson's VSS implementation has these properties if we use the broadcast implementation presented in the previous section. □

4.5.4 CC

Proposition 18. *If $n > 2k + 3t$ and there exist two polynomials p and q of degree $t + k$ such that, for all players i, the input of player i is a pair (x_i, y_i) where $p(i) = x_i$ and $q(i) = y_i$, then there exists a protocol $\vec{\pi}$ with the following properties:*

(a) *For all histories where the adversary controls at most $t + k$ players, if the honest players $a_1, \ldots, a_m$ terminate with outputs $o_1, \ldots, o_m$, respectively, then there exists a polynomial P of degree $t + k$ such that $P(a_i) = o_i$ for all i and $P(0) = p(0)q(0)$.*

(b) *For all histories with adversaries of size at most t, all honest players terminate.*

(c) *Given two polynomials p, q of degree n, let $\vec{H}^{p,q}(A)$ be the distribution of local histories when running $\vec{\pi}$ with an adversary A, and in which each player i has input $(p(i), q(i))$ respectively. Then for all adversaries $A = (T, \vec{\tau}_T)$ of size at most $t + k$ and all polynomials p, q, r, s with $p(i) = r(i)$ and $q(i) = s(i)$ for all $i \in T$, $\vec{H}^{p,q}(A)$ and $\vec{H}^{r,s}(A)$ are identically distributed.*

(d) Let $\vec{\pi}_*$ be the protocol in which each player runs $\vec{\pi}$ and then sends the output to each other player, and let $H_*^{p,q}(A)$ be defined as in (c) except that players play $\vec{\pi}_*$ instead of $\vec{\pi}$. Then for all adversaries $A = (T, \vec{\tau})$ of size at most $t+k$ and all polynomials p, q, r, s with $p(i) = r(i)$ and $q(i) = s(i)$ and such that $p(0)q(0) = r(0)s(0)$, $\vec{H}^{p,q}(A)$ and $\vec{H}^{r,s}(A)$ are identically distributed.

Proof. Ben-Or, Goldwasser and Widgerson's VSS implementation satisfies these properties. □

4.5.5 Constructing $\vec{\sigma}_{ACT}$

First we show that we can implement secure computation when $n > 2k+3t$ in such a way that conditions (b) and (c) are satisfied, without making any assumptions about the punishment strategy.

Given a function f, consider the following protocol $\vec{\pi}^f$:

Step 1: Each player shares its input using the VSS implementation described in Section 4.5.3.

Step 2: Players perform CC to compute $f(\vec{x})$, using the implementation described in Section 4.5.4.

Step 3: Players broadcast an 'OK' message when they complete Step 2. Each player waits until it receives at least $n - t$ broadcast messages before continuing to Step 4.

Step 4: Each player i sends its output o_i of the CC protocol of Step 2 to all other players.

Step 5: Let $s_{i,j}$ be the message received by i from player j the turn after performing Step 4 (if it received a message; if not, i takes $s_{i,j} = 0$). If there exists a set $S \subseteq [n]$ with $|S| \geq n - t$ and a polynomial p_i of degree $t + k + 1$ such that $p(j) = s_{i,j}$ for all $j \in S$, then i sends a message of the form $(final, p_i(0))$ to all players, outputs $p_i(0)$, and terminates.

Step 6: If a player i was not able to complete Step 5 but eventually receives at least $t + k + 1$ messages of the form $(final, v)$, it outputs v and sends the message $(final, v)$ to all other players.

Theorem 14. *If $n > 2k + 3t$ and f is a function with n inputs, then $\vec{\pi}^f$ satisfies the following properties:*

(a) For all histories with adversaries $A = (T, \vec{\tau}_T)$ of size at most $t + k$ in which all players not in T terminate, there exists a vector $\vec{y}_T$ of size $|T|$ such that all honest players output $f\left(\vec{x}/_{(T,\vec{y}_T)}\right)$.

(b) For all histories with adversaries of size at most t, all honest players terminate.

(c) For all histories with adversaries of size at most $t + k$, if an honest player does not terminate, at least $n - 2k - 2t$ honest players do not terminate.

(d) For all adversaries $A = (T, \vec{\tau}_T)$ with $|T| \leq t + k$, if some player not in T completes Step 4, then the output of players in T is a (randomized) function of $\vec{x}_T$ and the output $f\left(\vec{x}/_{(T,\vec{y}_T)}\right)$ of honest players; otherwise, it is just a function of $\vec{x}_T$.

Proof. Since all the primitives used satisfy property (b), $\vec{\pi}^f$ satisfies (b) as well. To see that it satisfies (c) assume that less than $n - 2t - 2k$ honest players do not terminate. Since the adversary is at most of size $t + k$, at least $t + k + 1$ honest

players i terminate the protocol and thus were able to compute the polynomial p_i at Step 5. Note that since p_i is of degree $t+k$ and at least $t+k+1$ values $s_{i,j}$ used in the construction of p_i were received from honest players, it is guaranteed that all honest players who terminate output the same value v. This means that each other honest player receives at least $t+k+1$ messages of the form $(final, v)$ and terminate as well.

Property (a) follows from property (a) of the VSS and CC implementations presented in Sections 4.5.3 and 4.5.4: To see this, suppose that i is an honest player. Then property (a) of the VSS implementation guarantees that the shares $s_{i,j}$ computed by each other honest player after completing i's VSS instance in step 1 encode i's input x_i. If i is not honest, then this is not necessarily true; however, the shares computed by honest players are guaranteed to encode some secret y_i in the input space. Thus, the tuple of secrets used in the CC phase at step 2 is of the form $\vec{x}/_{(T,\vec{y}_T)}$ for some vector $\vec{y}$. Finally, by property (a) of the CC implementation, the shares at the end of the circuit computation of f encode $f\left(\vec{x}/_{(T,\vec{y}_T)}\right)$.

By property (c) of the VSS and CC implementations, the distribution of histories before Step 4 of any subset T of at most $t+k$ players is independent of the inputs of playes not in T. This means that if no player not in T completes Step 4, the output of an adversary $(T, \vec{r}_T)$ with $|T| \leq t+k$ is a function only of its input $\vec{x}_T$. Similarly, by property (d) of the CC implementation, even if some honest player completes Step 4, the output of the adversary is a function only of its input and the output of f. This proves property (d). □

4.5.6 Proof of Theorem 11

We present next a naive construction of $\vec{\sigma}_{ACT}$ that does not quite satisfy all properties of Theorem 3; however, it is quite simple and gives the intuition for the final construction. Given $\vec{\sigma} + \sigma_d$, let $\vec{\sigma}' + \sigma'_d$ the one-round (k,t)-robust strategy that implements $\vec{\sigma} + \sigma_d$ given by Proposition 16. Let $\vec{\sigma}^{naive}_{ACT}$ be the strategy where the players first run the protocol $\vec{\pi}^{(f^{\vec{\sigma}+\sigma_d})}$ of Theorem 14 and then each player i plays the action of i encoded in the output of $f^{\vec{\sigma}+\sigma_d}$. If a player does not terminate, it plays its part of the punishment strategy. It might seem that the properties of Theorem 14 guarantee that $\vec{\sigma}^{naive}_{ACT}$ has all the required properties. However, consider the game Γ with 20 players, in which the set of actions is $\{1, 2, \ldots, 20\} \cup \{\perp\}$ and the payoffs are defined as follows:

- If at least 6 players play $\perp$, then all players get a payoff of $min(0, x - 10)$ where x is the number of players playing $\perp$.
- If no more than 5 people play $\perp$ and at least 13 players play the same non-$\perp$ action i, then player i receives a payoff of 20 while the rest of the players receive a payoff of -1, while if no more than 5 players play $\perp$ and fewer than 13 players play the same action, then all players receive a payoff of 0.

Consider the following strategy $\vec{\sigma} + \sigma_d$ for Γ_d: In round 1, the mediator chooses a number in $\{1, \ldots, 20\}$ uniformly at random and sends that number to all players. In round 2, the players play whatever action they receive from the mediator. Clearly, this strategy is $(2,5)$-robust and has an expected payoff of $1/20$, which means that playing $\perp$ is a 14-punishment strategy. However, $\vec{\sigma}^{naive}_{ACT}$ is not $(2,5)$-robust: Consider an adversary $(K \cup T, \vec{\tau}_{K \cup T})$, in which $K \cup T = \{1, \ldots, 7\}$ and $\vec{\tau}_{K \cup T}$ consists of all players $i \in K \cup T$ acting as if they were honest, except that

in Step 3 of the secure computation of $f^{\vec{\sigma}+\sigma_d}$, i sends its output of the CC only to player 8 instead of sending it to all players. In the next round, each player $i \in K \cup T$ computes its action a_i as described above and, if $a_i \in T \cup K$, i sends a message of the form $(final, a_i)$ to all players and terminates; otherwise it plays $\perp$ and terminates.

Note that, in this scenario, all players $i \in \{1, \ldots, 8\}$ compute the same value a_i. Thus, if $a_i \in K \cup T$, in Step 5 all honest players are guaranteed to receive exactly 8 messages of the form $(final, a_i)$, implying that all honest players play a_i and terminate. However, if $a_i \notin K \cup T$, no player $j \in \{9, \ldots, 20\}$ will be able to complete Step 5, since j won't receive enough shares of the CC (note that j must receive 15 shares to complete the step). Moreover, in Step 6, j will receive only a single message of the form $(final, a_i)$, which means that j will play $\perp$. Thus, all players receive a payoff of 0. It is easy to check that, with adversary A, the expected payoff for players in $K \cup T$ is $7/10$, which is greater than the equilibrium payoff.

The problem with $\vec{\sigma}^{naive}_{ACT}$ is that the adversary can decide whether honest players will terminate after learning some useful information about outcomes. Therefore, it can make honest players play the punishment strategy if it is in its interest to do so. (Note that the punishment strategy is worse than the equilibrium payoff only in expectation.) To fix this problem, in $\vec{\sigma}_{ACT}$, instead of computing $f^{\vec{\sigma}+\sigma_d}$, players compute a function $f_p^{\vec{\sigma}+\sigma_d}$ that outputs whatever $f^{\vec{\sigma}+\sigma_d}$ would output with a fixed probability $p \in (0, 1]$, and otherwise it outputs a fixed value $\perp \in F \setminus f^{\vec{\sigma}+\sigma_d}(I^n)$ (which is a value that could not be the output of $f^{\vec{\sigma}+\sigma_d}$). If the output of $f_p^{\vec{\sigma}+\sigma_d}$ is $\perp$, players compute $f_p^{\vec{\sigma}+\sigma_d}$ again; they continue to do so until the output of $f_p^{\vec{\sigma}+\sigma_d}$ is not $\perp$. However, before starting a new computation, players tell other players

if they encountered any problems during the previous computation. If enough players say that they had problems, all players play the punishment strategy. The intuition behind this is that the adversary does not know before performing Step 3 of the computation of $f_p^{\vec{\sigma}+\sigma_d}$ if the output is going to be relevant or not. If the adversary deviates in such a way that not enough honest players are able to perform Step 4 and the output is $\perp$, the adversary will be punished. We will choose the probability of $\perp$ to be sufficiently high that it is not worthwhile for the adversary to deviate.

More precisely, $(\vec{\sigma}^p_{ACT})_i$ proceeds as follows: each player i repeatedly computes $f_p^{\vec{\sigma}+\sigma_d}$ securely, using the implementation presented in Theorem 14. After the ℓth computation of $f_p^{\vec{\sigma}+\sigma_d}$ (if there is one), each honest player does the following: if the output v^ℓ is not $\perp$, i plays its part of the action profile encoded in v^ℓ and terminates; otherwise, i sends $\mathit{continue}^\ell$ to all players if it didn't detect anything wrong during the ℓth computation of $f_p^{\vec{\sigma}+\sigma_d}$. Honest players begin the $(\ell+1)$st computation only if they receive enough $\mathit{continue}^\ell$ messages; if they never do, they play their part of the punishment strategy. The intuition behind this is that if honest players are able to detect deviations by rational players, it won't be worthwhile for rational players to deviate because there's a high chance that the output of $f_p^{\vec{\sigma}+\sigma_d}$ will not give them useful information. In that case, the deviation will be detected, and the honest players will play the punishment strategy.

In more detail, i sends a $\mathit{continue}^\ell$ message if, during the computation of $f_p^{\vec{\sigma}+\sigma_d}$, i completed Step 2 before completing Step 3 and if it completed Step 5 the round after completing Step 3. If i receives at least $n-t$ $\mathit{continue}^\ell$ messages, it sends a $\mathit{continue}^\ell_*$ message to all players. If i receives at least $t+k+1$ $\mathit{continue}^\ell_*$ messages, it starts the $(\ell+1)$st computation of $f_p^{\vec{\sigma}+\sigma_d}$ and sends a $\mathit{continue}^\ell_*$ message to all

players (if it hasn't done so already), even if i hasn't completed the ℓth computation of $f_p^{\vec{\sigma}+\sigma_d}$. If a player does not terminate the protocol (for instance, because it waits for *continue* messages from rational players) it ends plays the punishment strategy.

The purpose of this addition to $\vec{\sigma}_{ACT}^{naive}$ is to guarantee two important properties:

Proposition 19. *For all input profiles $\vec{x}$ and all adversaries $A = (T, \vec{\tau}_T)$ with $|T| \leq t + k$, if $n > 2k + 3t$, for all histories $\vec{h}$ of $\vec{\sigma}_{ACT}^{p}$ with input $\vec{x}$ and adversary A and all $\ell \in \mathbb{N}$, either all honest players begin the ℓth computation of $f_p^{\vec{\sigma}+\sigma_d}$ or at least $n - 2k - 2t$ honest players do not.*

Proposition 20. *For all input profiles $\vec{x}$ and all adversaries $A = (T, \vec{\tau}_T)$ with $|T| \leq t + k$, if $n > 2k + 3t$, for all histories $\vec{h}$ of $\vec{\sigma}_{ACT}^{p}$ with input $\vec{x}$ and adversary A and all $\ell \in \mathbb{N}$, if there exists a player $i \notin T$ that begins the $(\ell + 1)$ of $f_p^{\vec{\sigma}+\sigma_d}$, then at least $n - t - |T|$ honest players complete Step 5 of the ℓth computation of $f_p^{\vec{\sigma}+\sigma_d}$ immediately after completing Step 3 and complete Step 2 before Step 3.*

Proposition 19 says that after terminating a computation of $f_p^{\vec{\sigma}+\sigma_d}$ with output $\perp$, either all honest players begin the next computation or the number of those who do not is enough to guarantee that no honest player is able to terminate it, which means that all honest players play their part of the punishment strategy. Proposition 20 says that if even a single honest player begins the next computation of $f_p^{\vec{\sigma}+\sigma_d}$, it is guaranteed that a sufficiently large number of honest players received their shares of the output without delay. Note that this means that rational players cannot deviate as in the previous example, since they have to decide whether to send their shares before knowing if the output is $\perp$. If they choose not to send them (or at least not to send them to a sufficiently large number of honest players) and the output is $\perp$, honest players will stop playing and play the punishment strategy instead.

Proof of Proposition 19. If fewer than $n - 2t - 2k$ players do not begin the ℓth computation of $f_p^{\vec{\sigma}+\sigma_d}$, then at least $2k + 2t + 1$ players do. At least $t + k + 1$ of those players are honest, and thus all honest players are guaranteed to receive at least $t + k + 1$ messages of the form $continue_*^\ell$. □

Proof of Proposition 20. An honest player must receive at least $t+k+1$ $continue_*$ messages in order to continue with the next computation of $f_p^{\vec{\sigma}+\sigma_d}$. Since the adversary has size at most $t + k$, at least one such messages comes from an honest player, who must have received at least $n-t$ *continue* messages. At least $n-t-|T|$ of those $n-t$ *continue* messages must come from honest players, and those honest players must have completed Step 2 before Step 3, and Step 5 immediately after completing Step 3. □

We now show that this protocol has the required properties. There are many ways that rational and malicious players can deviate in $\vec{\sigma}_{ACT}$. Our first step is to provide a simple characterization of all possible deviations. Let $\mathcal{R}$ be the domain of random inputs. Given a subset $T \subseteq [n]$ and six functions

$$
\begin{aligned}
g_1 &: \mathbb{N} \times I^{|T|} \times \mathcal{R} \to I^{|T|}, \\
g_2 &: \mathbb{N} \times I^{|T|} \times \mathcal{R} \to 2^{[n]}, \\
g_3 &: \mathbb{N} \times I^{|T|} \times E^{|T|} \times \mathcal{R} \to 2^{[n]}, \\
g_4 &: \mathbb{N} \times I^{|T|} \times E^{|T|} \times \mathcal{R} \to E^{|T|}, \\
g_5 &: \mathbb{N} \times I^{|T|} \times \mathcal{R} \to \{0,1\}, \\
g_6 &: \mathbb{N} \times I^{|T|} \times \mathcal{R} \to E^{|T|},
\end{aligned}
$$

let $A^{(T,g_1,g_2,g_3,g_4,g_5,g_6)}$ be the adversary that performs the following in $\vec{\sigma}_{ACT}$:

1. During the ℓth computation of $f_p^{\vec{\sigma}+\sigma_d}$ (if there is one), each player $i \in T$ computes $\vec{y}_T^\ell = g_1(\ell, \vec{x}_T, r)$ and performs Steps 1, 2, and 3 acting as an

honest player with input y_i^ℓ.

2. In Step 4, each player $i \in T$ computes $S_1^\ell := g_2(\ell, \vec{x}_T, r)$, and instead of sending its output at Step 2 to all players, i sends it only to players in S_1^ℓ.

3. In Step 5, each player $i \in T$ reconstructs the output v^ℓ of $f_p^{\vec{\sigma}+\sigma_d}$.

   - If $v^\ell \neq \perp$, i decodes the actions $\vec{a}_T^\ell$ of the players in T, computes $S_2^\ell := g_3(\ell, \vec{x}_T, \vec{a}_T)$, sends a message of the form $(final, v^\ell)$ to players in S_2^ℓ, computes $\overrightarrow{act}_T^\ell := g_4(\ell, \vec{x}_T, \vec{a}_T, r)$, and then plays act_i^ℓ.
   - If $v^\ell = \perp$, i computes $b^\ell := g_5(\ell, \vec{x}_T, r)$. If $b^\ell = 1$, i sends a $continue^\ell$ message and a $continue_*^\ell$ message to all players. Otherwise, i computes $\vec{a}_T = g_6(\ell, \vec{x}_T, r)$, plays a_i, and terminates.

If the adversary is not able to finish computing $f_p^{\vec{\sigma}+\sigma_d}$ (e.g., because not enough honest players finished the previous computation), it computes $\vec{a}_T = g_6(\ell, \vec{x}_T, r)$, plays a_i, and terminates.

The intuition behind the construction of $A^{(T,g_1,g_2,g_3,g_4,g_5,g_6)}$ is that each of the functions g_i represents a simple deviation that an adversary may perform in $\vec{\sigma}_{ACT}^p$: g_1 encodes how the adversary lies about its input in every computation of $f_p^{\vec{\sigma}+\sigma_d}$; g_2 encodes how many honest players terminate each of the subsequent computations of $f_p^{\vec{\sigma}+\sigma_d}$ (note that this is determined by the adversary's strategy, given that the honest players play $\vec{\sigma}$); g_3 encodes how many honest players terminate if $f_p^{\vec{\sigma}+\sigma_d}$ is successfully computed and the output is not $\perp$; g_4 encodes which action profile the adversary plays in this situation; g_5 encodes whether the adversary sends *continue* and $continue_*$ messages to all honest players or to none, and g_6 encodes which action profile the adversary plays if it never learns the output of $f_p^{\vec{\sigma}+\sigma_d}$.

We next show that all adversaries A of size at most $t + k$ are equivalent to an

adversary of the form $A^{(T,g_1,g_2,g_3,g_4,g_5,g_6)}$ for some $T \subseteq [n]$ and functions g_1–g_6. To do this, we introduce the following notation. Given a protocol $\vec{\pi}$, an adversary A, and an input profile $\vec{x}$, let $(\vec{\pi}, A, \vec{x})$ be the output distribution that results when running $\vec{\pi}$ with adversary A and input $\vec{x}$.

Proposition 21. *For all adversaries $A = (T, \vec{\tau}_T)$ with $|T| \leq t + k$, there exist functions $g_1 : \mathbb{N} \times I^{|T|} \times \mathcal{R} \to I^{|T|}$, $g_2 : \mathbb{N} \times I^{|T|} \times \mathcal{R} \to [n]$, $g_3 : \mathbb{N} \times I^{|T|} \times E^{|T|} \times \mathcal{R} \to [n]$, $g_4 : \mathbb{N} \times I^{|T|} \times E^{|T|} \times \mathcal{R} \to E^{|T|}$, $g_5 : \mathbb{N} \times I^{|T|} \times \mathcal{R} \to \{0, 1\}$ and $g_6 : \mathbb{N} \times I^{|T|} \times \mathcal{R} \to E^{|T|}$ such that, for all inputs $\vec{x}$, $(\vec{\sigma}^p_{ACT}, A, \vec{x})$ and $(\vec{\sigma}^p_{ACT}, A^{(T,g_1,g_2,g_3,g_4,g_5,g_6)}, \vec{x})$ are identically distributed.*

Proof. Given an adversary $A = (T, \vec{\tau}_T)$ and the history $\vec{h}^\ell$ at the beginning of the ℓth computation of $f_p^{\vec{\sigma}+\sigma_d}$, let g_2^A be the (randomized) function that determines which subset of the players complete Step 5 of the ℓth computation of $f_p^{\vec{\sigma}+\sigma_d}$. Because the adversary's deviations are a function of its local history (i.e., the local histories of the agents in T), and the distribution over the adversary's local histories is independent of the inputs of honest players as long as no honest player has terminated Step 4 of the ℓth computation (property (d) of Theorem 14), g_2^A is ultimately a function of the input profile $\vec{x}_T$ of players in T and ℓ. Similarly, the function g_1^A that gives the inputs used by the adversary during the ℓth computation of $f_p^{\vec{\sigma}+\sigma_d}$ is also a function of $\vec{x}_T$ and ℓ. Note that by Proposition 19, if not all honest players start the $(\ell + 1)$st computation of $f_p^{\vec{\sigma}+\sigma_d}$, then at least $n - 2t - 2k$ honest players never start it. Thus, no honest player completes Step 5 of the $(\ell + 1)$st computation (since this would require messages from at least $2t + 2k + 1$ players) and no honest player begins any further computation. By Theorem 14(d), if no honest player terminates the $(\ell + 1)$st computation of $f_p^{\vec{\sigma}+\sigma_d}$, the actions that the players in T play are given by a function g_6^A of just ℓ and their input $\vec{x}_T$. Since none of the honest players terminates, they all play their part of the punishment

strategy. This means that the output depends only on whether all honest players begin the $(\ell+1)$st computation of $f_p^{\vec{\sigma}+\sigma_d}$. If all of them do, we can assume without loss of generality that all malicious players sent *continue* and *continue*$_*$ messages to all honest players, since the outcome would not change if they did. Similarly, if some honest player did not begin the $(\ell+1)$st computation, we can assume without loss of generality that none of the malicious players sent any *continue* or *continue*$_*$ messages, since again, the outcome would not change if this were the case. This can be encoded by a randomized Boolean function g_5 that, analogously to g_1 and g_2, depends just on $\vec{x}_T$ and ℓ. Finally, by property (c) of Theorem 14, the functions g_3^A and g_4^A that compute which players complete the ℓth computation of $f_p^{\vec{\sigma}+\sigma_d}$ and what actions the adversary plays if the output of $f_p^{\vec{\sigma}+\sigma_d}$ is not $\perp$ depend at most on ℓ, $\vec{x}_T$, and the output v^ℓ of $f_p^{\vec{\sigma}+\sigma_d}$. By the construction of $f_p^{\vec{\sigma}+\sigma_d}$, they both depend only on the actions of players in T encoded in v^ℓ, since this is the only information that the adversary learns from the output.

Clearly, g_1^A–g_6^A completely determine the output distribution of playing $\vec{\sigma}_{ACT}^p$ with adversary A, and, by construction, $g_j^{A^{(T,g_1^A,\ldots,g_6^A)}} = g_j^A$ for all $j \in \{1,\ldots,6\}$, as desired. □

Let M_i be the maximum possible payoff of player i in Γ (where the maximum is taken over all input profiles $\vec{x}$), let $Q_i^{\vec{x}}$ be the expected payoff of player i when players play $\vec{\sigma}+\sigma_d$ and the input profile is $\vec{x}$, and let P_i be the maximum payoff that player i can get when at least $n-k-t$ players play their part of the punishment strategy. By the definition of punishment strategy, $P_i < Q_i^{\vec{x}}$ for all $\vec{x} \in I^n$. Let $Q_i := \min_{\vec{x}\in I^n}\{Q_i^{\vec{x}}\}$, let $p_i := \frac{Q_i - P_i}{2(M_i - P_i)}$, and let $p^* := \min_{i\in[n]}\{p_i\}$. Note that $p^* \in (0,1)$ since $P_i < Q_i \leq M_i$. We next show that the protocol $\vec{\sigma}_{ACT}^{p^*}$ has the

desired properties. We start by showing that

$$u_i(\vec{\sigma}^{p^*}_{ACT}, A^{(T,g_1,g_2,g_3,g_4,g_5,g_6)}, \vec{x}) \leq u_i(\vec{\sigma}^{p^*}_{ACT}, \vec{x})$$

for all $T \subseteq [n]$ with $|T| \leq t + k$, all functions $g_1, \ldots, g_6$, all $i \in [n]$, and all $\vec{x} \in I^n$. To do this, first note that we can assume that if there exist functions $g_1, \ldots, g_6$ that maximize $u_i(\vec{\sigma}^{p}_{ACT}, A^{(g_1,g_2,g_3,g_4,g_5,g_6)}, \vec{x})$, without loss of generality, none of these functions depends on ℓ (the first component of their input). We also assume without loss of generality that the adversary has size strictly greater than t, since all primitives are guaranteed to terminate for adversaries of size at most t (which means that the actual value of $f_p^{\vec{\sigma}+\sigma_d}$ is computed regardless of what the adversary does).

By Proposition 20, if $|T| > t$ and the output of the ℓth computation of $f_p^{\vec{\sigma}+\sigma_d}$ is $\perp$, all honest players begin the $(\ell+1)$st computation if and only if the intersection of $g_2(\ell, \vec{x}_T, r)$ and $[n] \setminus T$ has at least $n - t - |T|$ members. Moreover, since $|T| > t$, it is necessary that the adversary sends *continue* and *continue*$_*$ messages as well, and thus that $g_5(\ell, \vec{x}_T, r) = 1$.

This motivates the following notation: let $\alpha_i^{\vec{x}_T}$ be the probability that the intersection of the output of g_2^A and $[n] \setminus T$ has at least $n - t - |T|$ members given input $\vec{x}_T$, and let $\beta_i^{\vec{x}_T}$ be the probability that the output of g_5 is 1 given $\vec{x}_T$ conditional on the intersection of the output of g_2^A and $[n] \setminus T$ having size at least $n - t - |T|$. Note that by our previous observation, we can assume that neither of these probabilities depends on ℓ. To simplify the notation, we define $U_i^{\vec{x}} := u_i(\vec{\sigma}^{p}_{ACT}, A^{(g_1,g_2,g_3,g_4,g_5,g_6)}, \vec{x})$. With this notation we can bound the expected payoff of player i as follows:

$$U_i^{\vec{x}} \leq (1-p)(\alpha_i^{\vec{x}_T}\beta_i^{\vec{x}_T}U_i^{\vec{x}} + (1 - \alpha_i^{\vec{x}_T}\beta_i)^{\vec{x}_T}P_i) + p(\alpha_i^{\vec{x}_T}Q_i^{\vec{x}} + (1 - \alpha_i^{\vec{x}_T})M_i).$$

To see this, note that the output of $f_p^{\vec{\sigma}+\sigma_d}$ is $\perp$ with probability $1-p$; moreover, if the output of $f_p^{\vec{\sigma}+\sigma_d}$ is $\perp$, then all players continue to the next computation of $f_p^{\vec{\sigma}+\sigma_d}$ if and only if the intersection of the output of g_2 with $[n] \setminus T$ has size at least $n-t-|T|$ and the output of g_5 is 1, which occurs with probability $\alpha_i^{\vec{x}_T}\beta_i^{\vec{x}_T}$. If this is the case, then player i gets a payoff of $U_i^{\vec{x}}$, since all players begin a fresh computation of $f_p^{\vec{\sigma}+\sigma_d}$. Otherwise, at least $n-t-|T|$ honest players don't terminate and don't begin the next computation of $f_p^{\vec{\sigma}+\sigma_d}$, guaranteeing that no honest player will be able to complete the next computation of f. In this case, all honest players play their part of the punishment strategy and i's payoff is bounded by P_i. If the output of $f_p^{\vec{\sigma}+\sigma_d}$ is not $\perp$, which occurs with probability p if the intersection of the output of g_2 and $[n] \setminus T$ has size at least $n-t-|T|$ at least $t+k+1$ honest players are able to complete Step 5 of the computation of $f_p^{\vec{\sigma}+\sigma_d}$, which guarantees that all honest players complete Step 6. This means that all honest players play the action for them encoded in the output of $f_p^{\vec{\sigma}+\sigma_d}$. In this scenario, note that the payoff of i is bounded by $Q_i^{\vec{x}}$ even though rational and malicious players might play actions other than the ones given by $f_p^{\vec{\sigma}+\sigma_d}$. Otherwise, the same deviation (playing an action other than the one given by the mediator) could increase i's payoff in $\vec{\sigma}+\sigma_d$, which would contradict the assumption that $\vec{\sigma}+\sigma_d$ is (k,t)-resilient. An analogous argument can be used to show that i's payoff is still bounded by $Q_i^{\vec{x}}$ even though rational and malicious players may play as if they had inputs other than $\vec{x}_T$. Finally, in the remaining case, i's payoff is bounded by its maximum possible payoff M_i. If we isolate $U_i^{\vec{x}}$ from the previous equation, we get that

$$U_i^{\vec{x}} \leq \frac{(1-p)(1-\alpha_i^{\vec{x}_T}\beta_i^{\vec{x}_T})P_i + p\alpha_i^{\vec{x}_T}Q_i^{\vec{x}} + p(1-\alpha_i^{\vec{x}_T})M_i}{1-(1-p)\alpha_i^{\vec{x}_T}\beta_i^{\vec{x}_T}}.$$

The following lemma implies that $U_i^{\vec{x}} \leq Q_i^{\vec{x}}$, which shows that $\vec{\sigma}_{ACT}^{p}$ is (k,t)-resilient; t-immunity follows easily from property (b) of Theorem 14.

Lemma 12. *If $A < B \leq C$ and $0 < p < \frac{B-A}{C-A}$, then*

$$\sup_{x,y\in[0,1]} \frac{(1-p)(1-xy)A + pxB + p(1-x)C}{1-(1-p)xy} \leq B. \tag{4.3}$$

Proof. Let $\epsilon_1 = B - A$ and let $\epsilon_2 = C - B$. By assumption, $\epsilon_1 > 0$ and $\epsilon_2 \geq 0$. Replacing A with $B - \epsilon_1$ and C with $B + \epsilon_2$ in the numerator of the left-hand side of (4.3), and then rearranging terms, we get that

$$\frac{(1-p)(1-xy)A + pxB + p(1-x)C}{1-(1-p)xy} = B - \frac{(1-p)(1-xy)\epsilon_1 - p(1-x)\epsilon_2}{1-(1-p)xy}$$

and $p < \frac{\epsilon_1}{\epsilon_2+\epsilon_1}$.

Let $g(x,y) = (1-p)(1-xy)\epsilon_1 - p(1-x)\epsilon_2$. This lemma reduces to the fact that $g(x,y) \geq 0$ if $x, y \in [0,1]$. We can rewrite $g(x,y)$ as

$$g(x,y) = (1-xy)\epsilon_1 - p((1-xy)\epsilon_1 + (1-x)\epsilon_2).$$

Since $p < \frac{\epsilon_1}{\epsilon_2+\epsilon_1}$, it follows that

$$g(x,y) \geq (1-xy)\epsilon_1 - \frac{\epsilon_1((1-xy)\epsilon_1 + (1-x)\epsilon_2)}{\epsilon_1+\epsilon_2},$$

or equivalently,

$$g(x,y) \geq \frac{\epsilon_1\epsilon_2 x(1-y)}{\epsilon_1+\epsilon_2},$$

which is clearly positive for $(x,y) \in [0,1]^2$. □

CHAPTER 5

IMPLEMENTING SEQUENTIAL EQUILIBRIA

In this chapter we extend the results in Chapter 4 by showing that any k-resilient sequential equilibrium with a mediator can also be implemented without the mediator if $n > 3k$ in synchronous systems, or if $n > 4k$ in asynchronous systems. We begin with an introduction of the necessary game-theoretic concepts that are needed to understand the statements and results.

5.1 Game-Theoretic Definitions

A normal-form game Γ is a tuple (P, A, U), where $P = \{1, \ldots, n\}$ is the set of *players*, $A = A_1 \times \cdots A_n$, where A_i is the set of possible actions for player $i \in P$, and $U = (u, \ldots, u_n)$ is an n-tuple of *utility* functions $u_i : A \to \mathbb{R}$, again, one for each player $i \in P$. A *pure strategy profile* $\vec{a}$ is an n-tuple of actions $(a_1, \ldots, a_n)$, with $a_i \in A_i$. A *mixed strategy* for player i is an element of $\Delta(A_i)$, the set of probability distributions on A_i. We extend u_i to mixed strategy profiles $\sigma = (\sigma_1, \ldots, \sigma_n)$ by defining $u_i(\sigma) = \sum_{(a_1,\ldots,a_n)\in A} \sigma_1(a_1) \ldots \sigma_n(a_n) u_i(a_1, \ldots, a_n)$: that is, the sum over all pure strategy profiles $\vec{a}$ of the probability of playing $\vec{a}$ according to σ times the utility of $\vec{a}$ to i.

Definition 20. *In a normal-form game Γ, a (mixed) strategy profile $\vec{\sigma} = (\sigma_1, \ldots, \sigma_n)$ is a* Nash equilibrium *if, for all $i \in P$ and strategies $\sigma_i' \in \Delta(A_i)$*

$$u_i(\sigma_i, \vec{\sigma}_{-i}) \geq u_i(\sigma_i', \vec{\sigma}_{-i}).$$

Definition 21. *In a normal-form game $\Gamma = (P, A, U)$, a distribution $p \in \Delta(A)$ is a* correlated equilibrium *(CE) if, for all $i \in P$ and all $a_i', a_i'' \in A_i$ such that*

$p(a'_i) > 0$,

$$\sum_{\vec{a} \in \Delta(A): a_i = a'_i} u_i(a'_i, \vec{a}_{-i}) p(\vec{a} \mid a_i = a'_i) \geq \sum_{\vec{a} \in \Delta(A): a_i = a'_i} u_i(a''_i, \vec{a}_{-i}) p(\vec{a} \mid a_i = a'_i).$$

Intuitively, for a distribution p over action profiles to be a correlated equilibrium, it cannot be worthwhile for player i to deviate from a_i to a'_i if i knows only that $\vec{a}$ is sampled from distribution p (and a_i). Note that all Nash equilibria are correlated equilibria as well. As in Chapter 4, we can extend these definitions to a setting in which coalitions of k players may deviate in a coordinated way:

Definition 22. *In a normal-form game* Γ*, a (mixed) strategy profile* $\vec{\sigma} = (\sigma_1, \ldots, \sigma_n)$ *is a* k-resilient Nash equilibrium *if, for all coalitions* K *of size at most* k*, all* $i \in K$*, and all strategies* $\vec{\sigma}'_K \in \Delta(A_K)$

$$u_i(\vec{\sigma}_K, \vec{\sigma}_{-K}) \geq u_i(\vec{\sigma}'_K, \vec{\sigma}_{-K}).$$

Definition 23. *Given a normal-form game* $\Gamma = (n, A, U)$*, a distribution* $p \in \Delta(A)$ *is a* k-resilient correlated equilibrium *if, for all subsets* $K \subseteq P$ *such that* $|K| \leq k$*, all* $i \in K$*, and all action profiles* $\vec{a}'_K, \vec{a}''_K$ *for players in* K *such that* $p(\vec{a}'_K) > 0$*,*

$$\sum_{\vec{a} \in \Delta(A): \vec{a}_K = \vec{a}'_K} u_i(\vec{a}'_K, \vec{a}_{-K}) p(\vec{a} \mid \vec{a}_K = \vec{a}'_K) \geq \sum_{\vec{a} \in \Delta(A): \vec{a}_K = \vec{a}'_K} u_i(a''_K, a_{-K}) p(\vec{a} \mid \vec{a}_K = \vec{a}'_K).$$

Simply put, a distribution $p \in \Delta(A)$ is a k-resilient correlated equilibrium if no subset of at most k players would be better off by deviating if they knew that the action profile used was sampled according to p (and they knew their components of the profile), even if they could coordinate their deviations.

An *extensive-form game* Γ is a tuple (P, G, M, U, R), where

- $P = \{1, \ldots, n\}$ is the set of players.

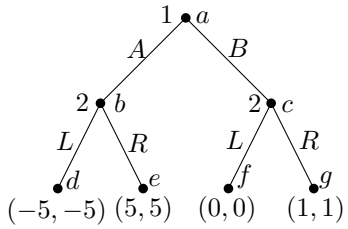
- G is a game tree in which each node represents a possible state of the game and each of the edges going out of a node an action that a player can perform in the state corresponding to that node. (In the sequel, we refer to the edges as actions.)
- M is a function that associates with each non-leaf node of G (we let G° denote the non-leaf nodes) a player in P that describes which player moves at each of these nodes; if $M(v) = i$, then all the edges going out of node v must correspond to actions that player i can play.
- $U = (u_1, \ldots, u_n)$ is an n-tuple of utility functions from the leaves of G to $\mathbb{R}$.
- $R = (\sim_1, \ldots, \sim_n)$ is an n-tuple of equivalence relations on the nodes of G°, one for each player. Intuitively, if $v \sim_i v'$, then nodes v and v' are indistinguishable to player i. Clearly, if $v \sim_i v'$ for some $i \in [n]$, then the set of possible actions that can be performed at v and v' must be identical and $M(v) = M(v')$ (for otherwise, i would have a basis for distinguishing v and v').

The equivalence relation $\sim_i$ induces a partition of G° into equivalence classes called *information sets* (of player i). It is more standard in game theory to define $\sim_i$ as an equivalence relation only on the nodes where i moves. We define it on all non-leaf nodes because if K is a subset of players, we are then able to define $\sim_K$ as the the intersection of $\sim_i$ for $i \in K$. Intuitively, $v \sim_K v'$ if the agents in K cannot distinguish v from v', even if they pool all their information together. We will need the equivalence relation $\sim_K$ for some of our definitions below.

A *(pure) strategy* s_i for player i in an extensive-form game Γ is a function that maps each node v where i moves to an action in v, with the constraint that if two nodes are indistinguishable to i, then s_i must choose the same action on both.

More precisely, if $v \sim_i v'$ then $s_i(v) = s_i(v')$. Each pure strategy profile $\vec{s}$ induces a unique path from the root of the tree to some leaf $\ell \in G \setminus G^\circ$ where, given node v on the path, the next node in the path is defined by $s_{M(v)}(v)$. The payoff of player i given strategy $\vec{\sigma}$ is given by $u_i(\ell)$, although we also write $u_i(\vec{\sigma})$ for simplicity. A *behavioral strategy* for player i allows randomization. More precisely, a behavioral strategy is a function s_i that maps each node v such that $M(v) = i$ to a distribution over i's possible actions at v (again, with the requirement that $s_i(v) = s_i(v')$ if $v \sim_i v'$). We denote by $u_i(\vec{\sigma})$ the expected payoff of player i if players play (behavioral) strategy profile $\vec{\sigma}$.

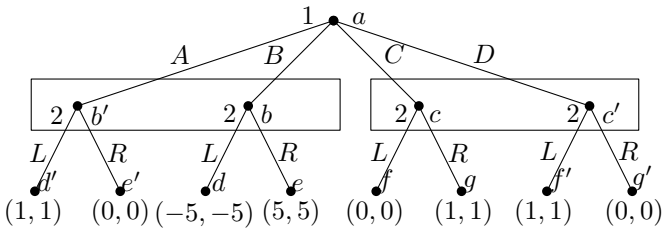
Nash equilibrium is defined for extensive-form games just as it is in strategic-form games. Given a NE $\vec{\sigma}$, the *equilibrium path* consists of the nodes of G that can be reached with positive probability using $\vec{\sigma}$. It is well known that in extensive-form games, Nash equilibrium does not always describe what intuitively would be a reasonable play. For instance, consider the following game for two players in which (p_1, p_2) describes the payoff of players 1 and 2 respectively:



In this case, the strategy profile in which player 1 plays B in a and player 2 plays L in b and R in c is a Nash Equilibrium. However, the reason that it is better for player 1 to play B is that if it plays A then player 2 would play L and they would both get a utility of -5. This means that player 1 is being influenced by an irrational threat, since if 1 plays A it is in player 2's best interest to play R instead of L. In order to avoid these situations, we can extend the notion of Nash

Equilibrium to *Subgame-Perfect Nash Equilibrium*, in which the strategy must be a Nash Equilibrium in all of its subgames as well. In the example above, the only subgame-perfect equilibrium is the one given by $\sigma_1(a) = A$, $\sigma_2(b) = R$ and $\sigma_2(c) = R$.

Unfortunately, subgame-perfect equilibrium may not be well-defined if players have nontrivial information sets. Consider the following game, where b and b' are in the same information set for player 2, as are c and c'.



In b', player 2 would play L; in b, it would play R. For player 2 to decide its move, it must have a belief about the probability of being at each node of the information set, and this belief must be consistent with the strategy $\vec{\sigma}$ being played. For example, if $\sigma_1(a) = B$ if player 2 is in information set $\{b, b'\}$, then 2 would be sure that the true node is b'. We capture the players' beliefs by using a *belief system* b, which is a function from information sets I to a probability distribution over nodes in I. Intuitively, if I is an information set for player i, then b represents how likely it is for i to be in each of the nodes of I. Note that if $\vec{\sigma}$ is *completely mixed*, which means that all actions are taken with positive probability, then b is uniquely determined by Bayes' rule. More generally, we say that a belief system b is *consistent* with a strategy $\vec{\sigma}$ if there exists a sequence $\vec{\sigma}^1, \vec{\sigma}^2, \ldots$ of completely mixed strategies that converges to $\vec{\sigma}$ such that the beliefs induced by Bayes' rule converge to b. Given an extensive-form game $\Gamma = (P, G, M, U, R)$, a strategy profile $\vec{\sigma}$, a belief system b, and an information set I for player i, let $u_i(\vec{\sigma}, I, b)$ denote i's

expected utility conditional on being in information set I and having belief system b, if $\vec{\sigma}$ is played.

Definition 24 ([24])**.** *A pair $(\vec{\sigma}, b)$ consisting of a strategy profile $\vec{\sigma}$ and a belief system b consistent with $\vec{\sigma}$ is a* sequential equilibrim *if, for all players i, all information sets I of player i, and all strategies τ_i for player i,*

$$u_i(\vec{\sigma}, I, b) \geq u_i((\tau_i, \vec{\sigma}_{-i}), I, b).$$

Even though it is standard to define a sequential equilibrium as a pair $(\vec{\sigma}, b)$ consisting of strategy profile and a belief assessment, for convenience we also say that a strategy profile $\vec{\sigma}$ is a sequential equilibrium if there exists a belief assessment b such that $(\vec{\sigma}, b)$ is a sequential equilibrium.

Abraham, Dolev, and Halpern [4] (ADH) generalized sequential equilibrium to allow for deviations by up to k players. To make this precise, we first need to define a generalization of a belief system.

Definition 25. *Let $\Gamma = (P, G, M, U, R)$ be an extensive form game and $K \subseteq P$ be a subset of players. We define the equivalence relation $\sim_K$ by $v \sim_K v'$ iff $v \sim_i v'$ for all $i \in K$. The equivalence classes of $\sim_K$ are called* K-information sets*.*

Definition 26. *A* k-belief system *b in a game Γ is a function that maps each K-information set I such that $K \subseteq P$ and $|K| \leq k$ to a distribution over the nodes in I.*

We can define what it means for a k-belief system to be consistent with $\vec{\sigma}$ just as we did in the case of belief systems. With these definitions in hand, we can find the desired generalization of sequential equilibrium.

Definition 27. *A pair $(\vec{\sigma}, b)$ consisting of a strategy profile $\vec{\sigma}$ and a k-belief system b consistent with $\vec{\sigma}$ is a* k-resilient sequential equilibria *if, for all $K \subseteq P$ with $|K| \leq k$, all K-information sets I, and all strategies τ_K for players in K,*

$$u_i(\vec{\sigma}, I, b) \geq u_i((\tau_K, \vec{\sigma}_{-K}), I, b)$$

for all $i \in K$.

Note that the notions of 1-resilient correlated equilibrium and 1-resilient sequential equilibrium are equivalent to the standard notions of correlated equilibrium and sequential equilibrium, respectively.

Bayesian Games

In all of the previous definitions, the utility of each player is assumed to be common knowledge (*perfect information*). However, this is not always the case. Bayesian games capture this idea by assuming that each player i has a type $t_i \in T_i$ sampled from a distribution $q \in \Delta(T)$, where $T = (T_1, \ldots, T_n)$, and that the utility u_i of i is not only a function of the action profile being played, but also of its type t_i. Formally, a *Bayesian games* is a tuple (P, T, q, A, U), where, as in normal-form games, P, A, and U are the set of players, their actions, and their utility functions, respectively; T is the set of possible type profiles, and q is a distribution in $\Delta(T)$.

A *strategy* in a Bayesian game for player i is a map $\mu_i : T_i \to \Delta(A_i)$. Intuitively, a strategy in a Bayesian game tells player i how to choose its action given its type. A *correlated strategy profile* is a map $\mu : T \to \Delta(A)$. Note that all distributions over strategy profiles can be viewed as correlated strategy profiles, but the converse is not true. For instance, in a game for two players in which $A_i = T_i = \{0, 1\}$ for all i, a correlated strategy profile may consist of both players playing action

$t_1 + t_2 \bmod 2$; this cannot be represented by a strategy profile, since players must independently choose their action given their type.

Since the distribution q is common knowledge, given a (possibly correlated) strategy profile $\vec{\mu}$ in Γ, the expected utility of a member i of a coalition K is

$$u_i(\vec{\mu}) = \sum_{\vec{t}_K \in T_K} q(\vec{t}_K) \sum_{\vec{t}} q(\vec{t} \mid \vec{t}_K) u_i(\vec{\mu}(\vec{t})),$$

where $u_i(\vec{\mu}(\vec{t}))$ denotes the expected utility of player i when an action profile is chosen according to $(\mu(\vec{t}))$. Intuitively, we are assuming that players in K can share their types, which is why we condition on $\vec{t}_K$. This allows us to define k-resilient Nash equilibrium in Bayesian games:

Definition 28. *In a Bayesian Game $\Gamma = (P, T, q, A, U)$, a strategy profile $\vec{\mu} := (\mu_1, \ldots, \mu_n)$ is a k-resilient Nash equilibrium if, for all type profiles $\vec{t} \in T$, all coalitions K with $|K| \leq k$, all maps $\mu'_K : T_K \to \Delta(A_K)$, and all $i \in K$,*

$$u_i(\vec{\mu}) \geq u_i(\vec{\mu}_{-K}, \mu'_K)$$

We can generalize correlated equilibrium to Bayesian games as follows. We can view a correlated equilibrium as a distribution $p \in \Delta(A)$ such that if a trusted mediator samples an action profile $\vec{a} \in p$ and sends action a_i to each player i, it is always better for i to play a_i rather than something else. In a Bayesian game, the mediator instead samples the action profile from a distribution that depends on the type profile. More precisely, suppose that players send their types to a trusted mediator, the mediator samples an action profile $\vec{a}$ from a distribution $\mu(\vec{t})$ that depends on the type profile $\vec{t}$ received, and then sends action a_i to each player i. We say that μ is a *communication equilibrium* if it is optimal for the players to (a) tell their true type to the mediator, and (b) play the action sent by the mediator. The following definitions make this precise.

Definition 29 ([19, 26])**.** *A correlated strategy profile $\mu : T \to \Delta(A)$ is a* communication equilibrium *of Bayesian Game $\Gamma = (P, T, q, A, U)$ if, for all $i \in P$, all $\psi : T_i \to T_i$ and all $\varphi : A_i \to A_i$, we have that*

$$\sum_{\vec{t}_{-i} \in T_{-i}} \sum_{\vec{a} \in A} q(t_{-i}, t_i)\mu(\vec{a} \mid \vec{t}_{-i}, t_i)u_i(\vec{t}_{-i}, t_i, \vec{a}) \geq$$

$$\sum_{\vec{t}_{-i} \in T_{-i}} \sum_{\vec{a} \in A} q(t_{-i}, t_i)\mu(\vec{a} \mid \vec{t}_{-i}, \psi(t_i))u_i(\vec{t}_{-i}, t_i, \vec{a}_{-i}, \varphi(a_i)).$$

Definition 30. *A correlated strategy profile $\mu : T \to \Delta(A)$ is a* k-resilient communication equilibrium *of Bayesian Game $\Gamma = (P, T, q, A, U)$ if, for all $K \subseteq P$, all $i \in K$, all $\psi : T_K \to T_K$, and all $\varphi : A_K \to A_K$, we have that*

$$\sum_{\vec{t}_{-K} \in T_{-K}} \sum_{\vec{a} \in A} q(t_{-K}, \vec{t}_K)\mu(\vec{a} \mid \vec{t}_{-K}, \vec{t}_K)u_i(\vec{t}_{-K}, \vec{t}_K, \vec{a}) \geq$$

$$\sum_{\vec{t}_{-K} \in T_{-K}} \sum_{\vec{a} \in A} q(t_{-K}, \vec{t}_K)\mu(\vec{a} \mid \vec{t}_{-K}, \psi(\vec{t}_K))u_i(\vec{t}_{-K}, \vec{t}_K, \vec{a}_{-K}, \varphi(\vec{a}_K)).$$

In Definition 30, the functions ψ and φ represent possible deviations for players in K; ψ describes how they might lie to the mediator about their type (by sending type profile $\phi(\vec{t}_K)$ instead of their actual type profile $\vec{t}_K$), while φ represents how the might deviate in what they do (playing strategy profile $\varphi(\vec{a}_K)$ instead of the strategy profile $\vec{a}_K$ suggested by the mediator). Definition 30 says that μ is a k-resilient communication equilibrium if none of these deviations is profitable for the players in K.

We can combine the notions of extensive-form game and Bayesian game in the obvious way to get *extensive-form Bayesian games*: we start with an extensive form game, add a type space T and a commonly known distribution q on T, and then have the utility function depend on the type profile as well the leaf reached. We leave formal details to the reader.

5.1.1 Communication games

As in Section 4.1, given a normal-form game $\Gamma = (n, A, U)$ or a Bayesian game $\Gamma = (P, T, q, A, U)$, consider an extension Γ_{CT} in which players can communicate among themselves before playing an action in Γ. We call this extension a *cheap-talk game*. More precisely, in a cheap-talk game Γ_{CT}, players are able to exchange messages and play an action in Γ (although they may play an action in Γ at most once). The payoff of each player is defined by U, given the action profile played in Γ.

We can view cheap-talk games as possibly infinite extensive-form games in which the nodes are the *global histories* $\vec{h} = (h_1, \ldots, h_n)$ of the players, and the edges are either performing an internal operation, sending messages or playing an action in the underlying game. A global history $\vec{h}$ is just a tuple of local histories, where the local history h_i of player i contains all internal computations (which may include randomization), all messages sent by i and received by i, along with their time stamps (in the synchronous case) or the number of times that i was scheduled since the beginning of the game (in the asynchronous case). Two global histories $\vec{h} = (h_1, \ldots, h_n)$ and $\vec{h}' = (h'_1, \ldots, h'_n)$ are indistinguishable by player i if $h_i = h'_i$. Analogously, $\vec{h}$ and $\vec{h}'$ are indistinguishable by a coalition K of players if $\vec{h}_K = \vec{h}'_K$.

We say that a strategy $\vec{\sigma}$ in a communication game Γ_{CT} *implements* a strategy τ in Γ if the outcome induced by $\vec{\sigma}$ in Γ is identical to that induced by τ. If Γ is a normal-form game, the outcome induced by $\vec{\sigma}$ is the distribution over action profiles in Γ that results when $\vec{\sigma}$ is played in Γ_{CT}. If Γ is a Bayesian game, the outcome is a map from type profiles to distributions over action profiles (i.e. a correlated strategy profile).

5.2 Main results

Given a normal-form or Bayesian game Γ, let $\Gamma_{CT,syn}$ be the synchronous cheap-talk extension of Γ and let $\Gamma_{CT,asyn}$ be its asynchronous extension. Let $\Gamma^d_{CT,syn}$ and $\Gamma^d_{CT,asyn}$ be the synchronous and asynchronous cheap-talk extensions of Γ in which players can also communicate with a trusted mediator d. If Γ is a normal-form game, let $CE_k(\Gamma)$ denote the set of possible distributions over action profiles induced by a k-resilient correlated equilibria in Γ; if Γ is a Bayesian game, let $Com_k(\Gamma)$ denote the set of possible maps from type profiles to distributions over action profiles induced by k-resilient communication equilibria in Γ; if Γ_{CT} is a communication game, let $SE_k(\Gamma_{CT})$ denote the set of possible strategies in Γ induced by k-resilient sequential equilibria in Γ_{CT} (note that if Γ is a normal-form game, $SE_k(\Gamma_{CT})$ is a set of distributions over action profiles, while if Γ is a Bayesian game, $SE_k(\Gamma_{CT})$ is a set of maps from type profiles to distributions over action profiles).

In the rest of this chapter, we consider only equilibria in which each action profile is sampled with rational probability. The reason for this restriction is that it is a standard assumption in distributed systems that messages have finite length and that players cannot perform arbitrarily large operations. This means that agents cannot encode real numbers into messages and they cannot perform operations on real numbers, in general. If we relax these constraints and allow players to send messages and operate with real numbers, our results can be extended to all equilibria inside the convex hull of rational equilibria, using techniques due to Gerardi [20]. These techniques are described in Section 5.5.

Theorem 15. *If $\Gamma = (P, T, q, A, U)$ is a Bayesian game for n players, then $SE_k(\Gamma_{CT,syn}) = SE_k(\Gamma^d_{CT,syn})$ for all k such that $n > 3k$.*

Theorem 16. *If* $\Gamma = (P, T, q, A, U)$ *is a Bayesian game for* n *players, then* $SE_k(\Gamma_{CT,asyn}) = SE_k(\Gamma^d_{CT,asyn})$ *for all* k *such that* $n > 4k$*, with both the default move and AH approaches.*

If Γ is a normal-form game, the sets $SE_k(\Gamma^d_{CT,syn})$ and $SE_k(\Gamma^d_{CT,syn})$ can be easily described (Gerardi [20] shows this result for the synchronous case with $k = 1$).

Proposition 22. *If* $\Gamma = (P, A, U)$ *is a normal-form game for* n *players, then* $SE_k(\Gamma^d_{CT,syn}) = SE_k(\Gamma^d_{CT,asyn}) = CE_k(\Gamma)$ *for all* $k \leq n$*.*

Proof. Clearly, the distribution over actions induced by k-resilient sequential equilibrium strategy $\vec{\sigma}$ in $\Gamma^d_{CT,syn}$ (or $\Gamma^d_{CT,asyn}$) is also a k-resilient correlated equilibrium of Γ, for otherwise there exists a coalition K of at most k players that can increase their utility by deviating from $\vec{\sigma}$ by playing a different action in the underlying game. For the opposite inclusion, observe that all k-resilient correlated equilibria p of Γ can be easily implemented with a mediator in both synchronous and asynchronous settings: the mediator samples an action profile $\vec{a}$ following distribution p and gives a_i to each player i. Then each player i plays whatever is sent by the mediator, and if it never receives an action during the communication phase, it plays the best response assuming that all other players play their part of $\vec{a}$ and $\vec{a}$ is sampled from p. □

Theorems 15 and 16 together with Proposition 22 imply the following corollaries:

Corollary 2. *If* $\Gamma = (P, A, U)$ *is a normal-form game for* n *players, then* $SE_k(\Gamma_{CT,syn}) = CE_k(\Gamma)$ *for all* k *such that* $n > 3k$*.*

Corollary 3. *If* $\Gamma = (P, A, U)$ *is a normal-form game for* n *players, then* $SE_k(\Gamma_{CT,asyn}) = CE_k(\Gamma)$ *for all* k *such that* $n > 4k$*, with both the DM and AH approaches.*

For Bayesian games, we have the following result.

Proposition 23. *If* $\Gamma = (P, T, q, A, U)$ *is a Bayesian game for* n *players, then* $SE_k(\Gamma^d_{CT,syn}) = Com_k(\Gamma)$ *for all* $k \leq n$*.*

The proof is analogous to that of Proposition 22, although we outline a more detailed construction in Section 5.3. With this proposition we can easily describe $SE_k(\Gamma_{CT,syn})$ as well:

Corollary 4. *Let* $\Gamma = (P, T, q, A, U)$ *be a Bayesian game for* n *players, then* $SE_k(\Gamma_{CT,syn}) = Com_k(\Gamma)$ *for all* k *such that* $n > 3k$*.*

Unfortunately, we do not believe that there is a simple description of $SE_k(\Gamma^d_{CT,asyn})$. Asynchrony seems to make things much more complicated. Reasoning analogous to that of the proof of Proposition 22 shows that $SE_k(\Gamma^d_{CT,asyn}) \subseteq Com_k(\Gamma)$, but it is still an open problem if all k-resilient communication equilibria of Γ can be implemented with a mediator in asynchronous systems. In synchronous systems, the mediator can assign a default $\perp$ type to players that didn't send a message. However, in asynchronous systems, the mediator cannot distinguish between players that didn't send their type and players that send it but are being delayed by the scheduler. Nevertheless, if players can somehow punish those that never send a type, the same construction used in the synchronous case would suffice for the asynchronous case as well. More precisely, with the AH approach, all $\mu \in Com_k(\Gamma)$ can be implemented in $\Gamma^d_{CT,asyn}$ if Γ has a *k-punishment equilibrium* with respect to μ.

Definition 31 ([13, 1]). *If $\Gamma = (P, T, q, S, U)$ is a Bayesian game for n players and $\mu \in Com_k(\Gamma)$, then a* k-punishment equilibrium *with respect to μ is a strategy profile $\vec{\tau}$ such that*

(a) $\vec{\tau}$ is a k-resilient Nash equilibrium of Γ.

(b) If at least $n - k$ players play their part of $\vec{\tau}$, all players are guaranteed to get a lower expected payoff than the expected utility of μ.

Intuitively, a punishment equilibrium with respect to μ is a strategy that is a k-resilient Nash equilibrium which, if played by enough players, results in all players being worse off than they would be with μ. If a punishment equilibrium exists, then players can implement communication equilibria in asynchronous systems:

Proposition 24. *If $\Gamma = (P, T, q, S, U)$ is a Bayesian game, $\mu \in Com_k(\Gamma)$, and Γ has a k-punishment equilibrium $\vec{\tau}$ with respect to μ, then $\mu \in SE_k(\Gamma^d_{CT,asyn})$ with the AH approach or with the default move approach if the default move is $\vec{\tau}$.*

Proof. Given μ, consider the strategy $\vec{\sigma}_{CT}$ in $\Gamma^d_{CT,asym}$ where each player sends its type to the mediator, the mediator waits until it receives the type of all players (note that it may never receive the types of all players), then computes $\vec{a} := \mu(\vec{t})$ and sends a_i to each player i. If a player receives an action from the mediator, it plays that action. If player i never receives an action from a mediator, it plays τ_i. It is easy to check that $\vec{\sigma}_{CT}$ is a k-resilient sequential equilibrium: since μ' is a k-punishment equilibrium with respect to μ, it is never in the interest of any player not to send its input. Since $\mu \in Com_k(\Gamma)$, it is also not in the interest of i to lie about its input or to play an action different from what the mediator sends. Moreover, if a player i does not receive an action to play, it will believe that no one else did, and since $\vec{\tau}$ is a k-resilient Nash equilibrium, playing τ_i is optimal. □

5.2.1 Outline of the proofs

The proof of Theorems 15 and 16 is divided in two parts. We first present a relatively simple strategy $\vec{\sigma}$ in Γ_{CT} that implements the desired communication equilibrium μ and tolerates deviations of up to k players. Moreover, with $\vec{\sigma}$, no subset of at most k players can sabotage the joint computation or learn anything other than what they were originally supposed to learn. These security properties of $\vec{\sigma}$ guarantee that it is a k-resilient Nash equilibrium.

The next step in the proof is extending $\vec{\sigma}$ to a k-resilient sequential equilibrium $\vec{\sigma}^{seq}$. The key idea for doing this is showing that there exists a belief system b such that all coalitions K of at most k players believe that all players that deviated from the protocol did so by sending inappropriate messages only to players in K; that is, each coalition K of size at most k believes that if $i, j \notin K$, then i and j always send the messages that they are supposed to send according to $\vec{\sigma}$ to each other. If players have belief system b, then all coalitions K with $|K| \leq k$ believe, regardless of their local history, that the remaining players will always compute their part of the communication equilibrium correctly, since $\vec{\sigma}$ tolerates deviations by up to k players and all the players not in K get the same messages that they would in K (and thus are not experiencing deviations by more than k players). Moreover, in the construction of $\vec{\sigma}$, if less than k players deviate, all players eventually receive enough information to compute their own action, even if they don't take part in the computation. These two properties are enough to guarantee that it is never to the benefit of players in K to deviate during the communication phase, since (they believe that) nothing they can do will affect what the remaining players play in the underlying game, and they can't learn any additional information.

Constructing a strategy profile that implements a given communication equi-

librium with a mediator is in general nontrivial, since players must not only jointly compute the action that the mediator sends to each player, but it should also be the case that each player i does not learn anything form the communication phase other than its own action a_i since if i was able to get such information, i might be tempted to deviate (since it might have more information about what the utility of deviating would be). However, players can achieve the desired security properties using the VSS and CC protocols described in Section 2.2.1.

5.3 Proof of Theorem 3

5.3.1 Constructing a k-resilient Nash Equilibrium

By definition, $SE_k(\Gamma_{CT,syn}) \subseteq SE_k(\Gamma_{d,syn})$. It thus remains to show that a k-resilient sequential equilibrium with a mediator can be implemented by a k-resilient sequential equilibrium in the communication game. As we noted in Section 5.2, $SE_k(\Gamma_{d,syn}) = Com_k(\Gamma)$, and thus $SE_k(\Gamma_{CT,syn}) \supseteq SE_k(\Gamma_{d,syn})$ reduces to show that all k-resilient communication equilibria μ of Γ can be implemented with cheap talk without a mediator. We begin by constructing a strategy profile $\vec{\sigma}$ in $\Gamma_{CT,syn}$ that is a k-resilient Nash Equilibrium and induces the same correlated strategy profile as μ.

Given μ, consider the canonical protocol $\vec{\sigma}_{can}$ for n players and a mediator in which each player i sends its type t_i to the mediator, the mediator then computes the type profile $\vec{t}$ of the players using the messages received, and replacing the types of players that didn't send theirs by a default type $\perp$. The mediator then samples $\vec{a} \in \Delta(A)$ according to the distribution $\mu(\vec{t})$ and sends a_i to each player

i. If i sent its true type at the beginning of the game and receives an action a_i from the mediator, it plays a_i. Otherwise, it plays the best response assuming that the remaining players are playing their part of $\mu(\vec{t})$ (note that the best response is well-defined since both μ and the distribution q of type profiles are common-knowledge). It is straightforward to check that $\vec{\sigma}_{can}$ induces the same correlated strategy profile as μ, and that $\vec{\sigma}_{can}$ is a k-resilient sequential equilibrium. The details are left to the reader.

The idea for constructing $\vec{\sigma}$ is that instead of having a mediator that receives the types of the players and then computes μ, the players perform these tasks by themselves using VSS and CC to help them simulate the mediator: each player i, instead of sending its type t_i to the mediator, it shares it between all players using a k-resilient VSS implementation. After the necessary amount of rounds to perform VSS, players compute $(a_1, \ldots, a_n) := \mu(t_1, \ldots, t_n)$ with CC, using as input the shares of each of the types t_i. Each player i then sends to each player j its share of a_j. That way, each player i can compute its own action a_i without learning anything about the other players' actions.

Note that, since $\vec{a}$ is sampled from $\mu(\vec{t})$, players are required to randomize in a coordinated way. The problem of jointly sampling an element from a distribution with rational probabilities over a finite set can be reduced to the problem of sampling an integer from $[N] := \{1, \ldots, N\}$ for some N, as the following proposition shows.

Proposition 25. *Given a correlated strategy profile μ with rational parameters, there exists an integer N and a function $\mu^* : T \times [N] \to \Delta(S)$ such that, if $X_{\vec{t}}$ is the distribution over action profiles induced by $\mu^*(\vec{t}, r)$ when r is uniformly sampled from $[N]$, then $X_{\vec{t}}$ and $\mu(\vec{t})$ are the same for all $\vec{t} \in T$.*

Proof. Fix an integer N such that, for all $\vec{t}$, all probabilities in $\mu(\vec{t})$ are integer multiples of $1/N$. For each $\vec{t}$, partition N into subsets $C_1, C_2, \ldots, C_m$ such that $|C_i|$ is proportional to the ith probability in $\mu(\vec{t})$. Let $\mu^*(\vec{t}, r)$ output the ith action profile in $\mu(\vec{t})$ if $r \in C_i$ for some i, It is straightforward to check that μ^* satisfies the desired properties. □

Players can thus compute the desired action profile using a deterministic function μ^* given that they are able to jointly sample an integer $r \in N$ uniformly at random. We present next a formal construction of the protocol $\vec{\sigma}$ described above:

Phase 1: Each player i generates a random number r_i between 1 and N uniformly at random. Then i shares r_i and its type t_i using a k-resilient VSS procedure. If a player i does not share t_i or r_i this way, t_i and r_i are assumed to be a default type $\perp \in T_i$ and 0 respectively.

Phase 2: For each player j, let r_j^i be i's share of r_j and t_j^i be i's share of t_j. Each player i initiates a k-resilient CC of the functions $\mu_1^*, \ldots, \mu_n^*$, where $\mu_j^*(\vec{t}, \vec{r})$ is the jth component of $\mu^*(\vec{t}, r)$ and $r := r_1 + \ldots + r_n \bmod N$. Player i's inputs for each of these CC instances are precisely the shares $r_1^i, \ldots, r_n^i$ and $t_1^i, \ldots, t_n^i$ computed in Phase 1.

Phase 3: For each player i, let o_j^i be i's output of the CC instance of μ_j^* (which is i's share of $\mu_j^*(\vec{t}, \vec{r})$). Each player i sends each share o_j^i to each player j.

Phase 4: If i shared its true type in Phase 1 and i receives at least $n-k$ shares o_i^j from different players such that every subset of size $k+1$ of those shares defines the same secret a_i, i plays a_i and terminates.

Phase 5: If i did not terminate in Phase 4, it plays a best response given its local

history, assuming that all the remaining players play their part of the communication equilibrium.

Note that the behavior of the players in $\vec{\sigma}$ is not always explicitly defined. For instance, if a player i doesn't successfully terminate Phase 2 (for instance, because other players deviated), the protocol does not say now to proceed. If the protocol does not explicitly say what to do, then player i simply does nothing.

Proposition 26. *If $n > 3k$, $\vec{\sigma}$ is a k-resilient Nash equilibrium and induces the same correlated strategy profile as μ.*

Proof. Clearly $\vec{\sigma}$ induces the same correlated strategy profile as μ by construction. The fact that $\vec{\sigma}$ is a k-resilient NE follows directly from the properties of k-resilient VSS and k-resilient CC: no matter what a coalition of at most k players does, the remaining $n-k$ players are guaranteed to terminate all their VSS and CC instances correctly. Since honest players choose r_i uniformly at random, the number $r := r_1 + \ldots + r_n$ is also randomly distributed in $[N]$ regardless of what a coalition of at most k players decides to share. This guarantees that the actions $a_1, \ldots, a_n$ encoded by the shares $(o_1^1, \ldots, o_1^n), \ldots, (o_n^1, \ldots, o_n^n)$ of honest players follow the same distribution as $\mu(\vec{t})$. The correctness of the output of each player follows from the fact that if a player i waits until receiving at least $2k+1$ shares that define the same secret a_i, at least $k+1$ of those shares were sent by honest players and thus define the correct value a_i (note that if $n > 3k$, all players are guaranteed to be able to reconstruct their action a_i in Phase 4 even if a coalition of k players deviates, since at least $2k+1$ players are honest). To complete the proof, note that the secrecy properties of the k-resilient VSS and CC guarantee that no coalition of at most k players can learn anything besides their own actions, and thus that playing the action a_i computed in Phase 4 is optimal. □

5.3.2 Constructing a k-resilient Sequential Equilibrium

Proposition 26 shows that, for all $\mu \in Com_k(\Gamma)$, there exists a strategy $\vec{\sigma}$ in Γ_{CT} such that $\vec{\sigma}$ is a k-resilient Nash Equilibrium that implements μ. Our aim is to extend $\vec{\sigma}$ to a k-resilient sequential equilibrium. The difficulty of constructing such an extension is due to the fact that the k-resilience of VSS and CC guarantees that no coalition of at most k players can prevent the remaining players from carrying out the VSS and CC computations if no other player deviates (which is all that is needed to show that the construction gives a k-resilient Nash Equilibrium). However, it is not clear what the coalition can accomplish starting at a point off the equilibrium path where they detect that other players are deviating as well.

For instance, consider a normal-form game Γ for n players such that the set of actions for each player is $[n]$. Given an action profile $\vec{a}$, if at least $n - k$ players played a different action, all players get a payoff of 0, and otherwise all players get a payoff of 1. Consider the k-resilient correlated equilibrium p consisting of all possible permutations of $[n]$, which means that the payoff of each player if an action profile in the support of p is 0. Consider the following situation, where players play the protocol $\vec{\sigma}$ described in Section 5.3.1: if a coalition K of at most k players is being honest but they detect an inconsistency between their local histories, it must be the case that one of the remaining $n - k$ players deviated. Since k-resilient implementations of VSS and CC can guarantee the correctness of the outputs only when at most k players deviate, it is certainly possible that there exists a strategy for players in K such that, from this point on, they get a payoff of 1. If this is the case, then we do not have a sequential equilibrium.

Our approach to implementing μ with a k-resilient sequential equilibrium is to show that there exists a k-belief system such that all coalitions K of size at most

k believe, regardless of their local history, that

(a) The remaining players will successfully complete their part of $\vec{\sigma}$, with or without the help of players in K.

(b) Players in K cannot deduce more information about the input profile $\vec{x}$ and action profile $\vec{a}$ being played other than their own actions $\vec{a}_K$ and their inputs $\vec{x}_K$. (and the fact that $\vec{a}$ is sampled from $\mu(\vec{x})$).

(c) If a player $i \in K$ receives at least $2k+1$ shares as required in Phase 4, then these shares are all correct.

It is important to stress that property (b) does in fact depend on the beliefs of players in K: If some player i in K receives a message $\perp$ from a player j that was not supposed to be sent in the equilibrium path and i believes that j sends that message only when j has type t_j (whether or not this is actually the case), then i will believe that it has acquired important information when it receives $\perp$, and adjust its play accordingly. However, if i believes that j chose that message uniformly at random, then i will not adjust its play. The following proposition formalizes properties (a), (b), and (c), and shows that is these properties are satisfied, then $\vec{\sigma}$ is k-sequentially rational (i.e., it is never in the interest of coalitions of size at most k to deviate from $\vec{\sigma}$).

Proposition 27. *Suppose b is a belief system compatible with $\vec{\sigma}$ such that the following properties hold:*

(a) For all histories $\vec{h}_K$ of players in K and all histories that occur with positive probability in $b(\vec{h}_K)$, we have that all players not in K share their true input in Phase 1 and terminate Phases 1,2,3 and 4; the shares computed by these

players at the end of each of these phases define a unique secret; and none of these players gets to Phase 5, regardless of what players in K do. Moreover, the probability that in some history $\vec{h}$ sampled from $b(\vec{h}_K)$ players not in K play action profile $\vec{a}_{-K}$ conditioned on the input profile $\vec{x}$ shared in Phase 1 is the same as in μ.

(b) if $b'(\vec{h}_K)$ is the function that returns the distribution over action profiles $\vec{a}_K$ encoded by the shares of the players not in K in $b(\vec{h}_K)$ (which define a unique secret, by part (a)), and the players in K have type profile $\vec{t}_K$, then there exists a function f_K such that $f_K(\vec{t}_K, b'(\vec{h}_K))$ and $b(\vec{h}_K)$ are identically distributed.

(c) For all histories $\vec{h}_K$ of players in K, and all histories h that occur with positive probability in $b(\vec{h}_K)$, if a player $i \in K$ terminates Phase 4 in h, then the shares received were the actual shares computed by the players at the end of Phase 2.

Then, $(\vec{\sigma}, b)$ is a k-resilient sequential equilibrium.

As we said, properties (a), (b), and (c) of Proposition 27 formalize the properties (a), (b), and (c) discussed above. Property (a) says that players outside of K have no trouble terminating phases 1, 2, 3, and 4, regardless of the strategy used by players in K. Moreover, it states that their outputs are *correct* (i.e., that their shares are consistent and that the actions that they reconstruct follow the same distribution as μ). Property (b) says that everything encoded in the local history of players in K can be deduced from the types and beliefs of the players in K about the action profile $\vec{a}_K$, which means that they essentially do not learn anything else. Property (c) says that if players in K terminate the protocol at Phase 4 (i.e., they manage to reconstruct the action profile $\vec{a}_K$), then they believe

that none of the players whose share was used to reconstruct $\vec{a}_K$ lied about their share.

Proposition 27. If b satisfies property (a), then at all nodes in the game tree (including nodes off the equilibrium path), not matter what the players in K do, the remaining players will eventually terminate their part of $\vec{\sigma}$ and thus sample an action profile $\vec{a} \in \mu(\vec{t})$ and play their part $\vec{a}_{-K}$. Property (a) also guarantees that by the end of Phase 4 (and the end of the communication phase), players not in K will send the shares of $\vec{a}_K$ to players in K, which means that before the last round, players in K believe that they will always learn $\vec{a}_K$. Property (b) guarantees that players in K won't learn anything besides their own actions. Properties (a) and (b) together imply that players in K believe that they cannot prevent the remaining players from playing their part of an action profile $\vec{a}$ sampled with a probability defined by μ, and that players in K will eventually learn $\vec{a}_K$ and nothing else. Then, since μ is a k-resilient sequential equilibrium, it is always a best response of the players in K to share their own type in Phase 1. This shows that it is never in the interest of players in K to deviate during the communication phase. It remains to show that the action played in Phase 4 is optimal. By construction, this reduces to showing that if a coalition K of players with $|K| \leq k$ shared their true type in Phase 1 and were able to compute their actions a_i in Phase 4, it is optimal for them to play a_i (note that otherwise i is already best responding given that b satisfies properties (a) and (b)). This follows from the fact that, by properties (a) and (b), players in K believe that all remaining players computed $\vec{a}$ and thus played $\vec{a}_{-K}$ as described in $\vec{\sigma}$. Moreover, by property (c), players in K believe that the action profile $\vec{a}_K$ computed in Phase 4 is correct (in the sense that it is correlated with $\vec{a}_{-K}$ in the same way as in $\vec{\mu}$). Since μ is a k-resilient communication equilibrium, it is then always in the interest of players in K to play $\vec{a}_K$. □

Proposition 27 shows that Theorem 3 reduces to finding a belief system b compatible with $\vec{\sigma}$ that satisfies properties (a), (b), and (c). We start by showing how we can construct a belief system that satisfies (a) and (b) for a fixed subset K with $|K| \leq k$. Consider the sequence $\vec{\sigma}^1, \vec{\sigma}^2, \ldots$ of strategy profiles in which player i proceeds as follows with $\vec{\sigma}_i^m$: Given local history h_i, i does exactly what it would have done with $\vec{\sigma}$, except that if it is supposed to send a message msg to another player j, if $j \in K$, with probability $1/m$ it replaces msg by a random message msg' sampled according to some fixed distribution that gives all messages positive probability, and sends msg' instead. If $j \notin K$, i replaces msg with msg' with probability m^{-m}. That is, players play as in $\vec{\sigma}$, except that they may *lie* to players in K with small probability, and to players outside of K with a much smaller probability. By construction, each $\vec{\sigma}^m$ is completely mixed and the belief b induced by the limits of the beliefs in $\vec{\sigma}^1, \vec{\sigma}^2, \ldots$ satisfies the properties (a) and (b) described above for the set K. To see this, note that property (a) is satisfied because players in K believe that the remaining $n - k$ players are being truthful between themselves (note that the probability of anyone lying to a player not in K is negligible) and thus, since the VSS and CC implementations are k-resilient, that they can carry out all the necessary computations without the help of players in K. Moreover, since players in K believe that if they get a message incompatible with $\vec{\sigma}$ (i.e., a message that could not have been sent in $\vec{\sigma}$, given their joint histories), then they are getting a message drawn from some fixed definition, they learn no useful information about the action profile being played by getting this message. So, we have property (b).

This example shows that if a subset K of players of size at most k believe that they are the only ones being lied to, then properties (a) and (b) hold for that subset K. This motivates the following definitions:

Definition 32. *Player i* lies relative to $\vec{\sigma}$ in global history $\vec{h}$ *if, in $\vec{h}$, it sends a message to a player $j \neq i$ that is incompatible with $\vec{\sigma}$. A subset K of players is* truthful relative to $\vec{\sigma}$ in $\vec{h}$ *if, in $\vec{h}$, no player in K ever lies to another player in K.*

Note that, given a global history $\vec{h}$, we can determine whether a message sent by i to j is compatible with $\vec{\sigma}$ even if $\sigma_i(h_i)$ gives a distribution over messages. This is because h_i includes all of i's randomization up to the point where it is supposed to send a message, so we can tell which message actually should be sent.

Definition 33. *Given a k-belief assessment b and the local histories $\vec{h}_K$ of a subset K of players, a global history $\vec{h}$ is b-*consistent *with $\vec{h}_K$ if the probability according to b that players have global history h conditional on players in K having history $\vec{h}_K$ is non-zero. A k-belief assessment b is k-*paranoid *if, for all subsets $K \subseteq [n]$ with $|K| \leq k$ and all histories $\vec{h}_K$ of players in K, in all histories $\vec{h}$ that are b-consistent with $\vec{h}_K$, the set $\bar{K}$ is truthful (where $\bar{K}$ is the complement of K).*

Intuitively, a k-belief assessment is k-paranoid if all subsets of at most K players believe, regardless of their local history, that they are the only ones being lied to. Our earlier discussion provides a proof of the following proposition.

Proposition 28. *Let $\vec{\sigma}$ be the protocol presented in Section 5.3.1. If b is a k-paranoid belief system consistent with $\vec{\sigma}$, then b satisfies properties (a) and (b).*

Additionally, we have the following:

Proposition 29. *Let $\vec{\sigma}$ be the protocol presented in Section 5.3.1. If b is a k-paranoid belief system consistent with $\vec{\sigma}$, then b satisfies property (c).*

Proof. If b is k-paranoid, then, for all sets K with $|K| \leq k$, since all primitives are k-resilient it means that, regardless of their local history, in their beliefs the remaining players always manage to compute the correct share of all computations and should theoretically send those shares to players in K by the end of Phase 3. Thus, players in K will always believe that the secret that is defined by the largest amount of shares is the correct one (the one that involves less deviations). Since at most one secret can be defined by (at least) $n - k$ shares in a k-resilient VSS/CC implementation, if they terminate Phase 4, they believe that the actions reconstructed are correct as in property (c). □

Thus, by Proposition 27, Theorem 3 reduces to find a k-paranoid belief assessment b that is consistent with $\vec{\sigma}$. The first candidate for generating such a belief is the sequence of strategy profiles $\vec{\sigma}^1, \vec{\sigma}^2, \ldots$ that we mentioned earlier in which, in the strategy profile $\vec{\sigma}^m$, each player lies to each other with probability $1/m$, except that here each player lies to all players with probability $1/m$ (as opposed to our previous example, in which they lied with probability $1/m$ only to players in a fixed subset K). We might hope that if players detect that they received inappropriate messages, they would assume that they are the only ones being lied to, since lies are highly unlikely. For instance, if a player i receives four inconsistent messages, we would hope that i would believe that these are the only lies. Unfortunately, this is not necessarily the case. If an earlier lie by some other player could result in players following $\vec{\sigma}$ sending these messages, then i would instead believe that there was only one lie. To prevent this, we want players to believe that earlier lies are much less likely than later lies. It turns out that if we do this, then we can get the result that we want.

Proposition 30. *For all strategies $\vec{\sigma}$ in $\Gamma_{CT,syn}$, there exists a k-paranoid belief assessment b that is consistent with $\vec{\sigma}$.*

Proof. Consider the sequence $\vec{\sigma}^1, \vec{\sigma}^2, \vec{\sigma}^3, \ldots$ of strategy profiles where in $\vec{\sigma}^n$ players act exactly as they do in $\vec{\sigma}$, except that whenever i would send a message msg to another player j, with probability $m^{-\left((2n)^{-r}\right)}$ (where r is the current round of communication), i replaces this message by a message msg' chosen according to some fixed distribution and sends msg' instead. Intuitively, these are protocols in which each player may lie to each other with a probability that (greatly) increases with the round number and that decreases with m. It is straightforward to check that the belief assessments induced by this sequence of strategy profiles converges to $\vec{\sigma}$. Let b be the belief assessment induced by the sequence. We want to show that b is k-paranoid. We need some preliminary lemmas.

Given a global history $\vec{h}$ of $\vec{\sigma}$, let $L(\vec{h})$ be the sequence $(\ell_1, \ell_2, \ldots, \ell_R)$, where R is the total number of communication rounds in $\vec{h}$ and ℓ_r is the number of lies sent at round r in $\vec{h}$.

Lemma 13. *If P^m is the probability on global histories induced by $\vec{\sigma}^m$, and $\vec{h}$ and $\vec{h}'$ are global history profiles such that $L(\vec{h}) < L(\vec{h}')$ in lexicographical order (the smaller sequence is the one that has the smaller value in the first position in which they differ), then $\lim_{m\to\infty} \frac{P^m(\vec{h}')}{P^m(\vec{h})} = 0$.*

Proof. If $L(\vec{h}) = (\ell_1, \ell_2, \ldots, \ell_R)$, then

$$\log(P^m(\vec{h})) = -\log(m)\left(\sum_{r=1}^{n} \ell_r (2n)^{-r}\right) + \Theta(1).$$

Thus, if $L(\vec{h}) = (\ell_1, \ell_2, \ldots, \ell_R)$ and $L(\vec{h}') = (\ell'_1, \ldots, \ell'_{R'})$, we must show that if $L(\vec{h}) < L(\vec{h}')$, then $\sum_{r=1}^{R'} \ell_r (2n)^{-r} < \sum_{r=1}^{R'} \ell'_r (2n)^{-r}$, which is equivalent to showing that $\sum_{r=1}^{\max(R,R')} (\ell'_r - \ell_r)(2n)^{-r} > 0$ (assuming that the elements beyond the end of the sequences are 0).

Since each player sends exactly one message to each other player, we have that $0 \leq \ell_i \leq n$. Therefore, if r^* is the first round such that $\ell'_r \neq \ell_r$, we have that

$$\sum_{r=1}^{\max(R,R')} (\ell'_r - \ell_r)(2n)^{-r} \geq (2n)^{-r^*} - n \sum_{r=r^*+1}^{\max(R,R')} (2n)^{-r} > 0,$$

where the last inequality comes from the fact that

$$n \sum_{r>0} (2n)^{-r} < n \sum_{r>0} 2^{-r} n^{-1} = 1.$$

□

With this lemma, we are able to prove our main result, which is implied by the following:

Lemma 14. *If $|K| \leq k$ and $\vec{h}^*_K$ is a local history of the players in K, then for every global history profile $\vec{h}$ such that $\bar{K}$ is not truthful in $\vec{h}$,*

$$\lim_{m \to \infty} P^m(\vec{h} \mid \vec{h}^*_K) = 0$$

Proof. Suppose that for some global history $\vec{h}$, $P^m(\vec{h} \mid \vec{h}^*_K) > 0$ and $\bar{K}$ is not truthful in $\vec{h}$. Since

$$P^m(\vec{h} \mid \vec{h}^*_K) = \frac{P^m(\vec{h})}{\sum_{\vec{h}' : \vec{h}'_K = \vec{h}^*_K} P^m(\vec{h}')},$$

by Lemma 13, this result reduces to finding a global history $\vec{h}'$ such that $\vec{h}'_K = \vec{h}_K$ and $L(\vec{h}') < L(\vec{h})$. This history can be constructed as follows: suppose that r is the first round in $\vec{h}$ such that a player $i \in \bar{K}$ lies to another player $j \in \bar{K}$. Consider a history $\vec{h}'$ in which players in K perform the same internal computations and send exactly the same messages as in $\vec{h}$, but players in $\bar{K}$ only do so until round $r-1$. In round r, players in $\bar{K}$ perform exactly the same internal computations as in $\vec{h}$ and send the same messages to players in K, but they send the correct messages to other players in K. From round $r+1$ on, they perform arbitrary internal

computations and send to players in K exactly the same messages as in $\vec{h}$. Clearly, by construction, $\vec{h}'_K = \vec{h}_K$. Let $L(\vec{h}) = (\ell_1, \ell_2, \ldots, \ell_R)$ and $L(\vec{h}') = (\ell'_1, \ell'_2, \ldots, \ell'_R)$. Then, $\ell_i = \ell'_i$ for all $i < r$ since players have the same local histories in $\vec{h}$ and $\vec{h}'$ during the first $r - 1$ rounds of communication, and they also send exactly the same messages. An analogous argument shows that, for all $r' \geq r$, the lies told by players in K in round r' is identical in h and $\vec{h}'$. Finally, note that the lies told by players in $\bar{K}$ in round r strictly decreases from h to $\vec{h}'$ since they have exactly the same local history in both, send exactly the same messages to players in K in both, but in $\vec{h}'$ they tell no lies between them (by assumption, there is at least one of such lies in h). Thus, $\ell'_r < \ell_r$, and $L(\vec{h}') < L(\vec{h})$ as desired. □

This shows that b is a k-paranoid belief system consistent with $\vec{\sigma}$. □

5.4 Proof of Theorem 16

The proof of Theorem 16 is analogous to the one of Theorem 3. We start with a k-resilient Nash equilibrium $\vec{\sigma}$ that implements the desired sequential equilibrium (for example, we can use the protocols given in Chapter 4), and then we extend $\vec{\sigma}$ to a k-resilient sequential equilibrium by showing that there exists a k-paranoid belief system:

Proposition 31. *For all strategies $\vec{\sigma}$ in $\Gamma_{CT,asyn}$, there exists a k-paranoid belief assessment b that is consistent with $\vec{\sigma}$*

The proof of Proposition 31 is identical to that of Proposition 30 except that players believe that the scheduler strategy σ_e is sampled from a known distribution

X_e, and that players lie with increasing probability every time they send a new message. The rest of the proof of Theorem 16 is analogous to that of Theorem 15.

5.5 Extending the set of equilibria

Up to now we have considered only distributions over action profiles that involve rational probabilities. We can extend our result to some extent to distributions described by real numbers. Specifically if we assume that players can send messages and operate with real numbers, and an equilibrium $e = \sum_i \lambda_i e_i$ is a convex combination of rational equilibria e_i. If players could jointly sampling a real number $r \in [0, 1)$ uniformly at random, they could easily implement e: after sampling $r \in [0, 1]$ uniformly at random, they play their part of the rational equilibrium e_m such that $\sum_{i=1}^{m-1} \lambda_i \leq r < \sum_{i=1}^{m} \lambda_i$. (Our earlier results show that e_m can be imlemented.) Note that this procedure guarantees that players implement e_m with probability λ_m, as desired.

The following is a k-resilient strategy $\vec{\tau}$ that allows players to jointly sample r in the synchronous setting if $n > 3k$:

1. Each player privately generates a random number $r_i \in [0, 1)$ and broadcasts it using Bracha's k-resilient broadcast protocol [15]. If i doesn't broadcast a number, r_i is assumed to be 0.
2. Players take $r := frac(r_1 + \ldots + r_n)$, where $frac(x)$ is the fractional part of x.

In the asynchronous setting, this process is harder because players cannot distinguish players that didn't broadcast a value from players that are being delayed

by the scheduler. Moreover, players cannot agree on using only a subset C of the values shared since players might want to influence the computation of C in such a way that the number r chosen is beneficial for them. However, if $n > 4k$, each player can generate r_i as in the synchronous setting, and then compute r using a k-resilient secure computation [9].

CHAPTER 6

LOWER BOUNDS

Our goal in this chapter is to prove lower bounds that match the upper bounds of Theorem 2 and of Theorem 6. The matching lower bound for Theorems 7, 10 and 11 are proven in ADH [3] and the lower bound for Theorem 9 follows automatically from the definition of (k, t)-robustness and punishment strategy: if a strategy $\vec{\sigma}+\sigma_d$ is (k, t)-robust and $n \leq 2k + 3t$, then there is no $(2k + 2t)$-punishment strategy in the underlying game.

To do so, we would like to reduce implementing $(k + t)$-secure computation to implementing (k, t)-robust mediators. If such a reduction were possible, the $n > 4(k + t)$ lower bound for implementing (k, t)-robust mediators would follow immediately from the same lower bound for secure computation [5, 12, 16]. Unfortunately, there does not seem to be such a reduction. However, we show that there exists a nontrivial reduction from a slightly weaker notion of $(k + t)$-secure computation, which we call $(k + t)$-*strict secure computation*, to implementing (k, t)-robust mediators. We thus start by providing a careful proof of the lower bound for $(k+t)$-strict secure computation in the asynchronous setting. In the process, we also give a simple alternative proof for the lower bound on asynchronous secure computation. [1]

Intuitively, a protocol t-strictly securely computes a function f if it satisfies the properties of secure computation but only for adversaries consisting of exactly t malicious agents. It might seem that t-secure computation should be equivalent to t-strict secure computation. After all, if a function can be securely computed

[1] As Ran Canetti [private communication] agreed, there is a nontrivial problem with the proof given in his thesis [16]; a different technique is needed. We thank him for his comments.

with adversaries of maximal size, surely it can be securely computed with smaller adversaries! As we show by example in Section 6.1, this is not the case. Intuitively, the problem is that an adversary consisting of fewer than t agents may not be permitted to learn as much as an adversary consisting of t agents. While investigating these issues, we noted an ambiguity in the definition of t-secure computation in BCG, which led us to consider yet another notion that we call t-weak secure computation. As the name suggests, it is weaker than t-secure computation; we show that it is actually equivalent to t-strict secure computation. By considering these variants of secure computation, we gain a deeper understanding of its subtleties.

6.1 Weaker Notions of Secure Computation

Note that the T in the second condition of Definition 1, that is, the T in the trusted-party adversary (T, h, c, O), is the same as the T in the adversary. This is also true in the BGW definition of t-secure computation. While we believe that this was also the intention of BCG, their definition simply says that that there exists a t-trusted-party adversary, without specifying T (the set of malicious agents) that satisfies the second bullet of Definition 1. Taking this definition seriously leads to a slightly weaker notion of secure computation that we call *t-weak secure computation*, which is defined just as t-secure computation except that the t-trusted-party adversary A^{tr} may involve any subset T' of malicious agents such that $|T'| = t$ and $T' \supseteq T$, as opposed to consisting of the same set T of malicious agents as A.

We show next that t-weak secure computation is strictly weaker than the standard notion of secure computation. To do so, first we introduce an intermediate notion of secure computation called *t-strict secure computation*; it is defined just as t-secure computation, except that we require only that the properties are sat-

isfied for adversaries of size exactly t (i.e., for $|T| = t$). As we mentioned in the introduction, somewhat surprisingly, t-strict secure computation is strictly weaker than t-secure computation, but, as we show next, it is actually equivalent to t-weak secure computation.

Theorem 17.

(a) If a protocol $\vec{\pi}$ t-securely computes a function f, it also t-strictly securely computes f.

(b) A protocol $\vec{\pi}$ t-strictly securely computes a function f iff it t-weakly securely computes f.

(c) If $t > 1$ and $n > 4t$, there exists a function f on n variables and a protocol $\vec{\pi}$ such that $\vec{\pi}$ t-strictly securely computes f but does not t-securely computes f.

Proof. Part (a) follows immediately from the definition of secure computation and strict secure computation. For part (b), first suppose that a protocol $\vec{\pi}$ t-stricly securely computes $f : D^n \to D^n$. Given an adversary $A = (T, \vec{\tau}_T, \sigma_e)$ with $|T| \leq t$, consider an adversary of the form $A' = (T \cup T', \vec{\tau}_T + \vec{\pi}_{T'}, \sigma_e)$ (Since the agents in T' play $\vec{\pi}$, they in fact do not deviate.) Because $\vec{\pi}$ such that $T \cap T' = \emptyset$ and $|T \cup T'| = t$. Since $\vec{\pi}$ t-strictly securely computes f, there exists a trusted-party adversary $A^{tr} = (T \cup T', h, c, O)$ such that $\vec{\pi}(\vec{x}, A') = \vec{\rho}(A^{tr}, \vec{x}; f)$. By construction, $\vec{\pi}(\vec{x}, A') \sim \vec{\pi}(\vec{x}, A)$, since the additional malicious agents in A' do not deviate from the protocol. Therefore, $\vec{\pi}(\vec{x}, A) \sim \vec{\rho}(A^{tr}, \vec{x}; f)$, so $\vec{\pi}$ t-weakly securely computes f.

The converse is almost immediate from the definitions. Suppose that protocol $\vec{\pi}$ t-weakly securely computes $f : D^n \to D^n$ for some t. Given an adversary

$A = (T, \vec{\tau}_T, \sigma_e)$ with $|T| = t$, then, by assumption, there is a t-trusted-party adversary $A' = (T, h, c, O)$ such that $\vec{\pi}(\vec{x}, A') = \vec{\pi}(\vec{x}, A)$.

For part (c), consider the following setup. Let $\mathbb{F}_2$ be the field with domain $\{0, 1\}$. Given n and t such that $t > 1$ and $n > 4t$, consider a function $f : (\mathbb{F}_2)^{n^3} \to (\mathbb{F}_2)^{n^2}$ that does the following. Given the input $(x^i, c^i, y^i, z^i) \in (\mathbb{F}_2)^{n^2}$ of each agent i, where $x^i \in \mathbb{F}_2$, $c^i \in (\mathbb{F}_2)^n$, $y^i \in (\mathbb{F}_2)^{t-1}$, and z^i consists of the remaining $n^2 - n - t$ coordinates (which do not affect the function f; they are needed because in Definition 1, the input space of each agent must be the same as the output space of the function), let $p_i \in \mathbb{F}_2[X]$ be the unique polynomial of degree $t-1$ such that $p_i(0) = x^i$ and $p_i(j) = y^i_j$ for all $j = 1, 2, \ldots, t-1$. The output of f is then $\{p_i(j) + c^j_i\}_{i,j \in [n]}$. In other words, f encodes the first coordinate of each agent's input using Shamir's agent secret sharing scheme [27]. The polynomial p_i that each agent i uses to do the encoding and the one-time pads c^j_i added by i to each of the shares are part of i's input, and not known by the other agents. However, a coalition T of t malicious agents can reconstruct the values $p_i(j)$ for all $i \in [n]$ and $j \in T$, and thus is able to reconstruct each x_i as well, since the agents in T know t points on each polynomial p_i, although no coalition of size strictly smaller than t knows those values.

Consider a protocol $\vec{\pi}$ that consists of the following: each agent i performs its part of BCG's t-secure computation protocol to compute f and then, if i is included in the core set of the output, i broadcasts the first bit of its input. By the earlier argument, if the adversary is of size exactly t, it can reconstruct the first coordinate of the inputs of the agents in the core-set from the output of f and its own inputs, which means that the values broadcast after BCG's secure computation protocol do not give any extra information about the inputs of honest

agents to the adversary. However, this is not true for smaller adversaries. Thus, $\vec{\pi}$ t-strictly securely computes f, but does not t-securely compute f. □

6.2 Main Results

In this paper we show that the bound in Theorem 6 is tight:

Theorem 18. *If $k+t+1 < n \leq 4k+4t$ there exists a (k,t)-robust (resp., strongly (k,t)-robust) strategy profile $\vec{\sigma}+\sigma_d$ for n players and a mediator such that there is no (k,t)-robust (resp., strongly (k,t)-robust) strategy profile $\sigma_{\vec{ACT}}$ that implements $\vec{\sigma}+\sigma_d$.*

The proof of Theorem 18 is divided in two parts.

6.2.1 Case 1: $3k + 3t \leq n \leq 4k + 4t$

Here, we show that that $(k+t)$-strictly securely computing a function f reduces to implementing a (k,t)-robust strategy $\vec{\sigma}+\sigma_d$ for some game $\Gamma^{f,k,t}$. To make this precise, we need the following definition:

Definition 34. *If $g : A \to B$, and σ is a strategy that plays actions in A, then $g(\sigma)$ is the strategy that is identical to σ except that each action $a \in A$ is replaced by $g(a) \in B$. If $\vec{\sigma}$ is a strategy profile where each player i plays actions in A, then $g(\vec{\sigma}) = (g(\sigma_1), \ldots, g(\sigma_n))$.*

Theorem 19. *If $f : D^n \to D$, D is a finite domain, and $2(k+t) < n$, then there exists a game $\Gamma_d^{f,k,t}$ in which all players have the same set A of possible actions,*

a function $g : A \to D$, and a (k,t)-robust strategy (resp., strongly (k,t)-robust strategy) $\vec{\sigma} + \sigma_d$ for n players and the mediator in $\Gamma_d^{f,k,t}$ such that if $\vec{\sigma}_{ACT}$ is (k,t)-robust implementation (resp., strongly (k,t)-robust implementation) of $\vec{\sigma}+\sigma_d$, then $g(\vec{\sigma}_{ACT})$ $(k+t)$-strictly securely computes f.

Theorem 19 shows that being able to implement all (k,t)-robust mediators with n players implies that all functions on n variables can be $(k+t)$-strictly securely computed. The proof of Theorem 18 for $3(k+t) \leq n \leq 4(k+t)$ follows from the fact that if $3(k+t) \leq n \leq 4(k+t)$ there exist functions that cannot be $(k+t)$-weakly securely computed:

Theorem 20.

(a) If $n > 4t$ or $n \leq 2t$, every function $f : D^n \to D$ can be t-weakly securely computed in asynchronous systems.

(b) If $3t \leq n \leq 4t$, there exists a domain D and a function $f : D^n \to D$ that cannot be t-weakly securely computed in asynchronous systems.

The proof of Theorem 20 is given in Section 6.3. A slight variation of the proof provides a simple proof for the well-known lower bound for secure computation on asynchronous systems:

Theorem 21.

(a) If $n > 4t$ or $n \leq t$, for all domains D, every function $f : D^n \to D$ can be t-securely computed in asynchronous systems.

(b) If $t < n \leq 4t$ there exists a domain D and a function $f : D^n \to D$ that cannot be t-securely computed in asynchronous systems.

6.2.2 Case 2: $k + t + 1 < n \leq 3k + 3t$

If $k + t + 1 < n \leq 3k + 3t$ we show that implementing $(k + t)$-resilient weak consensus with n players can be reduced to implementing (k, t)-robust mediators:

Theorem 22. *If $n > k+t+1$, then there exists a game $\Gamma_d^{k,t}$ in which all players have the same set A of possible actions, a function $g : A \to \{0, 1\}$, and a (k, t)-robust (resp., strongly (k, t)-robust) strategy $\vec{\sigma} + \sigma_d$ for n players and the mediator in $\Gamma_d^{k,t}$ such that if $\vec{\sigma}_{ACT}$ is a (k, t)-robust (resp., strongly (k, t)-robust) implementation of $\vec{\sigma} + \sigma_d$, then $g(\vec{\sigma}_{ACT})$ is a $(k + t)$-resilient implementation of weak consensus.*

The proof of Theorem 18 for $k + t + 1 \leq n \leq 3(k + t)$ follows from Lamport's lower bound for weak consensus [25].

6.3 Proof of Theorems 21 and 20

For Theorem 21(a), note that if $n > 4t$, Theorem 2 shows that every function $f : D^n \to D$ can be t-securely computed, and thus t-weak securely computed as well. If $n \leq t$, let $\perp$ be the input assigned to the agents that did not submit an input. It can be easily shown that the protocol where each agent sends no messages and outputs $(\emptyset, f(\perp^n))$ t-securely computes f. Similarly, for Theorem 20(a), it can be easily checked that if $n \leq 2t$, the protocol where each agent sends nothing and outputs $([n - t], f(\perp^n))$ t-weak securely computes f.

For the lower bounds (Theorems 21(b) and 20(b)), our proof is similar to that of Canetti [16] at a high level: We construct a function f with four inputs, the scheduler schedules the agents so that the fourth agent never gets to participate in

the computation, and one of the three remaining agents is malicious and manages to trick the other two participating agents into outputting something inappropriate. Canetti then claims that *conversations* between agents (where a conversation is just the collection of messages sent by two given agents) must be independent of the inputs of the agents, agent 3 can send messages to agents 1 and 2 in such a way that agents 1 and 2 believe they should output different values. However, there are two significant problems with this approach:

(a) First, the conversations between the agents might not be totally independent of their inputs, since they can depend on the output of the computation, and this ultimately does depend on the inputs. For example, agents can run Bracha's [14] consensus protocol (which tolerates t malicious agents if $n > 3t$) after terminating the secure computation protocol to decide the output. This would guarantee that all honest agents output the same value at the end of the computation, so their conversations are certainly not independent.

(b) Second, there is a more subtle issue when trying to simultaneously trick agents 1 and 2 into outputting some given values a and b, respectively. Even though Canetti proves that for the function f that he uses and a particular input $\vec{x}$, for each conversation $h_{1,2}$ between 1 and 2, there is a protocol for player 3 that results in a conversation $h_{1,3}$ between 1 and 3 such that 1 outputs a, and that for each conversation $h_{1,2}$ between 1 and 2 there exists a protocol for player 3 that results in a conversation $h_{2,3}$ between 2 and 3 such that 2 outputs b, there might not exist a protocol for agent 3 that results in 1 and 2 having conversation $h_{1,3}$ and agents 2 and 3 having conversation $h_{2,3}$ simultaneously. In fact, if a and b are different and agents run a consensus protocol as in (a), there is not.

Roughly speaking, we deal with these issues as follows. We prove that for our function f, a malicious agent can make all honest agents output the same incorrect value, and we show that in our case there does exist a conversation $h_{1,2}$ between 1 and 2 such that agent 3 can trick both of them simultaneously, as desired (see Lemma 15). Some of these techniques can also be applied to prove lower bounds for weak secure computation.

Consider the function $f^n : \{0, 1, \perp\}^n \to \{0, 1, \perp\}$ that essentially takes majority between 0 and 1: it outputs 1 if the number of agents with input 1 is greater or equal to the number of agents with input 0, otherwise it outputs 0. Players who do not submit an input are assumed to have input $\perp$. We start by showing that f^4 cannot be 1-weakly securely computed by four agents.

Suppose that f^4 can be 1-weakly securely computed using a protocol $\vec{\sigma}$. Let σ_e^N be the scheduler that schedules agents 1, 2, and 3 cyclically, and right before scheduling an agent, it delivers the messages that were sent by the other agents the last time they were scheduled. After scheduling each of the first three agents N times, it schedules agent 4 as well, adding it to the cyclic order.

Given a history H, let $\vec{x}_H$ denote the input profile of agents in H, let H_i denote agent i's local history in H, let H_e denote the scheduler's local history in H, and let $H_{(i,j)}$ denote the *conversation* between agents i and j (i.e., the messages sent and received between i and j, in addition to the relative times at which i and j were scheduled). We can now prove essentially what BCG claimed to prove (although, as we said, this claim does not hold for the BCG construction).

Lemma 15. *There exist N and two (finite) histories H and H' of $\vec{\sigma}$ where the scheduler uses σ_e^N, $\vec{x}_H = (1, 0, 1, 1)$, $\vec{x}_{H'} = (0, 1, 1, 1)$, agents 1, 2, and 3 all output 1 in H, agent 4 is never scheduled in either H or H', $H_{1,2} = H'_{1,2}$, and $H_e = H'_e$.*

To prove Lemma 15, we first need to prove what seems to be obvious: if all agents are honest, at most t agents have input 0, and $n \geq 3t$, then the output of a weakly secure computation of f^n will be 1. While this seems obvious (and is true), it is not quite so trivial. For example, it is not true if $n = 3t-1$. In this case, if we consider a trusted-party adversary $A = (T, c, h, O)$, in which $|T| = t$, h replaces all inputs of malicious players with 0, and c chooses all t malicious players and $t-1$ additional honest players, it is easy to check that the output of honest players is 0.

Lemma 16. *Let $n \geq 3t$ and let $\vec{\sigma}$ be a protocol that t-weakly securely computes f^n. Then for all schedulers, in all histories of $\vec{\sigma}$ in which all agents are honest and at most t agents have input 0, all agents output 1.*

Proof. Let S be the subset of agents that have input 0. Given a scheduler σ_e, consider an adversary $A = (T, \vec{\tau}_T, \sigma_e)$ such that $T \supseteq S$, $|T| = t$, and $\vec{\tau}_T = \vec{\sigma}_T$ (so all the malicious agents follow protocol $\vec{\sigma}$). By definition of t-weak secure computation, the output of honest agents with adversary A should be one that is possible with a trusted-party adversary of the form $A' = (T, c, h, O)$. However, no matter what the output C of c is, since $|C| \geq n - t$, there will be at least $n - 2t$ honest agents in C, all of them with input 1. Since $n \geq 3t$, this suffices to guarantee that all players not in T output 1. Since malicious agents play $\vec{\sigma}$, they are indistinguishable from honest agents. Thus, if all agents are honest, all agents not in T output 1. To see that agents in T also output 1 if all players are honest, consider an adversary $A'' = (T', \vec{\tau}_{T'}, \sigma_e)$ such that $T' \cap T = \emptyset$, $|T'| = t$ (such a set T' always exists since $n \geq 3t$), and $\vec{\tau}_{T'} = \vec{\sigma}_{T'}$. Since honest agents not in $T \cup T'$ (note that $[n] \setminus (T \cup T') \neq \emptyset$) have the same histories with A and A'', they must output the same value with both adversaries, and so must output 1 with adversary A''. By definition of t-weak secure computation, since $|T'| = t$, all agents not in T'

must output the same value. Thus, since $T \cap T' = \emptyset$, all agents in T also output 1 with adversary A''. Again, since agents in T' are indistinguishable from honest agents, this implies that agents in T also output 1 if all agents are honest. □

Proof of Lemma 15. By Lemma 16, there exists an integer N such that if agents 1, 2, and 3 are honest, with nonzero probability, they will output 1 with scheduler σ_e^N at or before the Nth time they are scheduled. Let H be a history where the agents use $\vec{\sigma}$, the scheduler uses σ_e^N, the input is $(1, 0, 1, 1)$, agents 1, 2, and 3 are honest and have been scheduled at most N times and all three have outputted 1. By the properties of secure computation, in particular, the secrecy of the inputs, there must exist a history H'' such that $\vec{x}_{H''} = (1, 1, 0, 1)$, $H''_1 = H_1$, and $H''_e = H_e$. (Note that this means that we can assume, without loss of generality, that the scheduler uses strategy σ_e^N.) If this were not the case and agent 1 were malicious in H'', then it would know that the input profile can't be $(1, 1, 0, 1)$ given histories H_1 and H_e. (Recall that we can assume without loss of generality that the malicious agents can communicate with the scheduler.) Similarly, there exists a history H' with $\vec{x}_{H'} = (0, 1, 1, 1)$ such that $H'_2 = H''_2$ and $H'_e = H''_e$. The fact that the scheduler has the same local history in H, H', and H'' and that $H_1 = H''_1$ and $H''_2 = H'_2$ implies that $H_{1,2} = H'_{1,2} (= H''_{1,2})$, as desired. In more detail, since $H'_2 = H''_2$, agent 2 sends the same messages to and receives the same messages from agent 1 in H' and H'', so 1 receives the same messages from and sends the same messages to 2 in both H' and H''. Thus, $H'_{1,2} = H''_{1,2}$. A similar argument shows that $H_{1,2} = H''_{1,2}$. □

Now suppose that agents have input profile $\vec{x} = (0, 0, 0, 0)$. We show that there exists a strategy τ_3 for agent 3 such that if all other agents play $\vec{\sigma}$ and the scheduler plays $\vec{\sigma}_e^N$, then with non-zero probability, agents 1 and 2 output 1. This

suffices to show that f^4 cannot be 1-weakly securely computed, since honest agents should output 0 when playing with any trusted-party adversary with at most one malicious agent.

Lemma 17. *If the agents have input profile* $(0,0,0,0)$*, then there exists a strategy* τ_3 *for agent 3 such that if all other agents play* $\vec{\sigma}$ *and the scheduler plays* $\vec{\sigma}_e^N$*, then with non-zero probability, agents 1 and 2 output 1.*

Proof. Let H and H' be the two histories guaranteed to exist by Lemma 15. The strategy τ_3 for agent 3 consists of sending agent 1 the messages that agent 3 sends to agent 1 in H' while sending agent 2 the messages that agent 3 sends to agent 2 in H. Suppose that agent 1 has the same random bits as in H', while agent 2 has the same random bits as in H. An easy induction now shows that, in the resulting history, agent 1 will have history H'_1 and agent 2 will have history H_2 after each having been scheduled at most N times, using the fact that, as shown in Lemma 15, $H_{12} = H'_{12}$. Thus, by Lemma 15, agents 1 and 2 output 1 in this case. This contradicts the fact that $\vec{\sigma}$ 1-weakly securely computes f^4, since Lemma 16 shows that, with input profile $(0,0,0,0)$, all honest players output 0. □

It is straightforward to extend this argument to all n and t such that $3t \leq n \leq 4t$. Given n and t such that $3t \leq n \leq 4t$, we divide the agents into four disjoint sets S_1, S_2, S_3, and S_4 such that $0 < |S_i| \leq t$ for all $i \in \{1,2,3\}$ and $0 \leq |S_4| \leq t$. Consider a scheduler σ_e^N that schedules agents in S_1, S_2 and S_3 cyclically and, right before scheduling an agent, it delivers the messages that were sent by the other agents the last time they were scheduled. After scheduling each of the agents in $S_1 \cup S_2 \cup S_3$ N times, it schedules the agents in S_4 as well. Suppose that $\vec{\sigma}$ is a strategy for n agents that t-securely computes f^n.

Lemma 18. *There exist N and two (finite) histories H and H' of $\vec{\sigma}$ where the scheduler uses σ_e^N, $\vec{x}_H = (1_{S_1}, 0_{S_2}, 1_{S_3}, 1_{S_4})$, $\vec{x}_{H'} = (\vec{0}_{S_1}, \vec{1}_{S_2}, \vec{1}_{S_3}, \vec{1}_{S_4})$, agents in $S_1 \cup S_2 \cup S_3$ output 1 in H, agents in S_4 are never scheduled in either H or H', $H_{S_1,S_2} = H'_{S_1,S_2}$ (which is the conversation between the agents in S_1 and the agents in S_2) and $H_e = H'_e$.*

Proof. The proof is analogous to the proof of Lemma 15; the subsets S_1, S_2, S_3, and S_4 play the roles of agents 1, 2, 3, and 4, respectively. □

We now have the tools we need to prove Theorem 20(b). Given H and H' from Lemma 18, consider a strategy $\vec{\tau}_{S_3}$ for agents in S_3 that consists of sending agents in S_1 and S_2 exactly the same messages they would send in H' and H respectively. Again, if agents have input $\vec{0}$, with non-zero probability, agents in S_2 will eventually have history H_{S_2}, and thus will output 1, contradicting the assumption that $\vec{\sigma}$ t-weakly securely computes f^n. This completes the proof of Theorem 20(b).

The proof of Theorem 21(b) follows similar lines. We start with an analogue of Lemma 16 which holds for a larger range of values of n:

Lemma 19. *Let $n \geq t+2$ and $\vec{\sigma}$ be a protocol that t-securely computes f^n. Then, in all histories of $\vec{\sigma}$ in which all agents are honest and at most $(n-t)/2$ agents have input 0, all agents output 1.*

Proof. Given any scheduler σ_e, if all agents are honest, their output should be one that is possible with a trusted-party adversary of the form $A = (\emptyset, c, h, O)$. No matter what the output C of c is, at most $(n-t)/2$ agents in C have input 0.

Since $|C| \geq n - t$, at least half of the agents in C have input 1 and thus all honest agents output 1. □

We also need the following technical result:

Lemma 20. *If* $t + 2 \leq n \leq 4t$ *then*

(a) $n \geq 3\lceil \frac{n-t}{3} \rceil$*;*

(b) $\lceil \frac{n-t}{3} \rceil \leq \frac{n-t}{2}$.

Proof. If $t + 2 \leq 4t$ then $t > 0$. To prove part (a), note that if $t = 1$, then n can be only 3 or 4. In both cases, the inequality is satisfied. If $t \geq 2$ then $\lceil \frac{n-t}{3} \rceil \leq \lceil \frac{n-2}{3} \rceil \leq \frac{n}{3}$, from which the desired result immediately follows. To prove part (b), let a and b be the two positive integers such that $n - t = 3a + b$ with $1 \leq b \leq 3$. Then $\lceil \frac{n-t}{3} \rceil = a + 1$ and $\frac{n-t}{2} = \frac{3a+b}{2} = a + \frac{a+b}{2}$. Since $n - t \geq 2$, then either $a > 0$ or $b > 1$. Since $b \geq 1$, in both cases, $a + 1 \leq a + \frac{a+b}{2}$. □

Given n and t such that $t + 2 \leq n \leq 4t$, we divide the agents into four disjoint sets S_1, S_2, S_3, S_4 such that $|S_i| = \lceil \frac{n-t}{3} \rceil$ for $i \leq 3$ and $|S_4| \leq t$ (which is always possible, by Lemma 19(a)). If $n \geq t + 2$, then by Lemma 20(b), $\lceil \frac{n-t}{3} \rceil \leq \frac{n-t}{2}$, and thus by Lemma 19, in all histories in which all agents are honest and have inputs $(0_{S_1}, 1_{S_2}, 1_{S_3}, 1_{S_4})$, $(1_{S_1}, 0_{S_2}, 1_{S_3}, 1_{S_4})$ or $(1_{S_1}, 1_{S_2}, 0_{S_3}, 1_{S_4})$, all the agents output 1. Reasoning analogous to that used in the proof of Theorem 20(b) then shows that f^n cannot be t-securely computed for $t + 2 \leq n \leq 4t$.

It remains to deal with the case where $n = t + 1$. To show that there exist functions that cannot be t-securely computed if $n = t + 1$, we reduce *t-resilient weak consensus* to t-secure computation.

Definition 35. *A protocol $\vec{\sigma}$ for n agents is a t-resilient implementation of weak consensus if the following holds for all adversaries $A = (T, \vec{\tau}_T, \sigma_e)$ with $|T| \leq t$ and all histories:*

(a) All agents not in T output the same value.

(b) If all agents are honest and have the same input x, all agents output x.

Lamport [25] showed that if $n \leq 3t$ there is no t-resilient implementation of weak consensus. Thus, if there exists a reduction from t-resilient weak consensus to t-secure computation for $n > t$, then there are functions $f : D^n \to D$ with $n = t + 1$ that cannot be t-securely computed.

The reduction proceeds as follows: Consider a function $g^n : \{0, 1, \perp\} \to \{0, 1, \perp\}$ such that $g^n(\perp, \ldots, \perp) = \perp$, and $g^n(x_1, \ldots, x_n) = x_i$ if $x_i \neq \perp$ and $x_j = \perp$ for all $j < i$; that is, g^n outputs the first non-$\perp$ value if there is one, and otherwise outputs $\perp$. Suppose, by way of contradiction, that $\vec{\sigma}$ t-securely computes g^n. Let $\vec{\tau}$ be identical to $\vec{\sigma}$, except hat, whenever agent i would have output (C, v) with σ_i, it outputs v instead if $v \neq \perp$, and otherwise it outputs 0. By the properties of t-secure computation, all honest agents output the same value when using $\vec{\tau}$. Moreover, if all honest agents have input 0 or all of them have input 1, if $n > t$, then the output of the secure computation has the form $(C, 0)$ or $(C, 1)$, respectively. Thus, if there exists a protocol that t-securely computes g^n for $n = t + 1$, then there exists also a t-resilient implementation of weak consensus for $t + 1$ agents, contradicting Lamport's result. This proves Theorem 21.

6.4 Proof of Theorem 19

We prove Theorem 19 only for the case of (k,t)-robustness; the proof in the case of (k,t)-strong robustness is identical.

A naive construction of Γ_d and $\vec{\sigma}+\sigma_d$ proceeds as follows. The set of actions of each player consists of all possible outputs of a secure computation of f in addition to their type (more precisely, actions are of the form (C,z,Q) with $C \subseteq [n]$ and $z \in D$ and $Q \in \{H,R,M\}$, where H stands for honest, R stands for rational, and M stands for malicious). If there is no subset S of at least $n-k-t$ honest players such that players in S securely compute f, that is, every subset S of $n-k-t$ players either do not all output the same value or they all output a value that is not a possible output of a secure computation of f, then rational players get a higher payoff and/or the honest players get a lower payoff. In $\vec{\sigma}+\sigma_d$, each player sends its input to the mediator when it is scheduled for the first time. The mediator waits until it receives the input x_i from a set C of players with $|C| \geq n-k-t$, then computes $z := f_C(\vec{x})$ and sends (C,z,H) to all players. Players play (C,z,H) when they receive the message.

It would seem that any (k,t)-robust implementation of $\vec{\sigma}+\sigma_d$ also $(k+t)$-strictly securely computes f. In fact, any $(k+t)$-secure computation of f is also a (k,t)-robust implementation of $\vec{\sigma}+\sigma_d$, but not all (k,t)-robust implementations $(k+t)$-securely compute f. As we suggested in the main text, consider a protocol $\vec{\sigma}_{ACT}$ in which players perform BCG's $(k+t)$-secure computation protocol and broadcast their inputs immediately afterwards. It is easy to check that $\vec{\sigma}_{ACT}$ is a (k,t)-robust implementation of $\vec{\sigma}+\sigma_d$ whenever $n > 4(k+t)$, but that $\vec{\sigma}_{ACT}$ does not $(k+t)$-securely compute f, since it leaks the honest players' inputs to all other players.

This shows that it is necessary to somehow encode all information that malicious players can learn into the set of actions of Γ_d in such a way that they can increase their payoff if they manage to learn anything about the other players' inputs besides what can be learned from the output of the computation. The idea for doing this is that, besides the output, the action of each player should include a *guess* as to what the input profile $\vec{x}$ is (they can also guess $\perp$ if they have no guess). If a player i guesses correctly it receives an additional positive payoff q_i, while if it guesses wrong, its payoff decreses by 1. The value of q_i should be chosen in such a way that (a) it is never worthwhile deviating if i is not able to learn anything besides the output, and (b) it is always worthwhile deviating if i is able to learn something (otherwise, $\vec{\sigma}_{ACT}$ may not $(k+t)$-strictly securely compute f even if it is (k,t)-robust, as in the example above). Given the set C of players whose inputs are included in the computation, the output z of f, and the input profile $\vec{x}$, let p_i be the probability that a player i guesses $\vec{x}$ conditional on its own input x_i, C, and z. Conditions (a) and (b) imply that $p_i q_i + (1-p_i)(-1) \leq 0$ and $p_i q_i + (1-p_i)(-1) \geq 0$ respectively, which means that $p_i q_i + p_i - 1 = 0$ and thus that $q_i = p_i^{-1} - 1$.

This approach cannot be generalized easily to a situation where a coalition of players may deviate. In this case, a player in the coalition will know the values of all players in the coalition, not just its own input. Moreover, if a player in the coalition plays just like an honest player, except that it tells the other coalition members its input, then this is completely indistinguishable (by the honeset players) from the scenario in which that player is honest and the other members of the coalition were just lucky guessing its input. In addition, a player can lie about its input if it is easier to guess the input profile with a different input than its own. Since the payoffs of a Bayesian game depend only on their actions and their real input

profile, it is always worthwhile for a player to lie about its input if this is the case. For instance, suppose that $D = \mathbb{F}_2$ and that $f(\vec{x}) = \prod_{i=1}^{n}(1 - x_i)$. If i has input 1 and plays honestly, then it learns absolutely nothing about the other players' inputs, since the output will be 0 no matter what. However, if i pretends to have input 0, i will learn more information: if the output is 1, then all players have input 0; otherwise, at least one player has input 1. In this case, it would always be worthwhile for player i to act as if it has input 0, regardless of its actual input. This shows that to compute the probability that the adversary guesses the inputs correctly, it is critical to know who is malicious and what inputs the malicious players are pretending to use in the computation.

To deal with the fact that we may not be able to tell which players are deviating, we require that exactly $k + t$ players must try to guess a non-$\perp$ value in order to get an additional (positive or negative) payoff. Moreover, their guesses must be identical. If honest players always guess $\perp$, this suffices to identify the coalition of $k + t$ deviating players given their action profile. Note that this is why we require strict secure computation in Theorem 19. If we required only (standard) secure computation, smaller adversaries wouldn't be able to get a better payoff, even if they managed to guess the inputs of everyone else (thus it wouldn't satisfy condition (b)). To deal with players lying about their inputs, we require that the action profile of the players encode the inputs used by the players for the computation (even though these inputs may differ from their actual inputs). The probability of guessing the input profile is based on the inputs used, not players' actual inputs. Note that these values must be encoded into the action profile without any coalition of $k + t$ players learning anything about them. This can be done as follows: each player i, in addition to the set C, the output z and their guess b_i, also outputs n values $s_{i1}, \ldots, s_{in}$ such that the values $s_{i,j}$ for a fixed j

encode the value used by j for the computation (using Shamir's scheme).

There is one final issue. The definition of $(0, t)$-robustness is equivalent to that of t-immunity, which means that no coalition of t players can decrease the payoff of other players by deviating. In this case, the effect of a coalition of t players being able to learn something about the inputs should be to decrease the payoffs of the remaining players, rather than increasing their own payoff. To deal with this, we require players to declare wither they are G (good), R (rational), or M (malicious). If a coalition of $(k + t)$ players tries to guess the inputs of everyone else, if they all declare R, then they get an additional payoff as described above. Otherwise, everyone gets the negative of that value.

We now formalize these ideas. Given f and integers k and t such that $n > 2k + 2t$, consider the game $\Gamma^{f,k,t}$, defined as follows. The input profile of the players is chosen uniformly at random from D^n. The set of actions of each player in $\Gamma^{f,k,t}$ is $\{G, R, M\} \times 2^{[n]} \times D \times D^n \times (D^n \cup \perp)$, so an action of player i has the form $a_i = (Q_i, C_i, z_i, \vec{s}_i, b_i)$, where $Q_i \in \{G, R, M\}$, $C_i \subseteq [n]$, $z_i \in D$, $x_i \in D$, and $b_i \in D^n \cup \perp$. Intuitively, Q_i denotes if i is good (G), rational (R), or malicious (M), (C_i, z_i) is i's output in the secure computation of f; s_{ij} is i's share of j's input (this will be made clearer below), and b_i is i's guess of the (supposedly secret) input, where $b_i = \perp$ if i has no guess.

We next define the utility function. We take $u_i = u_i^1 + u_i^2$, where, intuitively, u_i^1 is the utility that i gets if honest players either output different values or some honest player outputs a value that is not a possible output of a secure computation of f and u_i^2 is the utility that i gets from guessing the correct input of the other players. To define u_i^1, we first define what it means for an action profile $\vec{a}$ to be *secure for an input profile* $\vec{x}$. This is the case if there exist subsets $C, T \subseteq [n]$ with

$|C| \geq n - t - k$ and $|T| = k + t$, a vector $\vec{v} \in D^{k+t}$, and n polynomials $p_1, \ldots, p_n$ of degree $k + t$ (where, intuitively, p_j encodes j's input, so $p_j(i)$ is i's share of j's input) such that, for each player $j \notin T$, the action $a_j = (Q_j, C_j, z_j, \vec{s}_j, b_j)$ of player j satisfies (1) $Q_j = G$, (2) $C_j = C$, (3) $b_j = \perp$, (4) $z_j = f(\vec{y})$, where $\vec{y} = (\vec{x}/_{(T,\vec{v})})/_{(\overline{C},\vec{0})}$ (5) $p_{j'}(j) = s_{jj'}$ for all $j' \in [n]$, and (6) $p_j(0) = y_j$. We say that $\vec{a}$ *is* $(T, \vec{x})$*-secure* if these properties hold for the set T. Intuitively, $\vec{a}$ is $(T, \vec{x})$-secure if it could be the output of a $(k + t)$-secure computation of f with input $\vec{x}$, and the inputs used for the computation—which may differ from the the actual input profile due to deviating players lying about their inputs—were shared correctly between the players. If $\vec{a}$ is secure for $\vec{x}$, then $u_i^1(\vec{a}, \vec{x}) = 0$ for all $i \in [n]$; if $\vec{a}$ is not secure for $\vec{x}$ and at least one player i played R (i.e., played an action with $Q_i = R$), then $u_j^1(\vec{a}, \vec{x}) = 1$ for all players j; otherwise, $u_j^1(\vec{a}, \vec{x}) = -1$ for all players j.

If $\vec{a}$ is not secure for $\vec{x}$, then $u_i^2(\vec{a}, \vec{x}) = 0$. If $\vec{a}$ is secure for $\vec{x}$, let K be the subset of players that did not play G. Note that if $\vec{a}$ is secure for $\vec{x}$, then $|K| \leq k + t$. If $|K| < k + t$ or not all players in K guess the same value b (i.e. not all players in K have the same value b as the last component of their actions), then $u_i^2(\vec{a}, \vec{x}) = 0$ for all i. Otherwise, let b be the common guess of players in K and let p be the probability that a vector $\vec{w}$ sampled uniformly from D^n is equal to $\vec{x}$, conditional on $\vec{w}_K = \vec{y}_K$ and $f_C(\vec{w}) = z$. Note that if $\vec{a}$ is $(T, \vec{x})$-secure for some T, then C is uniquely determined by $\vec{a}$, and if, in addition, $n > 2(t + k)$, then $\vec{y}$ is also uniquely determined by the shares $\vec{s}_i$ of players $i \notin T$. If $b = \vec{\perp}$, then $u_i^2(\vec{a}, \vec{x}) = 0$ for all i. If at least one player $i \in K$ played R in its action, then, if $b = \vec{x}$, $u_i^2(\vec{a}, \vec{x}) = p^{-1} - 1$ for all $i \in K$; otherwise, $u_i^2(\vec{a}, \vec{x}) = -1$ for all $i \in K$. On the other hand, if no player $i \in K$ played R in its action, then, if $b = \vec{x}$, $u_i^2(\vec{a}, \vec{x}) = 1 - p^{-1}$ for all $i \notin K$; otherwise, $u_i^2(\vec{a}, \vec{x}) = 1$ for all $i \notin K$. Note that since $p(p^{-1} - 1) - (1 - p) = 0$, the payoffs u_i^2 are designed in such a way that the adversary can, in expectation,

either increase its payoff (if there are any rational players) or decrease the payoff of everyone else (if there are no rational players) if it can guess the inputs of honest players with higher probability than p (which is the probability of guessing the honest players' inputs if the adversary knew nothing but the output of the function and its own strategy and inputs).

Consider the following strategy $\vec{\sigma} + \sigma_d$ for $\Gamma_d^{f,k,t}$. According to σ_i, player i sends its input to the mediator at the beginning of the game. If i receives a message msg from the mediator, it plays msg in the underlying game. According to σ_d, the mediator waits until there exists a set $C \subseteq [n]$ with $|C| \geq n - t - k$ such that it has received exactly one message from each player $i \in C$ and each of these messages consists of a value $y_i \in D$. The mediator computes n polynomials $p_1, \ldots, p_n \in D[X]$ of degree $k + t$ whose non-constant coefficients are chosen uniformly at random and such that $p_i(0) = y_i$ if $i \in C$ and $p_i(0) = 0$ otherwise; it then computes $z := f(p_1(0), \ldots, p_n(0))$ and sends $(G, C, z, p_1(i), \ldots, p_n(i), \perp)$ to each player i.

Proposition 32. *$\vec{\sigma} + \sigma_d$ is (k, t)-robust and the equilibrium payoff is 0.*

Proof. Let $u_i(\vec{\sigma}, A)$ be the expected payoff of player i when playing $\vec{\sigma}$ with adversary A. It follows by construction that $u_i^1(\vec{\sigma} + \sigma_d, A) = 0$ for all adversaries $A = (T, \vec{\tau}_T)$ of size at most $k + t$ since, no matter what T is, the output profile $\vec{a}$ is $(T, \vec{x})$-secure for all input profiles $\vec{x}$.

Thus, $\vec{\sigma} + \sigma_d$ is not (k, t)-robust if and only if there exists an adversary $A = (T, \vec{\tau}_T)$ with $|T| \leq k + t$ and an input profile $\vec{x}_T$ such that, in expectation, (a) $u_i^2(\vec{\sigma} + \sigma_d, A, \vec{x}_T) > 0$ for some $i \in T$, or (b) $u_i^2(\vec{\sigma} + \sigma_d, A, \vec{x}_T) < 0$ for all $i \notin K$. The definition of u_i^2 guarantees that, in both cases, the adversary must consist of exactly $k+t$ players and these players must all play a non-G action. Moreover, these

players must guess the input of honest players with a probability higher than they could guess it by just knowing the output of the function, their strategy, and their inputs. However, the construction of $\vec{\sigma} + \sigma_d$ guarantees that don't have any extra information (note that C depends only on the adversary, and that the adversary does not get any information about the input of the honest players besides the value of z, since the polynomials p_i are all of degree $k + t$). □

We next show that if there exists a (k, t)-robust strategy that implements $\vec{\sigma}+\sigma_d$, then that strategy also $(k + t)$-securely computes f. We first need the following lemma.

Lemma 21. *If $\vec{\sigma}_{ACT}$ is a (k, t)-robust strategy that implements $\vec{\sigma} + \sigma_d$, then for all adversaries $A = (T, \vec{\tau}_T, \sigma_e)$ with $|T| = k + t$, all inputs $\vec{x}$, and all histories H of $\vec{\sigma}_{ACT}$ with adversary A and input $\vec{x}$, the action profile $\vec{a}$ played in H is $(T, \vec{x})$-secure.*

Proof. Suppose that $k > 0$. If there exists an input $\vec{x}$ and an adversary $A = (T, \vec{\tau}_T, \sigma_e)$ with $|T| = k + t$ such that, for some history H, the action profile $\vec{a}$ played in H is not $(T, \vec{x})$-secure, consider the adversary $A' = (T, \vec{\tau}'_T, \sigma_e)$ where $\vec{\tau}'_T$ is identical to $\vec{\tau}_T$, except that if a player $i \in T$ plays an action a with τ_i, then i instead plays $(R, \emptyset, 0, 0, \perp)$ with τ'_i. Thus, if an action profile $\vec{a}'$ played in some history H' when $\vec{\sigma}_{ACT}$ is run with adversary A' is $(T', \vec{x})$-secure, then $T \subseteq T'$ (note that for an action profile $\vec{a}$ to be $(T', \vec{x})$-secure, we require that all players not in T' play G in the first component, and none of the players in T plays G) and, since $|T| = k + t$, $T = T'$. Since the histories generated by playing with adversaries A and A' are indistinguishable by honest players, if there exists a history H with adversary A and input $\vec{x}$ such that the action $\vec{a}$ played in H is not $(T, \vec{x})$-secure, then the resulting action profile $\vec{a}'$ of playing $\vec{\sigma}_{ACT}$ with adversary A' and input $\vec{x}$

in which all players use the same randomization as in H is not $(T, \vec{x})$-secure, and all players in T would get a payoff of 1 rather than 0. It follows that $\vec{\sigma}_{ACT}$ is not (k,t)-robust. If $k = 0$, the argument is analogous, except that players in T play $(M, \emptyset, 0, 0, \perp)$ rather than $(R, \emptyset, 0, 0, \perp)$ □

To complete the proof of Theorem 19, we must show that if $\vec{\sigma}_{ACT}$ is a (k,t)-robust implementation of $\vec{\sigma} + \sigma_d$, the output of an adversary $A = (T, \vec{\tau}_T, \sigma_e)$ with $|T| = k + t$ is just a (randomized) function of its input $\vec{x}_T$ and the output v of the function. To do this, we need the following lemma:

Lemma 22. *Consider two random variables X and Y that take values on countable spaces S_1 and S_2 respectively. Then, $\Pr[X = x \mid Y = y] = \Pr[X = x \mid Y = y']$ for all $x \in S_1$ and $y, y' \in S_2$ if and only if X and Y are independent.*

Proof. If $\Pr[X = x \mid Y = y]$ does not depend on y, there exists a constant λ_x such that $\Pr[X = x \mid Y = y] = \lambda_x$ for all $y \in S$. Then, since $\Pr[X = x \mid Y = y] = \frac{\Pr[X=x, Y=y]}{\Pr[Y=y]}$, it follows that $\Pr[X = x, Y = y] = \lambda_x \Pr[Y = y]$. Therefore, $\sum_{y \in S_2} \Pr[X = x, Y = y] = \sum_{y \in S_2} \Pr[Y = y]$, which gives that $\lambda_x = \Pr[X = x]$, as desired. The converse is straightforward. □

We can now complete the proof of Theorem 19. Suppose that $k > 0$. Recall that if all players $i \in T$ set b_i to some input $\vec{x}$, they get a payoff of $p_{\vec{x}}^{-1} - 1$, where $p_x \in [0, 1]$ if the input profile is indeed $\vec{x}$, and otherwise they get -1. Given a history H_T in which the adversary has input $\vec{x}_T$ and honest players output v, let $p_v^{H_T}(\vec{x})$ be the probability that the input profile is $\vec{x}$ conditional on v and H_T. If $p_v^{H_T}(\vec{x}) > p_{\vec{x}}$, then $p_v^{H_T}(\vec{x})(p_{\vec{x}}^{-1} - 1) + (-1)(1 - p_v^{H_T}(\vec{x})) > 0$, which means that taking $b_i = \vec{x}$ is strictly better than taking $b_i = \perp$ for each of the players in T, contradicting the assumption that $\vec{\sigma}_{ACT}$ is (k,t)-robust. Thus, $p_v^{H_T}(\vec{x}) \leq p_{\vec{x}}$ for all

$\vec{x}$. Since both $\sum_{\vec{x}} p_v^{H_T}(\vec{x})$ and $\sum_{\vec{x}} p_{\vec{x}}$ are 1, it must be the case that $p_v^{H_T}(\vec{x}) = p_{\vec{x}}$ for all $\vec{x}$. This shows that for every history $\vec{h}_T$ of the adversary, the distribution of possible inputs of honest players conditional on $\vec{h}_T$ depends only on their inputs and what honest players output. By Lemma 22, this implies that the input of honest players and the history of the adversary are independent (given the input $\vec{x}_T$ of the adversary and the output v of honest players), and thus, again by Lemma 22, it follows that the distribution of possible histories of the adversary depends only on $\vec{x}_T$ and v. This shows that every possible output function of the adversary can be simplified to a function that has as inputs only $\vec{x}_T$ and v, as desired. The argument for $k = 0$ is analogous, except that in this case, if the distribution of possible histories of the adversary is not independent of the inputs of the honest players, the adversary decreases the payoffs of the honest players, rather than increasing the payoffs of the deviating players. This completes the proof of Theorem 19.

Note that a (k,t)-robust implementation of $\vec{\sigma} + \sigma_{ACT}$ may not necessarily (non-strictly) $(k+t)$-securely compute f, since if the adversary consists of fewer than $k+t$ malicious players, the malicious players might be able to deduce information about the honest players' inputs without being able to take advantage of it (recall that a subset K consisting of $k+t$ players must all guess the same value for $u_i^2(\vec{a},\vec{x})$ to be non-zero). However, a small variation in the construction of u_i^2 in $\Gamma_d^{f,k,t}$ allows us to construct a game $\Gamma_d^{f,k}$ such that any strongly $(k,0)$-robust implementation of the strategy used in Proposition 32 also k-securely computes f, so secure computation can be reduced to implementing strategies for certain mediator games. The idea is that instead of requiring the subset K of players who do not play G to have size exactly k, we only require it to have size at most k. This modification of u_i^2 leads to some of the problems discussed at the beginning of this section, namely, that if some rational players act like honest players, except

that they share their inputs with other rational players, the latter players might be able to guess the input profile and get a strictly positive expected payoff. This scenario is indistinguishable from one in which the players who shared their input are actually honest and rational players are just lucky. To deal with this issue, we further modify the payoffs in $\Gamma^{f,k}$ so that if the players in K guess the inputs correctly, then everyone else gets a huge negative payoff (rather than 0, as in the original construction). We can show that if this payoff is sufficiently small (e.g., $-n$ times the winnings), then if there exists a strategy in which rational players get a positive payoff from u^2, then there exists a strategy in which rational players get a positive payoff from u^2 and they all guess the same value in every possible history (if the negative payoffs are small enough, rational players not guessing any value gives a negative total payoff for rational players, regardless if some of them guess the correct value).

Note that this modification works only for strong (k,t)-robustness, since if we require only (k,t)-robustness, a rational player may decrease its own payoff if that helps other rational players, even if the total gain from doing so is negative. This is enough to show that the strategy used in Proposition 32 is strongly $(k,0)$-robust with these payoffs.

6.5 Proof of Theorem 22

Consider the game $\Gamma^{k,t}$ in which the set of actions of each player is $\{G,R\}\times\{0,1\}$. Given an action profile $\vec{a}$, in which each player i plays $a_i = (Q_i, y_i)$ with $Q_i \in \{G,R\}$ and $y_i \in \{0,1\}$, let T be the subset of players i such that $Q_i = R$. If $|T| > k+t$, if $k=0$ all players get a payoff of -1, otherwise all players get a payoff

of 1. If $|T| = t + k$ and there exist two players $i, j \notin T$ such that $y_i \neq y_j$, if $k = 0$ all players get a payoff of -1, otherwise all players get a payoff of 1. In all remaining cases, all players get a payoff of 0. Let g be the function such that $g(Q, y) = y$.

Consider the following protocol $\vec{\sigma} + \sigma_d$ for n players and a mediator. With σ_i, each player i sends the mediator its input x_i the first time it is scheduled. The mediator waits until receiving a message containing either 0 or 1, and sends that value y to all players. The players play (G, y) whenever they receive y from the mediator. Clearly, this strategy is (k, t)-robust (resp., strongly (k, t)-robust), since the only way that players get a payoff other than 0 with an adversary of size at most $k + t$ is if two honest players output different values, but they all receive the same value from the mediator. Suppose a strategy $\vec{\sigma}_{ACT}$ is a (k, t)-robust (resp., strongly (k, t)-robust) implementation of $\vec{\sigma} + \sigma_d$. We show next that (a) for all adversaries $A = (T, \vec{\tau}_T, \sigma_e)$ with $|T| \leq k + t$, all honest players play the same value y_i, and (b) if all players are honest and have the same input x, then they output x.

Property (b) follows trivially from the fact that $\vec{\sigma}_{ACT}$ implements $\vec{\sigma} + \sigma_d$: if all players are honest and have the same input x, the value received by the mediator in $\vec{\sigma} + \sigma_d$ is guaranteed to be x, and thus, in $\vec{\sigma} + \sigma_d$, all honest players play (G, x).

To prove (a), suppose that there exists an adversary $A = (T, \vec{\tau}, \sigma_e)$ with $|T| \leq k + t$ such that, in some history H of $\vec{\sigma}_{ACT}$ with A, there exist two players $i, j \notin T$ that play (Q_i, y_i) and (Q_j, y_j), respectively, with $y_i \neq y_j$, $Q_i = R$, or $Q_j = R$. Consider an adversary $A' = (T', \vec{\tau}_{T'}, \sigma_e)$ such that $|T'| = k + t$, $T \subseteq T'$, and $i, j \notin T'$ (we know that such a subset T' exists, since $n > t + k + 1$), and such that players in T act as in $\vec{\tau}_T$ and players in $T' - T$ act like honest players, except that all of them play $(R, 0)$. Since histories generated by playing with A and A' are

indistinguished by honest players, there exists a history H' in $\vec{\sigma}_{ACT}$ with adversary A' in which all honest players send and receive the same messages, and perform the same actions. If $Q_i = R$ in H, then there are $k + t + 1$ players that play R in H': the $k + t$ players in T' and i. Thus, all players get a payoff of 1 if $k > 0$, contradicting the assumption that $\vec{\sigma}_{ACT}$ is (k, t)-resilient, or all players get a payoff of -1 if $k = 0$, contradicting the assumption that $\vec{\sigma}_{ACT}$ is t-immune. The same argument shows that $Q_i = G$ in H and H' and, indeed, that all honest players must play G in H and H'. Now if $q_i \neq q_j$ in H, then $q_i \neq q_j$ in H', so (since all honest players play G, so exactly $k + t$ players in H' play R), again, all players in H' get a payoff of 1 if $k > 0$ and a payoff of -1 if $k = 0$, so we again get the same contradiction as before.

CHAPTER 7

CONCLUSION

In chapter 3 we proved that we can t-bisimulate any interaction of n players with a trusted mediator if $n > 4t$. This means that no adversary that controls at most t players can subvert the computation as long as the remaining players are honest. We also showed that some security properties are still preserved if $3t < n \leq 4t$. In particular, even if the adversary can halt the computation, it cannot induce honest players to output a wrong result.

In chapter 4 we used the results of chapter 3 to show that any (k, t)-robust strategy $\vec{\sigma} + \sigma_d$ with a mediator can be implemented without the mediator if $n > 4k + 4t$. This means that no coalition of t malicious players can decrease the payoff of any of the remaining players, and no coalition of k players can increase their payoff by deviating, even when taking advantage of the malicious behavior of the t other players. If there exists a punishment strategy, we show that we can obtain a (k, t)-robust implementation of $\vec{\sigma} + \sigma_d$ if $n > 3k + 4$ since honest players can threaten rational players by playing their part of the punishment strategy if they are caught deviating. If we allow an arbitrarily small probability of error, we get better results. In this case the upper bounds reduce to $n > 3k + 3t$ and $n > 2k + 3t$ depending if there is or not a punishment strategy for honest players. We also show that we can get the $n > 2k + 3t$ bound if the system is synchronous and there is a punishment strategy.

In chapter 5 we extend the results in Chapter 4 and show that any k-resilient sequential with a mediator can be implemented without the mediator if $n > 3k$ in synchronous systems or $n > 4k$ in asynchronous systems. To perform such extension, we proved the existence of k-paranoid beliefs for any protocol $\vec{\pi}$ for n

players and all $k < n$. Intuitively, if the protocol tolerates deviations of coalitions up to size k and players have a k-paranoid belief system, all such coalitions always believe that the remaining players are carrying out all computations by themselves, and thus that the outcome of the protocol is independent of their actions.

Finally, in Chapter 6 we prove lower bounds that match the upper bounds of the previous chapters with the exception of the $n > 3k + 4t$ upper-bound for (k, t)-bisimulation when there exists a punishment strategy in an asynchronous system. This is still an open problem. Besides this lower bound, there are other open questions. For instance, none of our results assume the existence of cryptographic primitives. We believe that if we assume the existence of one-way functions, most of our results can be improved at the expense of leaving a small probability of error. We also assume that the set of players is fixed and all pairs of players are always connected by an authenticated private channel. It would be interesting to see what results do we get if we consider networks with different topologies (e.g., permissionless networks, unauthenticated channels, etc.). We hope to give answer to some of these questions in subsequent work.

BIBLIOGRAPHY

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

[2] I. Abraham, D. Dolev, and J. Y. Halpern. An almost-surely terminating polynomial protocol for asynchronous Byzantine agreement with optimal resilience. In *Proc. 27th ACM Symposium on Principles of Distributed Computing*, pages 61–75, 2008.

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

[4] I. Abraham, D. Dolev, and J. Y. Halpern. Distributed protocols for leader election: a game-theoretic perspective. In *Proc. 27th International Symposium on Distributed Computing*, pages 61–75, 2013.

[5] I. Abraham, D. Dolev, and G. Stern. Revisiting asynchronous fault tolerant computation with optimal resilience. pages 139–148, 2020.

[6] E. Adar and B. Huberman. Free riding on Gnutella. *First Monday*, 5(10), 2000.

[7] R. J. Aumann. Correlated equilibrium as an expression of Bayesian rationality. *Econometrica*, 55:1–18, 1987.

[8] R. J. Aumann and S. Hart. Long cheap talk. *Econometrica*, 71(6):1619–1660, 2003.

[9] M. Ben-Or, R. Canetti, and O. Goldreich. Asynchronous secure computation. In *STOC '93: Proceedings of the 25 Annual ACM Symposium on Theory of Computing*, pages 52–61, New York, NY, USA, 1993. ACM Press.

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

[11] M. Ben-Or, S. Goldwasser, and A. Wigderson. Completeness theorems for

non-cryptographic fault-tolerant distributed computation. In *Proc. 20th ACM Symp. Theory of Computing*, pages 1–10, 1988.

[12] M. Ben-Or, B. Kelmer, and T. Rabin. Asynchronous secure computations with optimal resilience (extended abstract). In *Proc. 13th ACM Symp. Principles of Distributed Computing*, pages 183–192, 1994.

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

[14] G. Bracha. An asynchronous $[(n-1)/3]$-resilient consensus protocol. In *Proc. 3rd ACM Symposium on Principles of Distributed Computing*, pages 154–162, 1984.

[15] G. Bracha. Asynchronous Byzantine agreement protocols. *Information and Computation*, 75(2):130–143, 1987.

[16] R. Canetti. *Studies in Secure Multiparty Computation and Applications*. PhD thesis, Technion, 1996.

[17] K. Eliaz. Fault-tolerant implementation. *Review of Economic Studies*, 69(3):589–610, 2002.

[18] S. Even, O. Goldreich, and A. Lempel. A randomized protocol for signing contracts. *Commun. ACM*, 28(6):637–647, 1985.

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

[20] D. Gerardi. Unmediated communication in games with complete and incomplete information. *Journal of Economic Theory*, 114:104–131, 2004.

[21] D. Gerardi and R. B. Myerson. Sequential equilibria in Bayesian games with communication. *Games and Economic Behavior*, 60:104–134, 2007.

[22] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game. In *Proc. 19th ACM Symp. Theory of Computing*, pages 218–229, 1987.

[23] J. Y. Halpern and M. R. Tuttle. Knowledge, probability, and adversaries. *Journal of the ACM*, 40(4):917–962, 1993.

[24] D. M. Kreps and R. B. Wilson. Sequential equilibria. *Econometrica*, 50:863–894, 1982.

[25] L. Lamport. The weak Byzantine generals problem. *J. ACM*, 30(3):668–676, 1983.

[26] R. Myerson. Multistage games with communication. *Econometrica*, 54:323–358, 1986.

[27] A. Shamir. How to share a secret. *Communications of the ACM*, 22:612–613, 1979.

[28] A. Shamir, R. L. Rivest, and L. Adelman. Mental poker. In D. A. Klarner, editor, *The Mathematical Gardner*, pages 37–43. Prindle, Weber, and Schmidt, Boston, MA, 1981.

[29] A. Yao. Protocols for secure computation (extended abstract). In *Proc. 23rd IEEE Symp. Foundations of Computer Science*, pages 160–164, 1982.