Distributed Protocols for Leader Election: a Game-Theoretic Perspective

Ittai Abraham
Microsoft Research
ittaia@microsoft.com

Danny Dolev *
School of Computer Science and Engineering
The Hebrew University of Jerusalem
Jerusalem, Israel
dolev@cs.huji.ac.il

Joseph Y. Halpern †
Cornell University
Ithaca, NY 14850
halpern@cs.cornell.edu

Abstract

We do a game-theoretic analysis of leader election, under the assumption that each agent prefers to have some leader than to have no leader at all. We show that it is possible to obtain a *fair* Nash equilibrium, where each agent has an equal probability of being elected leader, in a completely connected network, in a bidirectional ring, and a unidirectional ring, in the synchronous setting. In the asynchronous setting, Nash equilibrium is not quite the right solution concept. Rather, we must consider *ex post* Nash equilibrium; this means that we have a Nash equilibrium no matter what a scheduling adversary does. We show that ex post Nash equilibrium is attainable in the asynchronous setting in all the networks we consider, using a protocol with bounded running time. However, in the asynchronous setting, we require that $n > 2$. We can get a fair ϵ-*Nash* equilibrium if $n = 2$ in the asynchronous setting, under some cryptographic assumptions (specifically, the existence of a pseudo-random number generator and polynomially-bounded agents), using ideas from bit-commitment protocols. We then generalize these results to a setting where we can have deviations by a coalition of size k. In this case, we can get what we call a fair k-resilient equilibrium if $n > 2k$; under the same cryptographic assumptions, we can a get a k-resilient equilibrium if $n = 2k$. Finally, we show that, under minimal assumptions, not only do our protocols give a Nash equilibrium, they also give a *sequential* equilibrium [Kreps and Wilson 1982], so players even play optimally off the equilibrium path.

*Danny Dolev is Incumbent of the Berthold Badler Chair in Computer Science. Part of the work was done while the author visited Cornell University. This research project was supported in part by The Israeli Centers of Research Excellence (I-CORE) program, (Center No. 4/11), by NSF, AFOSR grant FA9550-09-1-0266, and by the Google Inter-University Center for Electronic Markets and Auctions.

†Supported in part by NSF grants IIS-0534064, IIS-0812045, and IIS-0911036, IIS-0911036, and CCF-1214844, AFOSR grants FA9550-08-1-0438, FA9550-09-1-0266, and FA9550-12-1-0040, and ARO grant W911NF-09-1-0281.

1 Introduction

As has been often observed, although distributed computing and game theory are interested in much the same problems—dealing with systems where there are many agents, facing uncertainty, and having possibly different goals—in practice, there has been a significant difference in the models used in the two areas. In game theory, the focus has been on rational agents: each agent is assumed to have a utility on outcomes, and be acting so as to maximize expected utility. In distributed computing, the focus has been on the "good guys/bad guys" model. The implicit assumption here is that there is a system designer who writes code for all the processes in the system, but some of the processes may get taken over by an adversary, or some computers may fail. The processes that have not been corrupted (either by the adversary or because of a faulty computer) follow the designer's protocol. The goal has typically been to prove that the system designer's goals are achieved, no matter what the corrupted processes do.

More recently, there has been an interest in examining standard distributed computed problems under the assumption that the agents are *rational*, and will deviate from the designer's protocol if it is in their best interest to do so. [1] Halpern and Teague [2004] were perhaps the first to do this; they showed (among other things) that secret sharing and multiparty communication could not be accomplished by protocols with bounded running time, if agents were using the solution concept of iterated admissibility (i.e., iterated deletion of weakly dominated strategies). Since then, there has been a wide variety of work done at the border of distributed computing and game theory. For one thing, work has continued on secret sharing and multiparty computation, taking faulty and rational behavior into account (e.g., [Abraham, Dolev, Gonen, and Halpern 2006; Dani, Movahedi, Rodriguez, and Saia 2011; Fuchsbauer, Katz, and Naccache 2010; Gordon and Katz 2006; Lysyanskaya and Triandopoulos 2006]). There has also been work on when and whether a problem that can be solved with a trusted third party can be converted to one that can be solved using *cheap talk*, without a third party, a problem that has also attracted the attention of game theorists (e.g., [Abraham, Dolev, Gonen, and Halpern 2006; Abraham, Dolev, and Halpern 2008; Barany 1992; Ben-Porath 2003; Dodis, Halevi, and Rabin 2000; Forges 1990; Heller 2005; Izmalkov, Lepinski, and Micali 2011; Lepinski, Micali, Peikert, and Shelat 2004; McGrew, Porter, and Shoham 2003; Shoham and Tennenholtz 2005; Urbano and Vila 2002; Urbano and Vila 2004]). This is relevant because there are a number of well-known distributed computing problems that can be solved easily by means of a "trusted" mediator. For example, if fewer than half the agents are corrupted, then we can easily do Byzantine agreement with a mediator: all the agents simply tell the mediator their preference, and the mediator chooses the majority. Another line of research was initiated by work on the *BAR* model [Aiyer, Alvisi, Clement, Dahlin, Martin, and Porth 2005]; see, for example, [Moscibroda, Schmid, and Wattenhofer 2006; Wong, Levy, Alvisi, Clement, and Dahlin 2011]. Like the work in [Abraham, Dolev, Gonen, and Halpern 2006; Abraham, Dolev, and Halpern 2008], the BAR model allows Byzantine (or faulty) players and rational players; in addition, it allows for *acquiescent* players, who follow the recommended protocols.[2] Traditional game theory can be viewed as allowing only rational players, while traditional distribution computing considers only acquiescent and Byzantine players.

In this paper, we try to further understand the impact of game-theoretic thinking on standard problems in distributed computing. We consider the classic distributed computing problem of electing a leader in an anonymous network (a network where, initially, each process knows its own name, but does not know the name of any other process). Leader election is a fundamental problem in distributed computing. Not surprisingly, there are numerous protocols for this problem (see, e.g., [Chang and Roberts 1979; Dolev, Klawe, and Rodeh 1982; Le Lann 1977; Lynch 1997; Peterson 1982]) if we assume that no agents have

[1] We use "process" and "agent" in this paper to describe the entities in the system, trying to use "agent" when we think of them as rational, and "process" when they are just computers running a protocol, with no preferences.

[2] Originally, the "A" in "BAR" stood for *altruistic*, but it was changed to stand for "acquiescent" [Wong, Levy, Alvisi, Clement, and Dahlin 2011].

been corrupted; there have also been extensions that deal with corrupted agents [Feldman and Micali 1997; Katz and Koo 2006]. Much of this work focuses on leader election in a ring (e.g., [Chang and Roberts 1979; Dolev, Klawe, and Rodeh 1982; Le Lann 1977; Lynch 1997; Peterson 1982]).

In this paper we study what happens if we assume that agents are rational. It is easy to show that if all agents (a) prefer to have a leader to not having a leader and (b) are indifferent as to who is the leader, then all the standard distributed computing protocols work without change. This can be viewed as formalizing the intuition that in the standard setting in distributed computing, we are implicitly assuming that all the agents share the system designer's preferences. But what happens if the agents have different preferences regarding who becomes the leader? For example, an agent may prefer that he himself becomes the leader, since this may make the cost of routing to other agents smaller. In this case, the standard protocols (which typically assume that each agent has a distinct id, and end up electing the agent with the lowest id, or the agent with the highest id, as the leader) do not work; agents have an incentive to lie about their id. Nevertheless, there is always a trivial Nash equilibrium for leader election: no one does anything. Clearly no agent has any incentive to do anything if no one else does. We are thus interested in obtaining a *fair* Nash equilibrium, one in which each agent has an equal probability of being elected leader. Moreover, we want the probability that someone will be elected to be 1.[3] In the language of the BAR model, we allow acquiescent and rational players, but not Byzantine players.

It is easy to solve leader election with a mediator: the agents simply send the mediator their ids, and the mediator picks an id at random as the leader and announces it to the group. We cannot immediately apply the ideas in the work on solving the problem with a mediator and then replacing the mediator with cheap talk to this problem because all these results assume (a) that agents have commonly-known names, (b) that the network is completely connected, and (c) the network is synchronous. Nevertheless, we show that thinking in terms of mediators can be helpful in deriving a simple protocol in the case of a completely connected network that is a fair Nash equilibrium in which a leader is elected with probability 1. We can then modify the protocol so that it works when the network is a ring. We also show that our protocol is actually k-*resilient* [Abraham, Dolev, Gonen, and Halpern 2006; Abraham, Dolev, and Halpern 2008]: it tolerates coalitions of size k, as long as $n > k$. This forms an interesting contrast to work on Byzantine agreement, where it is known that the network must be $2k + 1$ connected to tolerate k Byzantine failures [Dolev 1982]. But we can tolerate coalitions of k rational players even in a unidirectional ring.

These protocols work if the network is synchronous. What happens in an asynchronous setting? Before answering this question, we need to deal with a subtlety: what exactly a Nash equilibrium is in an asynchronous setting? To make sense of Nash equilibrium, we have to talk about an agent's best response. An action for an agent i is a best response if it maximizes i's expected utility, given the other agents' strategies. But to compute expected utility, we need a probability on outcomes. In general, in an asynchronous setting, the outcome may depend on the order that agents are scheduled and on message-delivery times. But we do not have a probability on these. We deal with these problems in this setting by using the standard approach in distributed computing. We assume that an adversary chooses the scheduling and chooses message-delivery times, and try to obtain a strategy that is a Nash equilibrium no matter what the adversary does. This intuition gives rise to what has been called in the literature an *ex post Nash equilibrium*. We provide a simple protocol that gives a fair ex post Nash equilibrium provided that $n > 2$. More generally, we provide a fair ex post k-resilient equilibrium as long as $n > 2k$. We then show that these results are optimal: there is no

[3]Without the last requirement, the existence of a fair Nash equilibrium follows from well-known results, at least in the case of a completely connected network. We can model our story as a symmetric game, one where all agents have the same choice of actions, and an agent's payoff depends only on what actions are performed by others, not who performs them. In addition to showing that every game has a Nash equilibrium, Nash also showed that a symmetric game has a symmetric Nash equilibrium, and a symmetric equilibrium is clearly fair. However, in a symmetric equilibrium, it may well be the case that there is no leader chosen. For example, a trivial symmetric equilibrium for our game is one where everyone chooses a candidate leader at random. However, in most cases, agents choose different candidates, so there is no leader.

fair k-resilient ex post Nash equilibrium if $n \leq 2k$.

The lower bounds assume that agents are not computationally bounded. If we assume that agents are polynomially-bounded (and make a standard assumption from the cryptography literature, namely, that a pseudorandom number generator exists), then we can show, using ideas of Naor [1991], that there is a fair ex post ϵ-*Nash* equilibrium in this case (one where agents can gain at most ϵ by deviating) for an arbitrarily small ϵ; indeed, we can show that there is a fair ex post ϵ–k-resilient equilibrium as long as $n > k$.

As the title of the paper suggests, this is meant to be a case study, to illustrate some issues that might arise when trying to apply game-theoretic approaches to distributed computing problems. We have certainly not tried to see what happens in all variants of leader election, nor have we tried to get protocols that are optimal in terms of time, space, or message complexity. While we believe that our techniques for leader election can be used in other topologies than the ones that we have considered, we have not explored that either. Rather, our goal has been to understand the impact of dealing with rational, self-interested agents on standard distributed protocols such as leader election.

Finally, we show that, under minimal assumptions, not only do our protocols give a Nash equilibrium, they also give a *sequential* equilibrium [Kreps and Wilson 1982], so players even play optimally off the equilibrium path.

The rest of this paper is organized as follows. In the next section, we briefly review the model that we are working in. We present our protocols for achieving fair Nash equilibrium in Section 3. These protocols involve what are arguably *incredible threats*: once a player detects a problem, he simply refuses to elect a leader, thus punishing himself, as well as the perpetrator of the problem. However, we show in the full paper that by adding one plausible assumption to the utility function, our Nash equilibrium actually becomes a *sequential equilibrium* [Kreps and Wilson 1982], a solution concept that is meant to handle incredible threats. We conclude with some discussion in Section 4.

2 The Model

We model a network as a directed, simple (so that there is at most one edge between each pair of nodes), strongly connected, and finite graph. The nodes represent agents, and the edges represent communication links. We assume that the topology of the network is common knowledge, so that if we consider a completely connected network, all agents know that the network is completely connected, and know that they know, and so on; this is similarly the case when we consider unidirectional or bidirectional rings. Deviating agents can communicate only using the network topology; there is no "out of band" communication. We assume that, with each agent, there is associated a unique id, taken from some commonly-known name space, which we can take to be a set of natural numbers. Initially agents know their ids, but may not know the id of any other agent. For convenience, if there are n agents, we name them $1, \ldots, n$. These are names used for our convenience when discussing protocols (so that we can talk about agent i); these names are not known by the agents. Message delivery is handled by the channel (and is not under the control of the agents). Agents can identify on which of their incoming links a message comes in, and can distinguish outgoing links.

When we consider synchronous systems, we assume that agents proceed in lockstep. In round m, (if $m > 0$) after all messages sent in round $m - 1$ are received by all agents, agents do whatever internal computation they need to do (including setting the values of variables); then messages are sent (which will be received at the beginning of round $m + 1$). In the asynchronous setting, agents are scheduled to move at arbitrary times by a (possibly adversarial) scheduler. When they are scheduled, they perform the same kinds of actions as in the synchronous case: receive some messages that were sent to them earlier and not yet received, do some computation, and send some messages. For ease of exposition, we assume that the message space is finite. While we assume that all messages sent are eventually received (uncorrupted), there is no bound on message delivery time. Nor do we make any assumption on the number of times one agent

can be scheduled relative to another, although we do assume that agents are scheduled infinitely often (so that, for all agents i and times t, there will be a time after t when i is scheduled).

Since we are interested in leader election, we assume that each agent i has a variable $leader_i$ which can be set to some agent's id. If, at the end of the protocol, there is an id v such that $leader_i = v$ for all agents i, then we say that the agent with id v has been elected leader. Otherwise, we say that there is no leader. (Note that we are implicitly requiring that, when there is a leader, all the players know who that leader is.) We assume that each agent i has a utility on outcomes of protocols. For the purposes of this paper, we assume that agents prefer having a leader to not having one, in the weak sense that each agent i never assigns a higher utility to an outcome where there is no leader than to one in which there is a leader (although we allow the agent to be indifferent between an outcome where there is no leader and an outcome where there is a leader). We make no further assumptions on the utility function. It could well be that i prefers that he himself is the leader rather than anyone else; i could in addition prefer a protocol where he sends fewer messages, or does less computation, to one where he sends more messages or does more computation. Nevertheless, our eassumptions require that player i never prefers an outcome where there is no leader to one where there is, even if the latter outcome involves sending many messages and a great deal of computation (although in fact our protocols are quite message-efficient and do not require much computation). Note that our assumptions imply that agent i can "punish" other agents by simply setting $leader_i$ to $\perp$; this ensures that there will be no leader. In the language of [Ben-Porath 2003], this means that each agent has a *punishment strategy*.

A strategy profile (i.e., a strategy or protocol for each agent) is a *Nash equilibrium* if no agent can unilaterally increase his expected utility by switching to a different protocol (assuming that all the other agents continue to use their protocols). It is easy to see that if all the agents are indifferent regarding who is the leader (i.e., if, for each agent i, i's utility of the outcome where j is the leader is the same for all j, including $j = i$), then any protocol that solves leader election is a Nash equilibrium. Note that it is possible that one Nash equilibrium *Pareto dominates* another: all agents are better off in the first equilibrium. For example, if agents are indifferent about who the leader is, so that any protocol that solves leader election is a Nash equilibrium, all agents might prefer an equilibrium where fewer messages are sent; nevertheless, a protocol for leader election where all agents send many messages could still be a Nash equilibrium.

For the remainder of this paper, we assume that each agent has a *preference for leadership*. Formally, this means that : agent i's utility function is such that i does not give higher utility to an outcome where there is no leader than to one where there is a leader. (Agent i may also prefer to be the leader himself, or have preferences about which agent j is the leader if he is not the leader, of course. The details of these preferences do not play a

3 The protocols

We consider protocols in three settings: a completely connected network, a unidirectional ring, and a bidirectional ring. We also consider both the synchronous case and the asynchronous case.

3.1 Completely connected network, synchronous case

Consider leader election in a completely connected network. First suppose that we have a mediator, that is, a trusted third party. Then there seems to be a naive protocol that can be used: each agent tells the mediator his id, then the mediator picks the highest id, and announces it to all the agents. The agent with this id is the leader. This naive protocol has two obvious problems. First, since we assume that the name space is commonly known, and all agents prefer to be the leader, agents will be tempted to lie about their ids, and to claim that the highest id is their id. Second, even if all agents agree that an agent with a particular id v is the leader, they don't know which agent has that id.

We solve the first problem by having the mediator choose an id at random; we solve the second problem by having agents share their ids. In more detail, we assume that in round 1, agents tell each other their ids.

In round 2, each agent tells the mediator all the set of ids he has heard about (including his own). In round 3, the mediator compares all the sets of ids. If they are all the same, the mediator chooses an id v at random from the set; otherwise, the mediator announces "no leader". If the mediator announces that v is the leader, each agent i sets $leader_i = v$ (and marks the incoming link on which the id v was originally received); otherwise, $leader_i$ is undefined (and there is no leader).

It is easy to see that everyone using this protocol gives a Nash equilibrium. If some agent does not send everyone the same id, then the mediator will get different lists from different agents, and there will be no leader. And since a leader is chosen at random, no one has any incentive not to give his actual id. Note that this protocol is, in the language of [Abraham, Dolev, Gonen, and Halpern 2006; Abraham, Dolev, and Halpern 2008], *k-resilient* for all $k < n$, where n is the number of agents. That is, not only is it the case that no single agent has any incentive to deviate, neither does any coalition of size k. Moreover, the resulting Nash equilibrium is *fair*: each agent is equally likely to be the chosen leader.

Now we want to implement this protocol using cheap talk. Again, this is straightforward. At round 1, each agent i sends everyone his id; at round 2, i sends each other agent j the set of ids that he (i) has received (including his own). If the sets received by agent i are not all identical or if i does not receive an id from some agent, then i sets $leader_i$ to $\perp$, and leader election fails. Otherwise, let n be the cardinality of the set of ids. Agent i chooses a random number N_i in $\{0, \ldots, n-1\}$ and sends it to all the other agents. Each agent i then computes $N = \sum_{i=1}^{n} N_i \pmod n$, and then takes the agent with the Nth highest id in the set to be the leader. (If some agent j does not send i a random number, then i sets $leader_i = \perp$.) Call this protocol for agent i $LEAD_i^{cc}$. The formal pseudocode of the protocol appears as Protocol 1 in Appendix C. Let $\mathbf{LEAD}^{cc}$ denote the profile $(LEAD_1^{cc}, \ldots, LEAD_1^{cc})$ (we use boldface for profiles throughout the paper). Clearly, with the profile $\mathbf{LEAD}^{cc}$, all the agents will choose the same leader. It is also easy to see that no agent (and, indeed, no group of size $k < n$) has any incentive to deviate from this strategy profile.

We summarize these observation with the following theorem:

Theorem 3.1 *$\mathbf{LEAD}^{cc}$ is a fair, k-resilient equilibrium in a completely connected network of n agents, for all $k < n$.*[4]

Proof: First suppose that all agents use the protocol. Then clearly each agent is equally likely to become the leader. At the risk of belaboring the argument, it is worth bringing out the key ideas, since they are common to all our later proofs. First, what is necessary is a common naming system, so if the agents do decide on agent i as the leader, they all agree on who i is. The first two steps do that. When all the agents exchange their ids (provided that each agent sends the same id to all the other agents), they end up agreeing on what the names are and who has each name. Next, the agents need a common way of ordering themselves, so that when they choose a number $j \pmod n$ at random, the agents agree on who the jth agent is. Once we have all the ids, we can order the ids. Finally, the agents need a way of choosing a common number j at random. The last steps do that.

It now suffices to show that no group of $k < n$ agents has any incentive to deviate. Suppose that a group H of agents with $|H| < n$ agents all deviate, while no other agents deviate. If any of the agents in H send different ids to an agent not in H at step 1, then each agent j not in H will receive different sets in step 2, and thus set $leader_j = \perp$, making the agents in H no better off than if they do not deviate. (We still need to argue that agents in H send their actual ids; see below.) If an agent in H deviates at step 2 by sending different random numbers to agents not in H, then the agents not in H will disagree about the leader, again making i no better off. If an agent $i \in H$ deviates by not taking the agent with the Nth highest id to be the leader, then again agents in H will disagree with the agents not in H about the leader, making i no better off. It is clear that as long as all agents not in H follow the protocol, and each $i \in H$ sends each agent not in

[4] All proofs can be found in Appendix C.

H the same id in step 1 and the same number (not necessarily random) in step 2, then each agent is equally likely to be the leader. Thus, an agent $i \in H$ does not gain by deviating in step 1 by sending an id other than his actual id, nor does he gain by choosing a number in step 2 other than a random number. We have thus shown that all agents following the protocol is indeed a fair, k-resilient equilibrium. ∎

Note that we have implicitly assumed no *rushing* here. That is, we assume that all agents send simultaneously; an agent cannot delay sending his message until after he has heard from everyone else. Otherwise, an agent i will wait until he hears from everyone before choosing N_i in such a way to guarantee that he is the leader. Below we show how to extend this protocol to the asynchronous setting, where, of course, rushing is allowed.

Up to now we have implicitly assumed that each agent somehow gets a signal regarding when to start the protocol. This assumption is unnecessary. Even if only some agents want to start the protocol, they send a special round 0 message to everyone asking them to start a leader election protocol. The protocol then proceeds as above.

3.2 Unidirectional ring, synchronous case

We give a Nash equilibrium for leader election in a unidirectional ring, under the assumption that the ring size n is common knowledge. (We discuss in Section 4 why this assumption is necessary.) This assumption is necessary, for otherwise an agent can create k sybils, for an arbitrary k, and pretend that the sybils are his neighbors. That is, i can run the protocol as if the ring size is $n + k$ rather than n, simulating what each of his sybils would do. No other agent can distinguish the situation where there are n agents and one agent has created k sybils from a situation where there are actually $n + k$ agents. Of course, if any of i's sybils are elected, then it is as if i is elected. Thus, creating sybils can greatly increase i's chances of being elected leader, giving i an incentive to deviate. (However, the overhead of doing may be sufficient to deter an agent from doing so. See the discussion in Section 4.) Note that in the case of a completely connected network, given that the topology is common knowledge, the number of agents is automatically common knowledge (since each agent can tell how many agents he is connected to).

The protocol is based on the same ideas as in the completely connected case. It is easy to ensure that there is agreement among the agents on what the set of agents is; implementing a random selection is a little harder. We assume that the signal to start leader election may come to one or more agents. Each of these agents then sends a "signed" message (i.e., a message with his id) to his neighbor. Messages are then passed around the ring, with each agent, appending his id before passing it on. If an agent receives a second message that originated with a different agent, the message is ignored if the originating agent has a lower id; otherwise it is passed on. Eventually the originator of the message with the highest id gets back the message. At this point, he knows the ids of all the agents. The message is then sent around the ring a second time. Note that when an agent gets a message for the second time, he will know when the message should make it back to the originator (since the system is synchronous and he knows the size of the ring).

At the round when the originator gets back the message for the second time, each agent i chooses a random number $N_i < n$ and sends it around the ring. After n rounds, all agents will know all the numbers $N_1, \ldots, N_n$, if each agent indeed sent a message. They can then compute $N = \sum_{i=1}^{n} N_i \pmod n$, and take the agent with the Nth highest id in the set to be the leader. If agent i does not receive a message when he expects to, then he aborts, and no leader is elected. For example, if an agent who originated a message does not get his message back n rounds and $2n$ rounds after he sent it, or gets a message from an originator with a lower id, then he aborts. Similarly, if an agent who forwarded an originator's message does not get another message from that originator n rounds later or get a message from another originator with a lower id, then he aborts. Finally, for each of the n rounds after the originator with the highest id gets back his message for the second time, each agent i should get a random number from the appropriate agent (i.e., k rounds after

the originator with the highest id gets back his message for the second time, agent i should get agent j's random number, j is k steps before i on the ring). If any of these checks is not passed, then i aborts, and no leader is chosen. Call this protocol for agent i $LEAD_i^{uni}$. The formal pseudocode of the protocol appears as Protocol 2 in Appendix D.

We would now like to show that $\mathbf{LEAD}^{uni}$ gives a k-resilient fair Nash equilibrium. But there is a subtlety, which we already hinted at in the introduction. In a Nash equilibrium, we want to claim that what an agent does is a best response to what the other agents are doing. But this implicitly assumes that the outcome depends only on the strategies chosen by the agents. But in this case, the outcome may in principle also depend on the (nondeterministic) choices made by nature regarding which agents get an initial signal. Thus, we are interested in what has been called an *ex post* Nash equilibrium. We must show that, no matter which agents get an initial signal, no agent has any incentive to deviate (even if the deviating agent knows which agents get the initial signal, and knows the remaining agents are playing their part of the Nash equilibrium). In fact, we show that no coalition of $k < n$ agents has any incentive to deviate, independent of nature's choices.

Theorem 3.2 $\mathbf{LEAD}^{uni}$ *is a fair, k-resilient (ex post) equilibrium in a unidirectional ring with n agents, for all $k < n$.*

Proof: Again, it should be clear that if each agent follows the protocol, then each agent is equally likely to become leader. Thus, it suffices to show that no group H of agents with $|H| < n$ has any incentive to deviate. If an agent $i \in H$ deviates by not sending a signed message to his neighbor when he gets a signal to start leader election, then either it has no impact (if other agents also get a signal) or no leader is elected when one could have been (so i is not better off). If i does not forward some message that i should forward or delays in forwarding a message, again, either it has no impact (if an agent with a lower id also gets a signal) or no leader is elected when one could have been (so i is not better off). If i modifies a message that it should forward, it may be the case that different agents not in H end up with different sets of ids, or different sets of n numbers. In the latter case, if the sum of the numbers in the set each agent considers possible is the same, then the change has no impact. Otherwise, different agents will have different views on who the leader should be—that is, there is no one leader elected. In the former case, the change has no impact; in the latter, it makes the deviator no better off. Similarly, if the different agents have view the set of ids as being different, either it has no impact, or else different agents will have different views of who is the leader. Moreover, if an agent $i \in H$ does not send a number $2n$ rounds after the "winning" originator sent his message to an agent not in H, then no leader is elected, and i is not better off. Thus, each agent $i \in H$ will send some id if he gets a signal at the beginning, and will send some number N_i at the appropriate time. Agent i does not gain by deviating in step 1 by sending an id other than his actual id, nor does he gain by choosing a number N_i other than a random number. We have thus shown that all agents following the protocol gives a fair, k-resilient equilibrium. ∎

3.3 Bidirectional ring, synchronous case

It is easy to see that the same protocol will work for the case of the bidirectional ring. More precisely, if there is agreement on the ring orientation, each agent implements the protocol above by just sending left, ignoring the fact that he can send right. If there is no agreement on orientation, then each originating agent can just arbitrarily choose a direction to send; each agent will then continue forwarding in the same direction (by forwarding the message with his id appended to the neighbor from which he did not receive the message). The originator with the highest id will still be the only one to receive his original message back. At that point the protocol continues with round 2 of the protocol for the unidirectional case, and all further messages will be sent in the direction of the original message of the originator with the highest id. Since it is only in the

second round that agents append their random numbers to messages, what happened in the first round has no effect on the correctness of the algorithm; we still get a Nash equilibrium as before.

3.4 Asynchronous Ring

We now consider an asynchronous setting. It turns out to be convenient to start with a unidirectional ring, then apply the ideas to a bidirectional ring. For the unidirectional ring, we can find a protocol that gives an ex post Nash equilibrium provided that there are at least 3 agents in the ring.

Consider the following protocol. It starts just as the protocol for the unidirectional case in the synchronous setting. Again, we assume that the signal to start a leader election may come to one or more agents. Each of these agents then sends a message with his id to his neighbor. Messages are then passed around the ring, with each agent appending his id before passing it on. If an agent receives a second message that originated with a different agent, the message is ignored if the originating agent has a lower id; otherwise it is passed on. Eventually the originator of the message with the highest id gets back the message. The originator checks to make sure that the message has n (different) ids, to ensure that no "bogus" ids were added. The message is then sent around the ring a second time. When an agent i gets the message the second time, he chooses a random number N_i mod n and sends it to his neighbor (as well as passing on the list of names). Agent i's neighbor does not pass on N_i; he just keeps it. Roughly speaking, by sending N_i to his neighbor, agent i is committing to the choice. Crucially, this commitment must be made before i knows any of the random choices other than that of the agent j of whom i is the neighbor (if i is not the originator). When the originator gets the message list for the second time (which means that it has gone around the ring twice), he sends it around the ring the third time. This time each agent i adds his random choice N_i to the list; agent i's neighbor j checks that the random number that i adds to the list is the same as the number that i sent j the previous time. When the originator gets back the list for the third time, it now includes each agent i's random number. The originator then sends the list around the ring for a fourth time. After the fourth time around the ring, all agents know all the random choices. Each agent then computes $N = \sum_{i=1}^{n} N_i$ (mod n), and then takes the agent with the Nth highest id in the set to be the leader. Each time an agent i gets a message, he checks that it is compatible with earlier messages that he has seen; that is, the second time he gets the message, all the ids between the originator and his id must be the same; the third time he gets the message, all the ids on the list must be the same as they were the second time he saw the message; and the fourth time he gets the message, not only must the list of ids be the same, but all the random choices that he has seen before can not have been changed. If the message does not pass all the checks, then agent i sets $leader_i$ to $\perp$. The formal pseudocode of the protocol appears as Protocol 3 in Appendix D.

Clearly this approach will not work with two agents: The originator's neighbor will get the originator's random choice before sending his own, and can then choose his number so as to ensure that he becomes leader. (We discuss how this problem can be dealt with in Section 3.7.) As we now show, this approach gives a fair ex post Nash equilibrium provided that there are at least three agents. In an asynchronous setting, nature has much more freedom than in the synchronous setting. Now the outcome may depend not only on which agents get an initial signal, but also on the order in which agents are scheduled and on message delivery times. Ex post equilibrium implicitly views all these choices as being under the control of the adversary; our protocol has the property that, if all agents follow it, the distribution of outcomes is independent of the adversary's choices. However, for the particular protocol we have given, it is easy to see that, no matter what choices are made by the adversary, we have a Nash equilibrium. While considering ex post Nash equilibrium seems like a reasonable thing to do in asynchronous systems (or, more generally, in settings where we can view an adversary as making choices, in addition to the agents making choices), it is certainly not the only solution concept that can be considered. (See Section 4.)

What about coalitions? Observe that, for the protocol we have given, a coalition of size two does have an incentive to deviate. Suppose that i_1 is the originator of the message, and i_1 is i_2's neighbor (so that i_2

will be the last agent on the list originated by i_1). If i_1 and i_2 form a coalition, then i_2 does not have to bother sending i_1 a random choice on the second time around the ring. After receiving everyone's random choices, i_2 can choose N_{i_2} so that he (or i_1) becomes the leader. This may be better for both i_1 and i_2 than having a random choice of leader.

We can get a protocol that gives a k-resilient (ex post) Nash equilibrium if $n > 2k$. We modify the protocol above by having each agent i send his random choice k steps around the ring, rather than just one step (i.e., to his neighbor). This means that i is committing N_i to k other agents. In more detail, we start just as with the protocol presented earlier. Each agent who gets a signal to start the protocol sends a message with his id to his neighbor. The messages are then passed around the ring, with each agent appending his id. If an agent receives a second message that originated with a different agent, the message is ignored if the originating agent has a lower id; otherwise it is passed on. Eventually the originator of the message with the highest id gets back the message. The originator checks to make sure that the message has n ids, to ensure that no "bogus" ids were added. The message is then sent around the ring a second time; along with the message, each agent i (including the sender) sends a random number N_i. Agent i's neighbor does not pass on N_i; he just keeps it, while forwarding the list of ids. When the originator gets the message the third time, he forwards to his neighbor the random number he received in the previous round (which is the random number generated by his predecessor on the ring). Again, his neighbor does not forward the message; instead he sends to his successor the random number he received (from the originator) on the previous round. At the end of this phase, each agent knows his random id and that of his predecessor. We continue this process for k phases altogether. That is, to when the originator gets a message for the third time, he sends this message (which is the random number chosen by his predecessor's predecessor) to his successor. Whenever an agent gets a message, he forwards the message he received in the previous phases. At the end of the jth phase for $j \leq k$, each agent knows the random numbers of his j closest predecessors. After these k phases complete, the sender sends his random number to his neighbor; each agent then appends his id to the list, and it goes around the ring twice. Each agent checks that the random numbers of his k predecessors agree with what they earlier told him. At the end of this process, each agent knows all the random numbers. As usual, each agent then computes $N = \sum_{i=1}^{n} N_i \pmod n$ and chooses as leader the agent with the Nth highest id. This protocol requires a constant number of rounds around the circle; we do not try to optimize the number here.

Each time an agent i gets a message, he checks that that it is compatible with earlier messages that he has seen; that is, the second time he gets the message, all the ids between the originator and his id must be the same; the third time he gets the message, all the ids on the list must be the same as they were the second time he saw the message; and the fourth time he gets the message, not only must the list of ids be the same, but all the random choices that he has seen before can not have been changed. He also checks that he has gotten the random choices of his k predecessors on the ring. If the message does not pass all the checks, then agent i sets $leader_i$ to $\perp$. Call this protocol for agent i $A\text{-}LEAD_i^{uni}$.

We summarize the discussion above with the following theorem.

Theorem 3.3 *If $n > 2k$, then $\mathbf{A\text{-}LEAD}^{uni}$ is a fair, k-resilient ex post equilibrium in an asynchronous unidirectional ring.*

Proof: As usual, it is straightforward to confirm that if each agent follows the protocol, then each agent is equally likely to become leader, so again it suffices to show that if $n > 2k$, then no group H of agents with $|H| \leq k$ has any incentive to deviate, no matter what the adversary does. Again, we want to argue that there is no advantage for an agent $i \in H$ to deviate up to and including the point where he sends his random number N_i, because deviation either has no impact, or results in no leader being chosen. But after sending the random number, it is too late to bias the election of the leader.

As in the proof of Theorem 3.2, if an agent $i \in H$ deviates by not sending a signed message to his neighbor when he gets a signal to start leader election, then either it has no impact (if other agents also get a signal) or no leader is elected when one could have been (so i is not better off). If i does not forward some message that i should forward during the first two phases (when just the list of ids is going around the ring) then either it has no impact or no leader is elected when one could have been (so i is not better off). If i modifies a message that it should forward, it may be the case that different agents not in H end up with different sets of ids. As long as all agents agree on which id is the lowest, this will not make a difference. If they disagree on this, then we will not have a single leader. As before, this just shows that all the agents will send some id, not necessarily their own. But they clearly gain no benefit by not sending their own id.

Similarly, all the agents in H want to send messages in each of the k phases after the originator has received the list of ids for the second time, otherwise there will not be a leader elected. They must also send the messages in a consistent way, so that they pass all the checks. What the agents in H would really like to do is to arrange to have the sum $N = \sum_{i \in 1}^{n} N_i$ be such that N steps after the originator is in H. In some agent $i \in H$ could learn the random values of all the agents not in H, before sending (i.e., committing on) a random value to an agent not in H, then i could choose N_i so that this was the case. An easy induction shows that at the end of the rth phase after the originator has received the list of ids, agent $i \in H$ knows that values of at most the r agents in H that precede i on the ring. Thus, at the end of the kth phase, i can learn the random choices of at most the k agents not in H that precede i on the ring, as well as all the choices of the other agents in H. (Learning these choices does not require out-of-band communication, which we do not allow; the agents in H could pre-arrange their choices.) Thus, i knows at most $2k - 1$ random choices. But this is not enough for him to prevent the outcome from being random. Also note that by the end of the kth phase, for each $i \in H$, there will be a $j \notin H$ that knows i's random choice, since it must be sent to the k agents following i on the ring, and one of these cannot be in H. Agent i must send some value (otherwise there will be no leader chosen), and once an agent $j \notin H$ learns the value sent by i, there is no advantage to i in deviating and sending a different value. The key point here is that each agent $i \in H$ must commit to a value without knowing enough other values to be able to influence the final value of N. Given that i commits to a value, i does not gain by committing to a value other than a random value. ∎

We can also use this approach to get a fair Nash equilibrium in a bidirectional network. If agents know the network orientation, they send all their messages in only one direction, implementing the protocol in the unidirectional case. If they do not know the orientation, they first proceed as in the synchronous, exchanging ids to determine who has the highest id. That agent then chooses a direction for further messages, and again they can proceed as in the unidirectional case. Call the resulting protocol profile $\mathbf{A\text{-}LEAD}^{bi}$.

The situation is summarized in the theorem below.

Theorem 3.4 *If $n > 2k$ then $\mathbf{A\text{-}LEAD}^{bi}$ is a fair, ex post k-resilient equilibrium in an asynchronous bidirectional ring.*

Proof: The argument is almost identical to that of the unidirectional case, so we omit it here. ∎

3.5 Asynchronous completely connected network

We can use the ideas above to get a protocol for a completely connected network, embedding a unidirectional ring into the network, but now the added connectivity hurts us, rather than helping. When we embed a ring into the network, each coalition member may be able to find out about up to k other random choices. Since now coalition members can talk to each other no matter where they are on the ring, we must have $n > k(k + 1)$ to ensure that a coalition does not learn all the random choices before the last member

announces his random choice. We can do better by using ideas from secure multi-party computation and secret sharing [Ben-Or, Goldwasser, and Wigderson 1988].

To do secret sharing, we must work in a finite field; so, for ease of exposition, assume that n is a power of a prime. As in the synchronous case, agents start by sending their ids to all other agents, and then exchanging the set of ids received, so that they all agree on the set of ids in the system. (Of course, if an agent i does not get the same set of ids from all agents, then i sets $leader_i = \perp$.) We denote by agent i the agent with the ith largest id. Each agent i chooses a random value $N_i \in \{0, \ldots, n-1\}$ and a random degree-$(k+1)$ polynomial f_i over the field $F_n = \{0, \ldots, n-1\}$ such that $f_i(0) = N_i$. Then i sends each agent j the message $f_i(j)$. Once i receives $f_j(i)$ from all agents j, then i sends *DONE* to all agents. Once i receives *DONE* messages from all agents, i sends $s_i = \sum_{j=1}^{n} f_j(i)$ to all agents. After receiving these messages, i will have n points on the degree-$(k+1)$ polynomial $\sum_{j=1}^{n} f_j$ (if no agents have lied about their values). After i has received the messages s_j for all agents j, i checks if there is a unique polynomial f of degree $k+1$ such that $f(j) = s_j$ for $j = 1, \ldots, n$. If such a polynomial f exists, and $f(0) = N$, then i takes the agent with the Nth highest id as leader; otherwise, i sets $leader_i$ to $\perp$. Call this protocol $A\text{-}LEAD_i^{cc}$.

We summarize the discussion above with the following theorem.

Theorem 3.5 *If $n > 2k$ then $\mathbf{A\text{-}LEAD}^{cc}$ is a fair, ex post k-resilient equilibrium in an asynchronous completely connected network.*

Proof: It is easy to see that, if each agent follows the protocol, then each agent is equally likely to become the leader. To show that the protocol is k-resilient, suppose that $|H| \leq k$. It is clear that no agent in H gains by sending different ids to different agents, for then no leader will be elected. Nor does an agent in H gain by sending an id other than his actual id. Similarly, each agent $i \in H$ does not gain by not sending some value to each agent j (although we have yet to show that these values are of the form $f_i(j)$ for some random polynomial f_i). Since we use degree-$(k+1)$ polynomials, the coalition H has no information about $f(j)$ for $j \notin H$ before it sends a *DONE* message, since, for each $j \notin H$, all the agents in H together know only $f_i(j)$ for $i \in H$. Clearly agents in H do not gain by not sending a *DONE* message to a agent $j \notin H$ (since then j will not set $leader_j$, so there will be no leader elected). However, once an agent receives n *DONE* messages, the value of the leader is fixed by the non-coalition members. Hence, when the coalition learns about some value s_i, it is too late to change their commitments. ∎

3.6 A matching lower bound

We now show that Theorems 3.3 and 3.5 are the best we can hope for; we cannot find a fair ex post k-resilient strategy if $n \leq 2k$.

Theorem 3.6 *If $n \leq 2k$, then there is no fair, ex post k-resilient equilibrium for an asynchronous unidirectional ring (resp., bidirectional ring, completely connected network).*

Observe that all the protocols above are *bounded*; although they involve randomization, there are only boundedly many rounds of communication. This is also the case for the protocol presented in the next section. If we restrict to bounded protocols, using ideas of [Boppana and Narayanan 2000; Saks 1989], we can get a stronger result: we cannot even achieve an ϵ–k-resilient equilibrium (where agents do not deviate if they can get within ϵ of the utility they can get by deviating) for sufficiently small ϵ. [5]

Theorem 3.7 *If $n \leq 2k$, then there exists an $\epsilon > 0$ such that for all ϵ' with $0 < \epsilon' < \epsilon$, there is no fair, ex post ϵ'–k resilient equilibrium for an asynchronous unidirectional ring (resp., bidirectional ring, completely connected network).*

[5] We believe that this result should hold even without the restriction to bounded protocols, but have not proved it yet.

3.7 Doing better with cryptography

In the impossibility result of Section 3.6, we implicitly assumed that the agents were computationally unbounded. For example, even though our proof shows that, in the 2-agent case, one agent can always do better by deviating, it may be difficult for that agent to recognize when it has a history where it could do better by deviating. As we now show, if agents are polynomially-bounded and we make an assumption that is standard in cryptography, then we can get a fair ϵ–k-resilient equilibrium in all these topologies, even in the asynchronous settings, as long as $n > k$. Our solution is based on the bit-commitment protocol of Naor [1991]. Bit commitment ideas can be traced back to the coin-flipping protocol of Blum [1983]. [6]

The key idea of the earlier protocol is that i essentially commits N_i to his neighbor, so that he cannot later change it once he discovers the other agents' random choices. We can achieve essentially the same effect by using ideas from *commitment* protocols [Naor 1991]. In a commitment protocol, an agent Alice commits to a number m in such a way that another agent Bob has no idea what m is. Then at a later stage, Alice can reveal m to Bob. Metaphorically, when Alice commits to m, she is putting it in a tamper-proof envelope; when she reveals it, she unseals the envelope. Put another way, when Alice commits to m, she sends Bob $f(m)$ for some function f that is too hard for Bob to invert; then Alice reveals m by sending Bob information that allows him to compute m from $f(m)$.

It should be clear how commitment can solve the problem above. Each agent i commits to a random number N_i. After every agent has received every other agents' commitment, they all reveal the random numbers to each other. This approach will basically work in our setting, but there are a few subtleties. Naor's commitment protocol requires agents to have access to a *pseudorandom* number generator, and to be polynomially bounded. We can get an ϵ–k-resilient protocol for ϵ as small as we like (provided that Bob is polynomially bounded) by choosing a sufficiently large security parameter for the pseudorandom number generator, but we cannot make it 0. Thus, we actually do not get a fair ex post Nash equilibrium, but a fair ex post ϵ-*Nash* equilibrium. Players can deviate to better strategies (where they actually try to guess the values m being sent during the commitment phase, when playing the role of Bob, or try to find ways of saying that they sent m' when they actually sent m, when playing the role of Alice), but these strategies are only ϵ better. It follows from our impossibility result that we cannot avoid having an ϵ here. That is, there is no fair k-resilient equilibrium even for resource-bounded players. If there is some history at which deviation is profitable for some coalition of players, then the coalition can just encode the deviation into their strategies, without computing it.

In Appendix B, we show how the protocol in the synchronous setting for the unidirectional ring can be modified by using Naor's commitment scheme to get a protocol $A\text{-}LEAD_i^{ps,uni}$ that works in the asynchronous setting. There is another subtlety here. It is not enough for the commitment scheme to be secure; it must also be *non-malleable* [Dolev, Dwork, and Naor 2000]. Intuitively, this means that each choice made by each agent j must be independent of the choices made by all other agents. To understand the issue, suppose that the agent i just before the originator on the ring knows every other agent j's random choice N_j before committing to his own random choice; metaphorically, i has an envelope containing N_j for each agent $j \neq i$. (This is actually the case in our protocol.) Even if i cannot compute N_j, if he could choose N_i in such a way that $\sum_{i=1}^{n} N_i \pmod n$ is 3, he could then choose his id to be 3. If the scheme were malleable, it would be possible for j's choice to depend on the other agents' choices even if j did not know the other agents' choices. Indeed, we want not just non-malleability, but *concurrent* non-malleability. In the protocol, agents engage in a number of concurrent commitment protocols; we do not want information from one commitment protocol to be used in another one. We assume for ease of exposition that Naor's scheme is *concurrently pseudo-non-malleable*; not only can no agent guess other agents' bit with probabil-

[6] Blum provides a coin-flipping protocol that satisfies a relatively weak notion of fairness, which suffices for getting a Nash equilibrium for leader election in the case of $n = 2$. Protocols that satisfy stronger notions of fairness have been studied [Moran, Naor, and Segev 2009], but these stronger notions do not seem necessary for leader election.

ity significantly greater than $1/2$, they also cannot make a choice dependent on other agents' choices with probability significantly greater than $1/2$, even running many instances of the protocol concurrently. (Note that concurrent non-malleable commitment schemes are known; see [Lin and Pass 2009] for the current state of the art.)

We now have the following result.

Theorem 3.8 *For all ϵ, if agents are polynomially bounded and pseudorandom number generators exists, then $A\text{-}LEAD^{ps,uni}$ (with appropriately chosen security parameters) is a fair, ϵ–k-resilient ex post equilibrium in an asynchronous unidirectional ring, for all $k < n$.*

Proof: We leave it to the reader to check that if all agents use the protocol, then each agent is equally likely to become the leader. The proof that no group of size k gains by deviating is similar in spirit to all our earlier proofs. Again, the agents do not gain by not sending their own id initially. Nor do they gain by not sending a commitment vector. If agent i's commitment vector is not computed correctly (i.e., if it is not the case that for some bit vector $(b_{i,1}, \ldots, b_{i,l})$, some seed s, and some bit vector $(r_{i,1}, \ldots, r_{i,3hl})$ (not necessarily chosen at random), each entry $d_{i,k}$ is computed as described above), then this will be discovered in the reveal phase, and there will be no leader elected. Thus, all players will send commitment vectors. Note that it is possible that some member i of a deviating coalition H might learn all the commitment vectors before sending his own vector. Of course, he can also be assumed to know the seeds used by the other members of H (if they computed their commitments vectors using a seed). If i could correctly guess the random seeds of the players not in H, or if he could choose his commitment vector so that the sum of the N_i was some chosen value (such as 3), then i could clearly gain. But, by assumption, the probability that i can guess another player's seed is very low, and the scheme is non-malleable, so that the coalition can gain at most an ϵ-advantage by choosing their commitment vectors in a way that depends on the other vectors. Once the commitment is made, it is clearly too late to get an advantage by deviating. ∎

The same result holds in the case of a bidirectional ring and completely connected network; we can simply embed a unidirectional ring into the network, and run $A\text{-}LEAD^{ps,uni}$. Thus, we have the following result:

Theorem 3.9 *For all ϵ, if agent are polynomially bounded and pseudorandom number generators exists, then there is a a fair, ϵ–k-resilient ex post equilibrium in an asynchronous bidirectional ring (resp., completely connected network) for all $k < n$.*

4 Discussion and Open Questions

We have shown the existence of fair k-resilient equilibria for leader election in several topologies in both the synchronous and asynchronous case. In the asynchronous case, our equilibria require that $n > 2k$ in a unidirectional ring, bidirectional ring, or completely connected network, a requirement that we show is necessary. We show how to get a fair ϵ–k-resilient equilibria in the asynchronous case if $n > k$ in all these topologies under some standard cryptographic assumptions. Our discussion has revealed how cryptographic techniques can be used to help achieve equilibrium. In all these cases, we have obtained *ex post* fair k-resilient equilibria. That means that agents cannot gain by deviating even if they know the choices made by "nature".

The paper illustrates some issues that might arise when trying to apply game-theoretic approaches to distributed computing problems. We have certainly not tried to see what happens in all variants of leader election, nor have we tried to get protocols that are optimal in terms of time, space, or message complexity.

Perhaps what comes out most clearly in the case study is the role of *ex post* Nash equilibrium, both in the upper bounds and lower bounds. To us, the most important question is to consider, when applying game-theoretic ideas to distributed computing, whether this is the most appropriate solution concept. While it is the one perhaps closest to standard assumptions made in the distributed computing literature, it is a very strong requirement, since it essentially means that players have no incentive to deviate even if they know nature's protocol. Are there reasonable distributions we can place on adversary strategies? Do we have to consider them all?

Besides this more conceptual question, there are a number of interesting technical open problems that remain. We list a few here:

- We have focused on the case that agents are rational. In [Abraham, Dolev, Gonen, and Halpern 2006; Abraham, Dolev, and Halpern 2008], we also considered agents who were faulty. Our protocols break down in the presence of even one faulty agent. It is well known that Byzantine agreement is not achievable in a graph of connectivity $\leq 2f$, where f is the number of failures. This suggests that we will not be able to deal with one faulty agent in a unidirectional or bidirectional ring. But it may be possible to handle some faulty agents in a completely connected network.
- We have focused on leader election. It would be interesting to consider a game-theoretic version of other canonical distributed computing problems. We believe that the techniques that we have developed here should apply broadly, since many problems can be reduced to leader election. For example, we could consider consensus, where agents have preferences regarding what decision is reached, but would prefer to reach consensus to not reaching consensus. One way to reach consensus is to elect a leader, and then have the leader declare the consensus value. Can we do better? We suspect not, since if we can do consensus, we can solve leader election by obtaining consensus on who the leader should be. (Of course, it must be checked that these reductions apply n the presence of utilities.) This observation shows just how fundamental leader election is.
- In [Abraham, Dolev, and Halpern 2008], it is shown that, in general, if we can attain an equilibrium with a mediator, then we can attain the same equilibrium using cheap talk only if $n > 3k$. Here we can use cheap talk to do leader election in the completely connected asynchronous case (which is implicitly what was assumed in [Abraham, Dolev, and Halpern 2008]) as long as $n > k$. Thus, we beat the lower bound of [Abraham, Dolev, and Halpern 2008]. There is no contradiction here. The lower bound of [Abraham, Dolev, and Halpern 2008] shows only that there exists a game for which there is an equilibrium with a mediator that cannot be implemented using cheap talk if $n \leq 3k$. It would be interesting to understand what it is about leader election that makes it easier to implement. More generally, can we refine the results of [Abraham, Dolev, Gonen, and Halpern 2006; Abraham, Dolev, and Halpern 2008] to get tighter bounds on different classes of problems?
- We have focused on "one-shot" leader election here. If we consider a situation where leader election is done repeatedly, an agent may be willing to disrupt an election repeatedly until he becomes leader. It would be of interest to consider appropriate protocols in a repeated setting.
- We made one important technical assumption to get these results in rings: we assumed that the ring size is known. As we argued earlier, this assumption is critical, since otherwise an agent can create sybils and increase his chances of becoming leader. However, this deviation comes at a cost. The agent must keep simulating the sybils for all future interactions. This may not be worth it. Moreover, ids must also be created for these sybils. If the name space is not large, there may be an id clash with the id of some other agent in the ring. This will cause problems in the protocols, so if the probability of a name clash is sufficiently high, then sybils will not be created. It would be interesting to do a more formal game-theoretic analysis of the role of sybils.

Appendix

A Sequential equilibrium

Our Nash equilibria requires an agent i to set $leader_i = \perp$ whenever i detects a problem. While this punishes the agent that causes the problem, it also punishes i. Would a rational agent actually would play such a punishment strategy? Note agents punish only off the equilibrium path—it does not occur if all agents follow their part of the Nash equilibrium. Game theorists have developed solution concepts that take behavior off the equilibrium path into account, particularly incredible threats. Perhaps the most popular such solution concept is *sequential equilibrium* [Kreps and Wilson 1982], a refinement of Nash equilibrium, which, roughly speaking, requires that players also make best responses not only on the equilibrium path, but off the equilibrium path as well. In [Abraham, Dolev, and Halpern 2011], we extend the notion of sequential equilibrium to k-*sequential equilibrium*, which allows deviations by coalitions of size up to k. In this section, we show that, under minimal assumptions, our Nash equilibria are actually also k-sequential equilibria.

To make this precise, we first review the relevant definitions from [Abraham, Dolev, and Halpern 2011; Kreps and Wilson 1982]. Recall that, formally, an *extensive-form* game, where players make a sequence of moves, is characterized by a game tree Γ. While we omit the formal definition of a game tree here (see, for example, [Osborne and Rubinstein 1994] for details), it is worth recalling the notion of an *information set*. As is standard, we assume that, associated with each history h in the game tree (where a *history* is a path or prefix of a path in the game tree, and is defined by a sequence of actions), either a unique agent moves at history h, or nature moves. (Nature's moves are taken to be made according to some commonly-known probability distribution.) We assume that, for each agent i, there is a partition $\mathcal{I}_i$ of the histories in the game tree where some agent moves.[7] The cells of $\mathcal{I}_i$ are called i-*information sets*. Let $\mathcal{I}_i(h)$ denote the cell of $\mathcal{I}_i$ containing history h. Intuitively, when agent i is at some history in $\mathcal{I}_i(h)$, then i considers it possible that he could be at another history in $\mathcal{I}_i(h)$. A strategy for player i is a function from the information sets where i moves to an action for player i; thus, i must take the same action at all the histories in an information set.

The main problem in defining sequential equilibrium lies in making sense of what it means for an agent to make a best response at an information set off the equilibrium path, given that such an information set is supposed to be reached with probability 0. To do this, economists have used the notion of a *belief system*, that is, a function that determines for every information set I a probability μ_I over the histories in I. Intuitively, if I is an information set for player i, μ_I is i's subjective assessment of the relative likelihood of the histories in I.

We need to extend belief systems to deal with coalitions. To do this, for $K \subseteq N$, we need to define K-*information* sets. Let $\mathcal{I}_K$ be the refinement partition induced by $\mathcal{I}_i$ for $i \in K$. Thus, if $\mathcal{I}_i(h)$ is the i-information set in $\mathcal{I}_i$ containing h, then $\mathcal{I}_K(h) = \cap_{i \in K} \mathcal{I}_i(h)$.

Definition 1 *A* k-belief system *associates with every* K*-information set* I_K *for* $|K| \leq k$ *a distribution* μ_{I_K} *on the histories in* I_K.

A k-belief system is consistent with a strategy profile σ if, roughly speaking, the beliefs of the players in K at an information set I_K are the conditional beliefs induced by σ. This intuition can be formalized

[7] It is more standard to take $\mathcal{I}_i$ to be a partition of the histories where i moves, not a partition of the histories where some player moves. However, for our later discussion, it is important to define information sets as we have done. We assume that players know whether or not they move, so either i moves at all the histories in a cell of a partition, or at none of them. This means that what we are doing can be viewed as a straightforward generalization of the standard approach.

in a straightforward way on the equilibrium; off the equilibrium path we extend the notion of sequential equilibrium to coalitions as follows.

Given a strategy profile σ for a game Γ, let $\Pr_\sigma$ be the probability distribution induced by σ over the possible histories of Γ. $\Pr_\sigma(h)$ is just the product of the probability of each of the moves in h. For a history h, define $\Pr_\sigma(\cdot \mid h)$ to be the distribution induced by σ over the extensions of h.

Definition 2 *A k-belief system μ is* consistent *with a strategy profile σ if there exists a sequence $\sigma^1, \sigma^2, \ldots$ of* completely mixed *strategy profiles (i.e., $\sigma_i^m(I_i)$ assigns positive probability to all actions that can be played at I_i for all m, all $i \in N$, and all $I_i \in \mathcal{I}_i$ where i moves) converging to σ such that*

- *if $\Pr_\sigma(I_K) > 0$, then $\mu_{I_K}(h) = \Pr_\sigma(h \mid I_K)$; thus, beliefs are obtained by conditioning whenever this makes sense.*
- *If $\Pr_\sigma(I_K) = 0$, then $\mu_{I_K}(h) = \lim_{k\to\infty} \Pr_{\sigma^m}(h \mid I_K)$; thus, the beliefs at I_K are the limit of the beliefs induced by σ^m using conditioning, as $m \to \infty$.*

Definition 3 *σ is a k-*sequential equilibrium *in an extensive-form Γ if there exists a k-belief system μ consistent with σ such that, for all strategy profiles τ in the game $\Gamma + CT(K)$, all K such that $|K| \leq k$, and all information sets $I_K \in \mathcal{I}_K$, we have*

$$\mathrm{EU}_K((\Gamma, \sigma, \mu) \mid I_K) \geq \mathrm{EU}_K((\Gamma + CT(K), \tau_K, \sigma_{-K}, \mu) \mid I),$$

where $\mathrm{EU}_K((\Gamma, \sigma, \mu) \mid I) = \sum_{h \in I} \sum_{z \in Z} \mu_I(h) \Pr_\sigma(z \mid h) u_i(z)$.

Very roughly speaking, this says that σ is a k-sequential equilibrium if, for each player i, σ_i is a best response even off the equilibrium path, according to some beliefs that the players could have about how that information set off the equilibrium path could have been reached. In the special case that $k = 1$, a 1-sequential equilibrium is just a sequential equilibrium as defined by Kreps and Wilson [1982].

We now show that the strategies that we have defined are k-sequential equilibria, if we make one additional assumption. Up to now, we have not made any assumptions about the agents' utility of outcomes where there is no leader. For this section, we assume that, among outcomes where there is no leader, player i's utility of outcomes where $leader_i = \bot$ is at least as high as the utility of any other outcome (so that, in particular, all outcomes where $leader_i = \bot$ are equally good from i's point of view). This would make sense if, for example, i considers it at least as important to realize that there is no leader, if in fact there is no leader, then to mistakenly act as if some player j is the leader. While this seems reasonable, note that it means that i is indifferent as to how much computation is expended. For example, i would have to give equal utility an outcome where $leader_i = \bot$ and i sends many messages to one where $leader_i = \bot$ and i sends no message. i performs an action that involves setting $leader_i$ to $\bot$ (there are many such actions, depending on what messages are sent; we can assume that each such action is performed with equal probability); and with probability $1/m^2$, i performs an action that does not involve setting $leader_i$ to $\bot$. If I is disjoint from the equilibrium path, then i knows that there has been a deviation. In this case, if i has not yet set $leader_i$, then i performs an action that involves setting $leader_i$ to $\bot$ with probability $1 - 1/m$ (again, each such action is performed with equal probability); the remaining probability $1/m$ is uniformly distributed over all actions that do not involve setting $leader_i$ to $\bot$. Finally, if I is disjoint from the equilibrium path and i has already set $leader_i$, the i chooses one of the possible available actions with uniform probability. (If we assume that i can only set $leader_i$ once, then the available actions do not include actions that set $leader_i$.)

It is easy to see that this sequence of strategies generates k-beliefs with the properties we need. Specifically, if a rational player i is an information set that i considers consistent with equilibrium play (which

means that i has been playing σ_i, and he considers it possible that the other agents played σ_{-i}), then i will believe that (with probability 1) the other agents were in fact playing σ_{-i}, and his best response is σ_i. On the other hand, if i is is an information set where he first detects a deviation from equilibrium play, then i will assign probability 1 to some other agent j having set $leader_j$ to $\perp$ (since this is overwhelmingly likely at the first deviation from the equilibrium), so setting $leader_i$ to $\perp$ is a best response for i.

B The protocol A-$LEAD_i^{ps,uni}$

In this section, we present the details of the protocol A-$LEAD_i^{ps,uni}$, which uses Naor's commitment protocol to do leader election in an asynchronous unidirectional ring.

We follow Naor's [1991] presentation of the commitment protocol closely. We consider a fixed pseudorandom generator $G \in \cup_{h=1}^{\infty}(\{0,1\}^h \to \{0,1\}^{m(h)})$; that is, G maps an input seed s of length h to an output of length $m(h) > h$. (G can have inputs of arbitrary length.) We use $B_i(s)$ to denote the ith bit of the pseudorandom sequence $G(s)$. The fact that G is pseudorandom means that no probabilistic polynomial time machine can distinguish between the outputs of G and truly random sequences. Formally, for all polynomials p and all probabilistic polynomial-time machines A, except for finitely many h's, we have

$$|(|(\{y \in \{0,1\}^{m(h)} : A(y) = 1\}|/2^{m(h)}) - (|\{s \in \{0,1\}^h : G(s) = 1]\}|/2^h)| < 1/p(h) .$$

Thus, the cryptographic assumption we need is that such a pseudorandom number generator exists. Naor [1991] discusses how this assumptions relates other cryptographic assumptions.

We want a protocol that is an ϵ-Nash equilibrium, so we choose h to be such that an agent's expected gain by trying to guess a seed s is at most ϵ. If an agent i correctly guesses a seed, the most that he can gain is the difference between his utility if he (i) is the leader and if some other agent j is the leader. Suppose that the maximum such difference is M. Let $\delta(l)$ be the probability of correctly guessing a seed if seeds have length l. Then it suffices to choose l such that $\delta(l)M < \epsilon$.[8]

Before explaining our protocol, we briefly sketch Naor's protocol. Naor initially considers the problem of committing a single bit b in the 2-agent case. So suppose that Alice wants to commit a bit b to Bob. Given a security parameter l, Bob selects a random bit vector $(r_1, \ldots, r_{3l})$ of length $3l$ and sends it to Alice. Alice selects a seed $s \in \{0,1\}^l$ and sends a *commitment vector* $(d_1, \ldots, d_{3l})$ to Bob, where $d_i = B_i(s)$ if $r_i = 0$ and $d_i = B_i(s) \oplus b$ if $r_i = 1$. When Alice is ready to reveal b to Bob, she just sends him s. Bob can confirm that there is a b such that $d_i = B_i(s) \oplus b$ for all i such that $r_i = 1$ and $d_i = B_i(s)$ for all i such that $r_i = 0$. This b is then taken to be Alice's bit.

As Naor [1991] shows, if a polynomial-time bounded agent Bob could guess b with probability nontrivially greater than $1/2$ just from the information he gets during the commit stage, then he would be able to break the pseudorandom number generator. Moreover, the probability that Alice can cheat by finding a pair of seeds s_1 and s_2 such that $B_i(s_1) = B_i(s_2)$ for all i such that $r_i = 0$ and $B_i(s_1) \neq B_i(s_2)$ for all i such that $r_i = 1$ (so that Alice can convince Bob that the bit is either 0 or 1, depending on whether she claims that s_1 is the seed or s_2 is the seed) is at most $1/2^n$.

In our protocol, each agent i needs to commit to a number N_i between 0 and $n-1$, where n is the number of agents. This amounts to committing to $h = \lceil \log(n) \rceil$ bits. For ease of exposition, we take a relatively inefficient approach to doing this, by essentially running h instances of Naor's protocol in parallel. Naor [1991, Section 4] shows how we can improve the bit complexity.

[8] An agent must in fact guess the seeds of $n-1$ agents in order to get an advantage, so it would suffice to choose l such that $\delta(l)^{n-1}M < \epsilon$.

The protocol starts just as in the synchronous case. Each agent who gets a signal to start leader election sends his id to his neighbor. Messages are passed around the ring, with each agent appending his id before passing it on. If an agent receives a second message that originated with a different agent, the message is ignored if the originating agent has a lower id; otherwise it is passed on. Eventually the originator of the message with the highest id gets back the message. He then sends the message around the ring a second time. When the message makes it back to the originator the second time, everyone knows the ids of all agents on the ring. (As above, the originator needs to ensure that there are exactly n (different) ids on the message before passing it around.)

Suppose that each agent j has a bit vector $(b_{j,1}, \ldots, b_{j,l})$ to which he wants to commit. (This vector represents the randomly chosen N_j.) Agent j chooses a random bit vector $(r_{j,1}, \ldots, r_{j,3hl})$ and a random seed s_j. Thus, each agent will play the role of both Alice and Bob. Note that the random bit vector is h times as long as the vector in Naor's protocol, since we are essentially running h copies of the protocol. The originator sends his random vector to his left neighbor. The neighbor add his random vector to the list, and so on. The list goes around the ring twice (just like the list of ids). When the originator gets back the random vector list the second time, everyone knows everyone else's random vector. Next, each agent j computes the vector $(d_{j,1}, ..., d_{j,3hl})$ essentially as in Naor's paper, using the random sequence of his right neighbor. (That is, each agent plays the role of Alice, taking Bob to be his right neighbor.) In more detail, if the right-hand neighbor of j is j', and uses random vector $(r_{j',1}, \ldots, _{j',3hl})$, then $d_{j,l} = B_l(s_j)$ if $r_{j',l} = 0$, and $d_{j,l} = B_l(s_j) + b_{j,h'}$ if $r_{j',1} = 1$, where $3(h'-1)l < l \leq 3hl$. (That is, agent j uses $b_{j,1}$ in computing $d_{j,1}, \ldots, d_{j,3l}$, $b_{j,2}$ for computing $d_{j,3l+1,\ldots,d_j,6l}$, and so on.) Call the sequence $(d_{j,1}, \ldots, d_{j,3hl})$ j's *commitment vector*. Now the originator i starts another list going around the ring by sending his commitment vector $(d_{i,1}, \ldots, d_{i,3hl})$ to his left neighbor; the neighbor then adds his commitment vector, and so on. Again, this list goes around the ring twice. At this point, everyone knows everyone else's commitment vector.

After the list goes around twice, the originator starts the reveal phase, by sending his seed s_i around the ring. Each agent adds his seed, and, as usual, the list of seeds goes around twice. After receiving the list of seeds, each agent can compute the random choices N_j for each other agent, and elect a leader as usual. Of course, if an agent i does not receive the full list of seeds, or if some seed cannot be used by i to compute the random choices, then i sets $leader_i$ to $\perp$. Call this protocol $A\text{-}LEAD_i^{ps,uni}$.

C Proofs

Theorem 3.1: $\mathbf{LEAD}^{cc}$ *is a fair, k-resilient equilibrium in a completely connected network of n agents, for all $k < n$.*

Proof: First suppose that $\mathbf{LEAD}^{cc}$ is used. Then clearly each agent is equally likely to become the leader. At the risk of belaboring the argument, it is worth bringing out the key ideas, since they are common to all our later proofs. First, what is necessary is a common naming system, so if the agents do decide on agent i as the leader, they all agree on who i is. The first two steps do that. When all the agents exchange their ids (provided that each agent sends the same id to all the other agents), they end up agreeing on what the names are and who has each name. Next, the agents need a common way of ordering themselves, so that when they choose a number j (mod n) at random, the agents agree on who the jth agent is. Once we have all the ids, we can order the ids. Finally, the agents need a way of choosing a common number j at random. The last steps do that.

It now suffices to show that no group of $k < n$ agents has any incentive to deviate. Suppose that a group H of agents with $|H| < n$ agents all deviate, while no other agents deviate. If any of the agents in H send different ids to an agent not in H at step 1, then each agent j not in H will receive different sets in step 2,

and thus set $leader_j = \perp$, making the agents in H no better off than if they do not deviate. (We still need to argue that agents in H send their actual ids; see below.) If an agent in H deviates at step 2 by sending different random numbers to agents not in H, then the agents not in H will disagree about the leader, again making i no better off. If an agent $i \in H$ deviates by not taking the agent with the Nth highest id to be the leader, then again agents in H will disagree with the agents not in H about the leader, making i no better off. It is clear that as long as all agents not in H follow the protocol, and each $i \in H$ sends each agent not in H the same id in step 1 and the same number (not necessarily random) in step 2, then each agent is equally likely to be the leader. Thus, an agent $i \in H$ does not gain by deviating in step 1 by sending an id other than his actual id, nor does he gain by choosing a number in step 2 other than a random number. We have thus shown that $\mathbf{LEAD}^{cc}$ is indeed a fair, k-resilient equilibrium. ∎

Theorem 3.2: $\mathbf{LEAD}^{uni}$ *is a fair, k-resilient (ex post) equilibrium in a unidirectional ring with n agents, for all $k < n$.*

Proof: Again, it should be clear that if $\mathbf{LEAD}^{uni}$ is used, then each agent is equally likely to become leader. Thus, it suffices to show that no group H of agents with $|H| < n$ has any incentive to deviate. If an agent $i \in H$ deviates by not sending a signed message to his neighbor when he gets a signal to start leader election, then either it has no impact (if other agents also get a signal) or no leader is elected when one could have been (so i is not better off). If i does not forward some message that i should forward or delays in forwarding a message, again, either it has no impact (if an agent with a lower id also gets a signal) or no leader is elected when one could have been (so i is not better off). If i modifies a message that it should forward, it may be the case that different agents not in H end up with different sets of ids, or different sets of n numbers. In the latter case, if the sum of the numbers in the set each agent considers possible is the same, then the change has no impact. Otherwise, different agents will have different views of who the leader should be—that is, there is no one leader elected. In either case, the deviator does not gain. Similarly, if the different agents have view the set of ids as being different, either it has no impact, or else different agents will have different views of who is the leader. Moreover, if an agent $i \in H$ does not send a number $2n$ rounds after the "winning" originator sent his message to an agent not in H, then no leader is elected, and i is not better off. Thus, each agent $i \in H$ will send some id if he gets a signal at the beginning, and will send some number N_i at the appropriate time. Agent i does not gain by deviating in step 1 by sending an id other than his actual id, nor does he gain by choosing a number N_i other than a random number. We have thus shown that $\mathbf{LEAD}^{uni}$ is a fair, k-resilient equilibrium. ∎

Theorem 3.3: *If $n > 2k$, then $\mathbf{A}$-$\mathbf{LEAD}^{uni}$ is a fair, k-resilient ex post equilibrium in an asynchronous unidirectional ring.*

Proof: As usual, it is straightforward to confirm that if $\mathbf{A}$-$\mathbf{LEAD}^{uni}$ is used, then each agent is equally likely to become leader, so again it suffices to show that if $n > 2k$, then no group H of agents with $|H| \leq k$ has any incentive to deviate, no matter what the adversary does. Again, we want to argue that there is no advantage for an agent $i \in H$ to deviate up to and including the point where he sends his random number N_i, because deviation either has no impact, or results in no leader being chosen. But after sending the random number, it is too late to bias the election of the leader.

As in the proof of Theorem 3.2, if an agent $i \in H$ deviates by not sending a signed message to his neighbor when he gets a signal to start leader election, then either it has no impact (if other agents also get a signal) or no leader is elected when one could have been (so i is not better off). If i does not forward some message that i should forward during the first two phases (when just the list of ids is going around the ring) then either it has no impact or no leader is elected when one could have been (so i is not better off). If i modifies a message that it should forward, it may be the case that different agents not in H end up with different sets of ids. As long as all agents agree on which id is the lowest, this will not make a difference. If

they disagree on this, then we will not have a single leader. As before, this just shows that all the agents will send some id, not necessarily their own. But they clearly gain no benefit by not sending their own id.

Similarly, all the agents in H send messages in each of the k phases after the originator has received the list of ids for the second time, otherwise there will not be a leader elected. They must also send the messages in a consistent way, so that they pass all the checks. The agents in H would like to have the sum $N = \sum_{i \in 1}^{n} N_i$ be such that the Nth highest id is in H. If some agent $i \in H$ could learn the random values of all the agents not in H before sending (i.e., committing on) a random value to an agent not in H, then i could choose N_i so that this was the case. An easy induction shows that at the end of the rth phase after the originator has received the list of ids, agent $i \in H$ knows that values of at most the r agents in H that precede i on the ring. Thus, at the end of the kth phase, i can learn the random choices of at most the k agents not in H that precede i on the ring, as well as all the choices of the other agents in H. (Learning these choices does not require out-of-band communication, which we do not allow; the agents in H could pre-arrange their choices.) Thus, i knows at most $2k - 1$ random choices. But this is not enough for him to prevent the outcome from being random. Also note that by the end of the kth phase, for each $i \in H$, there will be a $j \notin H$ that knows i's random choice, since it must be sent to the k agents following i on the ring, and one of these cannot be in H. Agent i must send some value (otherwise there will be no leader chosen), and once an agent $j \notin H$ learns the value sent by i, there is no advantage to i in deviating and sending a different value. The key point here is that each agent $i \in H$ must commit to a value without knowing enough other values to be able to influence the final value of N. Given that i commits to a value, i does not gain by committing to a value other than a random value. ∎

Theorem 3.5: *If $n > 2k$, then $\mathbf{A\text{-}LEAD}^{cc}$ is a fair, ex post k-resilient equilibrium in an asynchronous completely connected network.*

Proof: It is easy to see that, if $\mathbf{A\text{-}LEAD}^{cc}$ is used, then each agent is equally likely to become the leader. To show that $\mathbf{A\text{-}LEAD}^{cc}$ is k-resilient, suppose that $|H| \leq k$. It is clear that no agent in H gains by sending different ids to different agents, for then no leader will be elected. Nor does an agent in H gain by sending an id other than his actual id. Similarly, each agent $i \in H$ does not gain by not sending some value to each agent j (although we have yet to show that these values are of the form $f_i(j)$ for some random polynomial f_i). Since we use degree-$(k+1)$ polynomials, the coalition H has no information about $f(j)$ for $j \notin H$ before it sends a *DONE* message, since, for each $j \notin H$, all the agents in H together know only $f_i(j)$ for $i \in H$. Clearly agents in H do not gain by not sending a *DONE* message to an agent $j \notin H$ (since then j will not set $leader_j$, so there will be no leader elected). However, once an agent receives n *DONE* messages, the value of the leader is fixed by the non-coalition members. Hence, when the coalition learns about some value s_i, it is too late to change their commitments. ∎

Theorem 3.6: *If $n \leq 2k$, then there is no fair, ex post k-resilient equilibrium for an asynchronous unidirectional ring (resp., bidirectional ring, completely connected network).*

Proof: We first consider the case that $n = 2$ and $k = 1$. Note that for two agents, a unidirectional ring, a bidirectional ring, and a completely connected network all coincide. It is well known that there is no fair coin tossing protocol for two computationally unbounded agents. Leader election and fair coin tossing were studied intensively [Ben-Or and Linial 1985; Boppana and Narayanan 2000; Saks 1989] and a lower bound of $n = 2$ if known for bounded games with full information in a broadcast environment. A fair ex post Nash equilibrium essentially gives us a fair coin-tossing protocol, so the impossibility of fair coin tossing for two agents essentially implies the impossibility of ex post Nash equilibrium. We provide a careful proof here, bringing out how the requirement of ex post equilibrium plays a critical role.

Call the agents 0 and 1. Suppose, by way of contradiction, that (σ_0, σ_1) is an ex post fair Nash equilibrium. Consider the strategy ρ for nature according to which agent 0 is the only agent to get a signal, agents

are scheduled alternately, and whatever messages they send are delivered immediately. That is, if nature uses strategy ρ, we can think of runs as proceeding in rounds. If agent 0 sends a message at round k at all, it is sent at time $4k+1$; this message is received by agent 1 at time $4k+2$; if agent 1 sends a round k message at all, it is sent at time $4k+3$, and received by agent 0 at time $4k+4 = 4(k+1)$. This means that, before agent i sends a round k message, i has received all the messages that the other agent, agent $1-i$, sent earlier. Since we want to show that (σ_0, σ_1) is not an ex post Nash equilibrium, it suffices to show that it is not a Nash equilibrium conditional on nature following strategy ρ.

Consider a run r that occurs with positive probability conditional on nature using strategy ρ and the agents using (σ_0, σ_1), and suppose that 0 ends up being the leader in run r. Let t be the first time at which one of 0 or 1 knows that 0 will be the leader. For the purposes of this proof, we interpret agent i *knows* an event E at time t as meaning that, conditional on nature following strategy ρ, the agents using (σ_0, σ_1), and i's message history at time t, i assigns E probability 1. *Events* are just sets of runs, so the event "0 is the leader" is the set of runs where 0 is the leader.

Going on with the proof, let h_i be agent i's history at time t. Suppose that agent 1 is the one who knows that 0 is the leader at time t. Thus, agent 1's state must have changed at time t, so 1 either sent or received a message. Given that nature's strategy is ρ, agent 0's state didn't change at time t. It follows that agent 0 does not know that 0 is the leader at time t (otherwise 0 would have known it earlier). Thus, there must be a history h_1' for agent 1 and a run r' such that (a) r' occurs with positive probability conditional on nature following strategy ρ and the agents using (σ_0, σ_1), (b) 1 is chosen leader in r', and (c) (h_0, h_1') are the message histories of 0 and 1 at some time t' in run r. If t has the form $4k+2$ (so agent 1 has just received a message from 0), then, given that nature is following ρ, h_1 must be a prefix of h_1'. But this contradicts the assumption the agent 1 knows when he has history h_1 that 0 will be the leader. So we can assume that $t = t'$ and has the form $4k+3$. It must be the case that 1's histories at time $t-1$ are identical in runs r and r', and that 1 just sent a message at time t in at least one of r and r'. Let h_1'' be 1's history at time $t-1$ in runs r and r'. Since 1 does different things at time t in r and r' using strategy σ_1, although he has the same history at time $t-1$, it must be the case that σ_1 is randomizing. But then 1 is clearly better off deviating when his coin lands as it does in r, and pretending that his coin landed as it did in r', since this increases his probability of being leader.

Now suppose that agent 0 is the one who knows that 0 is the leader at time t. Now it must be the case that 1 does not know at time t that 0 is the leader. So there must be a run r'' such that (a) r'' occurs with positive probability conditional on nature following strategy ρ and the agents using (σ_0, σ_1), (b) 1 is chosen leader in r'', and (c) (h_0', h_1) are the message histories of 0 and 1 at some time t' in run r. As above, it must be the case that t has the form $4k+1$ (so agent 0 has just sent a message), that 0's histories at time $t-1$ in runs r and r'' are identical, and the differences between h_0' and h_0 must be due to agent 0 getting different random outcomes in r and r''. But now 0 is clearly better off deviating when his coin lands as it does in r'', and pretending that it landed as it did in r, so that he is certain to become leader in r''.

In either case, it follows that (σ_0, σ_1) is not a Nash equilibrium conditional on nature using strategy ρ, which means that (σ_0, σ_1) is not an ex post Nash equilibrium. For future reference, note that exactly the same argument as above shows that for any choice of α with $0 < \alpha < 1$, there is no Nash equilibrium conditional on nature using strategy ρ where player 1 is the leader with probability α and player 2 is the leader with probability $1-\alpha$.

We now show how to extend this result to the case that $n > 2$. First consider a unidirectional ring. Suppose that $(\sigma_1, \ldots, \sigma_n)$ is an ex post k-resilient equilibrium for n players. For simplicity, suppose that we name the agents going around the ring $0, \ldots, n-1$. Split the agents into two groups, $A = \{0, \ldots, \lfloor (n-1)/2 \rfloor\}$ and $B = \{\lfloor (n-1)/2 \rfloor + 1, \ldots, n\}$. Note that since $n \leq 2k$, each group has size at most k. Now we consider nature's strategy ρ_n where only player 0 gets a signal, players are scheduled in order around the ring (first 0, then 1, ..., then $n-1$, then 0 again, and so on), and player i's round k message (if there is

one) is received by player $i+1 \pmod n$ before player $i+1 \pmod n$ sends his round k message. (Clearly ρ_n generalizes ρ.) Since $(\sigma_1, \ldots, \sigma_n)$ is an ex post k-resilient equilibrium, it must be a fair k-resilient equilibrium conditional on nature using ρ_n.

We claim that it now follows there must be a Nash equilibrium for 2 players conditional on nature using ρ where player 1 has a probability $\lfloor (n+1)/2 \rfloor / n$ of being the leader, contradicting our earlier argument. Player 1 simply simulates what A does, and 2 simulates what B does. That is, 1 can simulate what the agents in A do when they send message internally, and 2 can simulate what the agents in B do when they send messages internally. When agent $n-1$ in B sends a message to agent $0 \in A$ in the simulation, 2 sends a message to 1 in the simulation; similarly, when agent $\lfloor n/2 \rfloor$ sends a message to agent $\lfloor n/2 \rfloor$, 1 sends a message to 0. Agent 0 declares himself the leader iff some member of A becomes the leader in the simulation; similarly, agent 1 declares himself the leader if some member of B declares himself leader in the simulation. Now suppose that this is not a Nash equilibrium. If 1 has a profitable deviation, then the group B (which has size less than k) has a profitable deviation in the size n ring, since they can simulate 1's deviation; likewise group A can simulate any deviation by 0. It follows that there can be no fair k-resilient protocol for the asynchronous unidirectional ring.

The same argument essentially works for the bidirectional ring and the completely connected network. In both cases, we again split the agents into two groups that are as close to being equal in size as possible (so that they both have size at most k). In a bidirectional ring, we consider a strategy for nature where again agents are scheduled sequentially, but messages from 0 to $n-1$ are delayed so that they arrive at the same time as messages from $\lfloor (n-1)/2 \rfloor$ to $\lfloor (n-1)/2 \rfloor + 1$ (i.e., all messages from an agent in A to an agent in B arrive at the same time; similarly, all message from $\lfloor (n-1)/2 \rfloor + 1$ to $\lfloor (n-1)/2 \rfloor$ are delayed so that they arrive at the same time as message from $n-1$ to 0. In the 2-player setting, when agent 0 sends a message to agent 1, 0 must specify what message in the simulation $0 \in A$ sends to $n-1$ and what message $\lfloor (n-1)/2 \rfloor$ sends to $\lfloor (n-1)/2 + 1 \rfloor$, and similarly when 1 sends a message to 0.

For the completely connected network, we again do the same thing, holding up messages so that all messages from an agent in A to an agent in B arrive at the same time, and similarly in the other direction. In the simulation in the 2-player ring, player 0 has to specify, for each agent in A, what messages he sends and which agents in B they are being sent to, and similarly for messages from 1 to 0. ∎

Theorem 3.7: *If $n \leq 2k$, then there exists an $\epsilon > 0$ such that for all ϵ' with $0 < \epsilon' < \epsilon$, there is no fair, ex post ϵ'–k resilient equilibrium for an asynchronous unidirectional ring (resp., bidirectional ring, completely connected network).*

Proof: Again, we start by considering a 2-player game. Again, call the agents 0 and 1. Let G_i be i's utility if i is the leader, and let B_i be i's utility of $1-i$ is the leader. In a fair equilibrium, i's utility is $(B_i + G_i)/2$. Let $\epsilon = \min((G_0 - B_0)/2, (G_1 - B_1)/2))$. We show that there cannot be a fair ex post ϵ'-Nash equilibrium for $\epsilon' < \epsilon$.

Suppose that there were such an equilibrium. We show that this would contradict well-known impossibility results in the *perfect-information model of distributed computing* considered by Ben-Or and Linial [1985]. In this model, informally, players take turns broadcasting messages. All nth round message are received before any $(n+1)$st round message is sent; no assumptions are made on the relative order of delivery. In general, the Ben-Or–Linial model is incomparable to ours. We do not assume broadcast; players send point-to-point messages, and who they can send to depends on the network topology. Moreover, in the asynchronous setting, we cannot in general assume that messages are sent in rounds. However, in the case of two players, for the particular adversary considered in the proof of Theorem 3.6, the two models are essentially identical.

To prove a contradiction, it suffices to show that there is no ϵ'-fair equilibrium when playing with this adversary. This follows easily from a result claimed by Saks [1989], and proved by Boppana and Narayanan

[2000] that there is no ϵ-fair leader election protocol in the Ben-Or–Linial model if more than half the players are faulty. We now formalize the details.

In Saks' formalization, a distributed protocol in the Ben-Or–Linial model is characterized by a binary tree. Each node in the tree is labeled by a player, and the edges leading out of a non-leaf node are labeled by probabilities (so that the sum of the two labels is 1).[9] We now associate with each player $i \in \{0, 1\}$ and each tree T a probability $P_i(T)$. We can think of $P_i(T)$ as the probability that $1 - i$ will be elected leader if i deviates from the protocol while $1 - i$ follows the protocol. We define $P_i(T)$ by induction on the height of the tree. If T is a single node, then $P_i(T)$ is 0 if T is labeled i, and 1 otherwise. In general, if T is a tree whose left edge has probability p_L and whose left and right subtrees are T_L and T_R respectively, then

$$P_i(T) = \begin{cases} \min(P_i(T_L), P_i(T_R) & \text{if the root of } T \text{ is labeled } i \\ p_L P_i(T_L) + (1 - p_L)P_i(T_R) & \text{if the root of } T \text{ is labeled } 1 - i. \end{cases}$$

Intuitively, if i plays at the root of T, i will deviate from the protocol and choose whichever move is most favorable to him; if $1 - i$ plays at the root of T, then $1 - i$ will follow the protocol.

Saks [1989] claimed, and Boppana and Narayanan [2000] proved, that $\min_i(P_0(T), P_1(T)) = 0$ for all protocols T.[10] Thus, one of the players, say i, can deviate in such a way as to guarantee that he will be leader. It follows that i can gain a utility of $(G_i - B_i)/2 \geq \epsilon > \epsilon'$ by deviating. This gives the desired contradiction.

The extension to the case $n > 2$ proceeds along lines similar to the proof of Theorem 3.6. We omit details here. ∎

Theorem 3.8: *For all ϵ, if agent are polynomially bounded and pseudorandom number generators exists, then A-$LEAD^{ps,uni}$ (with appropriately chosen parameters) is a fair, ϵ–k-resilient ex post equilibrium in an asynchronous unidirectional ring, for all $k < n$.*

Proof: We leave it to the reader to check that if A-$LEAD^{ps,uni}$ is used, then each agent is equally likely to become the leader. The proof that no group of size k gains by deviating is similar in spirit to all our earlier proofs. Again, the agents do not gain by not sending their own id initially. Nor do they gain by not sending a commitment vector. If agent i's commitment vector is not computed correctly (i.e., if it is not the case that for some bit vector $(b_{i,1}, \ldots, b_{i,l})$, some seed s, and some bit vector $(r_{i,1}, \ldots, r_{i,3hl})$ (not necessarily chosen at random), each entry $d_{i,k}$ is computed as described above), then this will be discovered in the reveal phase, and there will be no leader elected. Thus, all players will send commitment vectors. Note that it is possible that some member i of a deviating coalition H might learn all the commitment vectors before sending his own vector. Of course, he can also be assumed to know the seeds used by the other members of H (if they computed their commitments vectors using a seed). If i could correctly guess the random seeds of the players not in H, or if he could choose his commitment vector so that the sum of the N_i was some chosen value (such as 3), then i could clearly gain. But, by assumption, the probability that i can guess another player's seed is very low, and the scheme is non-malleable, so that the coalition can gain at most an ϵ-advantage by choosing their commitment vectors in a way that depends on the other vectors. Once the commitment is made, it is clearly too late to get an advantage by deviating. ∎

D Protocols

[9] The binary tree corresponds to there being only two possible messages. In our protocols, there is a finite set of possible messages, although in general there are more than two. This is not a problem; if there are k possible messages, we can use $\log(k)$ steps in a binary tree to specify which message is being sent.

[10] The Boppana-Narayanan proof assumes that the probability $p_L = 1/2$, but the identical proof works for an arbitrary p_L.

Protocol 1: Protocol $LEAD_i^{cc}$, executed by each agent i:

Outputs $leader_i$ as the leader chosen by agent i and then halts.

1: send id_i to all
   $IDset := \{id_j | j$'s id received in this round$\}$
2: send $IDset_i$ to all agents
   if $\exists j, k$ s.t. $IDset_j \neq IDset_k$ **then** $leader_i := \perp$; halt *// punish*
   else
      $n := |IDset_i|$ *// n is the number of agents*
      draw a random number N_i from $\{0, \ldots . n-1\}$
3: send N_i to all agents
      if N_j not received from some agent j **then** $leader_i := \perp$; halt *// punish*
      else
         $N := \sum_j N_j (\mathrm{mod} n)$
         set $leader_i$ to the Nth largest id in $IDset_i$
         halt *// choose a leader*

Protocol 2: Protocol $LEAD_i^{uni}$, executed by each agent i :

Outputs $leader_i$ as the leader chosen by agent i and then halts.
Gets n (the ring size—presumed to be commonly known) as input.

1: Upon waking: *// either spontaneously or by receiving a message from predecessor*
$count_i := n$ *// when you next expect a message*
$highest_i := id_i$ *// who you expect the message from*
$phase = 1$ *//what kind of message you expect*
send Collect_Msg$\langle id_i, \langle id_i \rangle \rangle$ to your successor in the ring

2: In each round $count := count - 1$

3: Upon receiving Collect_Msg$\langle id, IDset \rangle$ from your predecessor:
if $id > highest_i$ and $phase = 1$ **then**
$IDset := IDset \cdot id_i$ *// id_i is appended to the list IDset*
$count_i := n$
$highest_i := id_j$
send Collect_Msg$\langle id, IDset \rangle$ to your successor
if $id = highest_i$ **then**
if $|IDset| \neq n$ or $phase \neq 1$ or $count \neq 0$ or $id \neq id_i$ **then** $leader_i := \perp$; halt *// punish*
else $phase = 2$
$IDset_i := IDset$
$count := n$
send Setup_Msg$\langle id_i, n, IDset \rangle$ to your successor

4: Upon receiving Setup_Msg$\langle id, k, IDset \rangle$ from your predecessor
if $id = highest_i$ and $phase = 1$ and $count = 0$ and $|IDset| = n$ **then**
$phase := 2$
$IDset_i := IDset$
$count = k - 1$
send Setup_Msg$\langle id, k-1, IDset \rangle$ to your successor
else if $phase = 1$ **then** $leader_i := \perp$; halt *// punish*

5: **if** $phase = 2$ and $count = 0$ **then**
Draw a random number N_i from $\{0, \ldots . n-1\}$
Send Rand_msg$\langle id_i, N_i \rangle$ to your successor
$phase := 3$
$count := n$

6: **if** $phase = 3$ and $count = j > 0$ **then**
if i receives Rand_msg$\langle id, N' \rangle$ from its predecessor and id is the id of the jth agent prior to i in $IDset_i$
then forward Rand_msg$\langle id, N' \rangle$ to your successor
$N_j := N'$
$j = j - 1$
else $leader_i = \perp$ *// punish*

7: **if** $phase = 3$ and $count = 0$ and $leader_i \neq \perp$ **then**
$N := \sum_j N_j (\mathrm{mod} n)$
set $leader_i$ to the Nth largest id in $IDset_i$
halt

Protocol 3: Protocol $\mathbf{A\text{-}LEAD}^{uni}$, executed by each agent i:

Outputs $leader_i$ as the leader chosen by agent i and then halts.
Gets n (the ring size—presumed to be commonly known) as input.

1: Upon waking: *// either spontaneously or by receiving a message from predecessor*
  $highest_i := id_i$ *// who you expect the message from*
  $phase_i := 1$ *//what kind of message you expect*
  send Collect_Msg$\langle id_i, \langle id_i \rangle \rangle$ to your successor in the ring
2: Upon receiving Collect_Msg$\langle id, IDset \rangle$ from your predecessor:
  if $id > highest_i$ and $phase_i = 1$ **then**
    $IDset := IDset \cdot id_i$ *// id_i is appended to the list $IDset$*
    $highest_i := id_j$
    send Collect_Msg$\langle id, IDset >$ to your successor
  if $id = highest_i$ **then**
    if $|IDset| \neq n$ or $phase_i \neq 1$ or $id \neq id_i$ **then** $leader_i := \perp$; halt *// punish*
    else $phase_i := 2$
      $IDset_i := IDset$
      send Setup_Msg$\langle id_i, IDset \rangle$ to your successor
3: Upon receiving Setup_Msg$\langle id, IDset \rangle$ from your predecessor
  if $id = highest_i$ and $phase_i = 1$ and $|IDset| = n$ **then**
    $phase_i := 2$
    $IDset_i := IDset$
    send Setup_Msg$\langle id, IDset \rangle$ to your successor
  else if $phase_i = 1$ **then** $leader_i := \perp$; halt *// punish*
4: **if** $phase_i = 2$ and $id = id_i$ and $IDset = IDset_i$ **then**
    Draw a random number N_i from $\{0, \ldots .n-1\}$
    Send Rand_msg$\langle IDset_i, N_i \rangle$ to your successor
    $phase_i := 3$
5: Upon receiving Rand_msg$\langle IDset, N' \rangle$ from predecessor
  $N_{pre} := N'$
  if $phase = 2$ **then**
    Draw a random number N_i from $\{0, \ldots .n-1\}$
    Send Rand_msg$\langle IDset_i, N_i \rangle$ to your successor
    $phase_i := 3$
6: **else if** $phase_i = 3$ **then**
      Send Collect_list$\langle IDset_i, \langle N_i \rangle \rangle$ to your successor
      $phase_i := 4$
7: Upon receiving Collect_list$\langle IDset, N_{list} \rangle$ from predecessor
  if $phase = 3$ and N_{pre} is the final element in N_{list} **then**
    append N_i to N_{list}
    Send Collect_list$\langle IDset, N_{list} \rangle$ to your successor
    $phase_i := 4$
8: **else if** $phase_i = 4$ and N_{pre} is the final element in N_{list} **then**
      Send Choose_msg$\langle IDset_i, N_{list} \rangle$ to your successor
9: Upon receiving Choose_msg$\langle IDset, N_{list} \rangle$ from predecessor
  if $phase = 4$ and N_{list} agrees with previously seen N_{list} **then**
    Send Choose_msg$\langle IDset, N_{list} \rangle$ to your successor
    $N := \sum_{N' \in N_{list}} N' (\mathrm{mod} n)$
    set $leader_i$ to the Nth largest id in $IDset_i$
    halt.

References

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

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

Abraham, I., D. Dolev, and J. Y. Halpern (2011). On the power of cheap talk. Unpublished manuscript.

Aiyer, A. S., L. Alvisi, A. Clement, M. Dahlin, J. P. Martin, and C. Porth (2005). BAR fault tolerance for cooperative services. In *Proc. 20th ACM Symposium on Operating Systems Principles (SOSP 2005)*, pp. 45–58.

Barany, I. (1992). Fair distribution protocols or how the players replace fortune. *Mathematics of Operations Research 17*, 327–340.

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

Ben-Or, M. and N. Linial (1985). Collective coin flipping, robust voting schemes and minima of Banzhaf values. In *Proc. 26th IEEE Symp. Foundations of Computer Science*, pp. 408–416.

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

Blum, M. (1983). Coin flipping by telephone: a protocol for solving impossible problems. *SIGACT News 15*, 23–27.

Boppana, R. B. and B. O. Narayanan (2000). Perfect-information leader election with optimal resilience. *SIAM Journal on Computing 29*(4), 1304–1320.

Chang, E. and R. Roberts (1979). An improved algorithm for decentralized extrema-finding in circular configurations of processes. *Communications of the ACM 22(5)*, 281–283.

Dani, V., M. Movahedi, Y. Rodriguez, and J. Saia (2011). Scalable rational secret sharing. In *Proc. 30th ACM Symposium on Principles of Distributed Computing*, pp. 187–196.

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

Dolev, D. (1982). The Byzantine generals strike again. *Journal of Algorithms 3*(1), 14–30.

Dolev, D., C. Dwork, and M. Naor (2000). Non-malleable cryptography. *SIAM Journal on Computing 30*(2), 391–437.

Dolev, D., M. Klawe, and M. Rodeh (1982). An $o(n \log n)$ unidirectional distributed algorithm for extrema finding in a circle. *Journal of Algorithms 3*(3), 245–260.

Feldman, P. and S. Micali (1997). An optimal probabilistic protocol for synchronous Byzantine agreement. *SIAM Journal on Computing 26*, 873–933.

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

Fuchsbauer, G., J. Katz, and D. Naccache (2010). Efficient rational secret sharing in standard communication networks. In *Proc. 7th Theory of Cryptography Conference (TCC '10)*, pp. 419–436.

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

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

Heller, Y. (2005). A minority-proof cheap-talk protocol. Unpublished manuscript.

Izmalkov, S., M. Lepinski, and S. Micali (2011). Perfect implementation. *Games and Economic Behavior 71*, 121–140.

Katz, J. and C.-Y. Koo (2006). On expected constant-round protocols for Byzantine agreement. In *CRYPTO 2006: 26th International Cryptology Conference*, pp. 445–462.

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

Le Lann, G. (1977). Distributed systems—towards a formal approach. In *IFIP Congress*, Volume 7, pp. 155–160.

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

Lin, H. and R. Pass (2009). Constant-round non-malleable commitments from any one-way function. In *Proc. Twenty-First International Joint Conference on Artificial Intelligence (IJCAI '09)*, pp. 153–158.

Lynch, N. A. (1997). *Distributed Algorithms*. San Francisco: Morgan Kaufmann.

Lysyanskaya, A. and N. Triandopoulos (2006). Rationality and adversarial behavior in multi-party comptuation. In *CRYPTO 2006: 26th International Crypology Conference*, pp. 180–197.

McGrew, R., R. Porter, and Y. Shoham (2003). Towards a general theory of non-cooperative computing. In *Theoretical Aspects of Rationality and Knowledge: Proc. Ninth Conference (TARK 2003)*, pp. 59–51.

Moran, T., M. Naor, and G. Segev (2009). An optimally fair coin toss. In *Proc. 6th Theory of Cryptography Conference (TCC '09)*, pp. 1–18.

Moscibroda, T., S. Schmid, and R. Wattenhofer (2006). When selfish meets evil: Byzantine players in a virus inoculation game. In *Proc. 25th ACM Symposium on Principles of Distributed Computing*, pp. 35–44.

Naor, M. (1991). Bit commitment using pseudorandomness. *Journal of Cryptology 4*, 151–158.

Osborne, M. J. and A. Rubinstein (1994). *A Course in Game Theory*. Cambridge, Mass.: MIT Press.

Peterson, G. L. (1982). An $O(n \log n)$ unidirectional distributed algorithm for the circular extrema problem. *ACM Transactions on Programming Languages and Systems 4*(4), 758–762.

Saks, M. E. (1989). A robust noncryptographic protocol for collective coin flipping. *SIAM Journal on Discrete Mathemantics 2*(2), 240–244.

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

Urbano, A. and J. E. Vila (2002). Computational complexity and communication: coordination in two-player games. *Econometrica 70*(5), 1893–1927.

Urbano, A. and J. E. Vila (2004). Computationally restricted unmediated talk under incomplete information. *Economic Theory 23*(2), 283–320.

Wong, E., I. Levy, L. Alvisi, A. Clement, and M. Dahlin (2011). Regret-freedom isn't free. In *Principles of Distributed Systems—15th International Conference (OPODIS 2011)*, pp. 80–95.