Distributed Protocols for Leader Election: a Game-Theoretic Perspective*

ITTAI ABRAHAM, VMWARE Research, Israel
DANNY DOLEV†, The Hebrew University of Jerusalem, Israel
JOSEPH Y. HALPERN, Cornell University, USA

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 show that we can get a fair ex post ϵ-*Nash* equilibrium if $n = 2$ in the asynchronous setting under some cryptographic assumptions (specifically, the existence of a one-way functions), using a *commitment protocol*. 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 in a completely connected network if $n > 2k$; under the same cryptographic assumptions, we can a get a k-resilient equilibrium in a completely connected network, unidirectional ring, or bidirectional ring if $n > k$. Finally, we show that under minimal assumptions, not only do our protocols give a Nash equilibrium, they also give a *sequential* equilibrium, so players even play optimally off the equilibrium path.

CCS Concepts: • **Networks → Network protocols**; • **Theory of computation → Algorithmic game theory and mechanism design**.

Additional Key Words and Phrases: leader election, *ex post* Nash equilibrium

ACM Reference Format:
Ittai Abraham, Danny Dolev, and Joseph Y. Halpern. 2018. Distributed Protocols for Leader Election: a Game-Theoretic Perspective. *ACM Transactions on Economics and Computation* 1, 1 (December 2018), 26 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

As has often been 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.

*A preliminary version of this papers appeared in *Proceedings of the 27th International Symposium on Distributed Computing*, 2013, pp. 61-75.
†Danny Dolev is Incumbent of the Berthold Badler Chair in Computer Science. Part of the work was done while the author visited Cornell University.

Authors' addresses: Ittai Abraham, VMWARE Research, Herliya, Israel, ittaia@cs.huji.ac.il; Danny Dolev, The Hebrew University of Jerusalem, School of Computer Science and Engineering, Jerusalem, Israel, dolev@cs.huji.ac.il; Joseph Y. Halpern, Cornell University, Computer Science Department, Ithaca, NY, 14853, USA, halpern@cs.cornell.edu.

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 et al. 2006; Dani et al. 2011; Fuchsbauer et al. 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 et al. 2006, 2008; Barany 1992; Ben-Porath 2003; Dodis et al. 2000; Forges 1990; Heller 2005; Izmalkov et al. 2011; Lepinski et al. 2004; McGrew et al. 2003; Shoham and Tennenholtz 2005; Urbano and Vila 2002, 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 preference of the majority. Another line of research was initiated by work on the *BAR* model [Aiyer et al. 2005]; see, for example, [Moscibroda et al. 2006; Wong et al. 2011]. Like the work in [Abraham et al. 2006, 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 considering 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 et al. 1982; Le Lann 1977; Lynch 1997; Peterson 1982]) if we assume that no agents have 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 et al. 1982; Le Lann 1977; Lynch 1997; Peterson 1982]). 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

[1] We use "process" and "agent" in this paper to describe the entities in the system, tryi 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 et al. 2011].

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 same 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: no leader will be elected if no one does anything, and this is still the case if one person votes for a particular leader. We are 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.

A standard approach for solving distributed computing problems is to implement a "trusted" mediator. Solving basic leader election with a mediator is easy: 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. The challenge is to implement this mediator in settings that have weaker communication primitives. The existing work on implementing mediators in a game-theoretic setting assumes that (a) agents have commonly-known names, (b) the network is completely connected, and (c) the network is synchronous. In the distributed computing world, the problem of leader election was studied in many settings. For example, it has been studied when agents are anonymous, when the network topology is restricted (for example, bidirectional and unidirectional rings), and when the network is asynchronous. One of the main contributions of this paper is to extend the game-theoretic mediator implementation frameworks to these more challenging settings. There has already been work applying game-theoretic ideas to leader election [Altman and Tennenholtz 2008; Antonakopoulos 2006; Gradwohl 2010]; we discuss how this work relates to ours below.

We begin by showing 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 et al. 2006, 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.

The protocols above work if the network is synchronous. What happens in an asynchronous setting? The game theory literature has largely ignored the question of asynchrony; indeed, the only work of asynchrony in a game-theoretic setting that we are aware has come from the computer science side [Antonakopoulos 2006; Friedman and Shenker 1998; Gradwohl 2010; Monderer and Tennenholtz 1999]. But asynchrony is perhaps the most common model considered in the distributed computing world, and certainly arises in many real-world settings where timing matters and there are strategic concerns. To take just one example, the standard model of the blockchain does not assume synhronized clocks and that there is a bound Δ such that messages may take up to Δ time steps to arrive. As a consequence, we cannot assume that clocks are synchronized, and agents cannot necessarily tell the order in which actions are performed.

The solution concept we consider here is a natural variant of Nash equilibrium that has been considered before in asynchronous settings [Antonakopoulos 2006; Gradwohl 2010]. 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

[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.

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 assuming 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 for a completely connected network in an asynchronous setting provided that $n > 2k$.

In a preliminary version of this paper [Abraham et al. 2013], we claimed to provide a protocol that gave a k-resilient equilibrium in unidirectional and bidirectional rings for $n > 2k$. As Yifrach and Mansour [2018] showed, the protocol was incorrect.[4] Yifrach and Mansour also provide a protocol that for rings of size n that can tolerate coalitions of size $O(\sqrt{n})$. It is an open question whether that can be improved. In the case of a competely connected network, they cannot: we show there is no fair k-resilient ex post Nash equilibrium if $n \leq 2k$ for completely connected networks; this bound also applies to undirectional and bidirectional rings.

The lower bounds assume that agents are not computationally bounded. If we make a standard assumption in the cryptography literature, namely, that one-way functions exist, then we can get fair ϵ–k-resilient equilibrium in all these topologies (i.e., no group of up to k agents can gain more than ϵ by deviating), even in the asynchronous setting, as long as $n > k$, using a *commitment protocol* [Goldreich 2001].

It is interesting to compare our results to those of Antonakapoulos [2006] and Gradwohl [2010]. They consider what has been called the *full-information model* [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 (so there is limited asynchrony). In general, the full-information model is incomparable to ours. We do not assume broadcast. Players send point-to-point messages; 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. Gradwohl considers the effect of up to k Byzantine (i.e., malicious) agents and one rational agent in the full-information model, proving both possibility and impossibility results. Gradwohl's results on leader election essentially subsume those of Antonokpoulos. In the language of Abraham et al. [2006], Gradwohl is interested in $(1, k)$-robust equilibria, which tolerate 1 rational agent and up to k Byzantine agents; Antonakopoulos is essentially concerned with $(0, 1)$-robustness. By way of contrast, we are interested in $(k, 0)$-robust equilibria, that tolerate k rational agents but no Byzantine agents. (Traditional Nash equilibrium is a (1,0)-robust equilibrium.) Like us, Gradwhol and Antonkaopoulos are interested in ex post equilibria, ones that obtain no matter what the scheduler does (although their schedulers are more limited than the ones that we consider in asynchronous settings). (Altman and Tennenholtz [2008] also consider leader election, but they focus on on a different type of solution concept, namely *weak dominance*, and do not really consider the effect of the scheduler.)

This paper 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 topologies other 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 some of the issues that arise when analyzing distributed protocols such as leader election.

[4] Actually, Yifrach and Mansour's argument showed that the leader-election protocol proposed by Afek et al. [2014] was incorrect, however, their ideas also can be applied to show that our protocol also does not work.

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 Section 4 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 5.

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 agents know its size and how agents are connected to each other. For example, 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. Agents also know all their incoming and outgoing links. 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. It is common knowledge when the protocol is initiated. 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, and messages may arrive in arbitrary order (i.e., not necessarily in the order in which they were sent). 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 $leader_i$ that can be set to some agent's id. For definiteness, we assume that $leader_i$ is initialized to a special non-integer value $\perp$. If, at the end of the protocol, there is an agent id v such that $leader_i = v$ for all agents i, and all agents agree on which agent has id v, that is, there is a unique node in the network such that all agents know that that node has id v, then we say that the agent with id v has been elected leader. If agents do not agree on the id and location of a leader, then there is no leader. In the case of a complete network, knowing which node has id v means that each agent knows which of his links is connected to the node with id v. In the case of a ring, it means that each agent knows the size of ring and his offset from v.

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 further assume that an agent i's utility is completely determined by who is the leader and whether or not there is a leader. Except for one technical result, we make no assumptions about which agent i prefers as leader. While i might prefer that he himself be the leader rather than anyone else, our results do not depend on this. However, our assumptions about the utility function do not let us express that i would prefer a protocol where he sends fewer messages, or does less computation, to one where he sends more messages or does more 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.

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, if each agent prefers to be the leader (which is possible, although we do not assume that it is necessarily the case), 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 the set of ids that 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 $leader_i = v$; 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 to lie about his id. Note that this protocol is, in the language of [Abraham et al. 2006, 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.

Round 1: send id_i to all agents
Round 2: $IDset_i := \{id_j | j$'s id received in Round 1$\}$
send $IDset_i$ to all agents
if $IDset_i = IDset_j$ for all agents j and $|IDset_i| = n$ *// n is the (known) number of agents*
then draw a random number N_i from $\{0, \ldots .n-1\}$
else halt *// punish*
Round 3: send N_i to all agents
if N_j received from all agents j
then $N := \sum_j N_j (\text{mod } n)$
set $leader_i$ to the $(N+1)$st highest id in $IDset_i$ *// choose a leader*
halt.

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

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 halts with $leader_i$ still set to $\perp$, so 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 $N = \sum_{i=1}^{n} N_i \text{ (mod } n)$ and takes the agent with the $(N+1)$st highest id in the set to be the leader. (If some agent j does not send i the same number as to others, then $leader_i$ is still set to $\perp$. Moreover, if not all agents set the leader value to be the same, by assumption, there is no chosen leader.) Call this protocol for agent i $LEAD_i^{cc}$. The pseudocode for $LEAD_i^{cc}$ is in Figure 1.

Let $\mathbf{LEAD}^{cc}$ denote the profile $(LEAD_1^{cc}, \ldots, LEAD_n^{cc})$ (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.

The following theorem summarizes the discussion above.

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 all agents use the protocol. Then clearly each agent is equally likely to become the leader, since $\sum_j N_j \text{ (mod } n)$ is uniformly distributed in $\{0, \ldots, n-1\}$. 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 $j \text{ (mod } n)$ at random, the agents agree on who the agent with the jth highest id 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. The only way that the agents in H can gain is by having all the agents choose the same leader, but biasing the choice of leader. Note that if some agent $i \in H$ does not send an id in Round 1 to some agent $j \notin H$, then there will be no leader. If the agents in H do not send the same set of ids to all the agents not in H, then if $|H| \leq n-2$, each agent $j \notin H$ will receive different sets in step 2, and thus halt with $leader_j = \perp$, making the agents in H worse off than if they do not deviate. (If $H| = n-1$, it is trivially the case that all the agents not in H get the same set of ids from all the agents in H, since there is only one agent not in H.) We still

need to argue that agents in H send their actual ids; see below. If $|H| \leq n-2$ and the agents in H deviate at step 2 in such a way that two agents not in H receive sets of random numbers with different sums from agents in H, then the agents not in H will disagree about the leader, again making agents in H worse off. If an agent $i \in H$ deviates by not taking the agent with the $(N+1)$st highest id to be the leader, then again agents in H will disagree with the agents not in H about the leader, making i worse off. It is clear that as long as all agents in H send the agents not in H the same set of ids in step 1 and sets of numbers (not necessarily random) with the same sum in Round 2, then each agent is equally likely to be the leader. Thus, an agent $i \in H$ does not gain by deviating in Round 1 by sending an id other than his actual id, since (given that all ids are equally likely to be chosen) the choice of id does not affect the probability that any specific agent will be the leader. (Here we are using the assumption that an agent's utility depends only on whether there is a leader and who the leader is and not, for example, the id of the leader. If an agent particularly likes have a leader whose id is 17, then he might say that he has an id of 17 even if this is not his actual id.) Nor does an agent in H gain by choosing a number in step 2 other than a random number, since the sum will be random as long as one agent chooses a random number, and the agents not in H will chose 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. Suppose that only some agents want to initiate the protocol. We call this spontaneous initiation *self-initiation*, although it may be due to an external signal or due to algorithmic requirements. For simplicity, we treat the choice of which agents initiate the protocol as a nondeterministic choice of nature. All the agents who self-initiate the protocol 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. 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 if i prefers to be the leader. (However, the overhead of doing may be sufficient to deter an agent from doing so; see the discussion in Section 5.) 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 one or more agents may decide to initiate leader election. Each of these initiating agents sends a "signed" message (i.e., a message with his id) to his neighbor. In the protocol described in Figure 2, if i is an initiating agent, it sends

the message Collect_Msg$\langle id_i, \langle id_i \rangle\rangle$; the first id_i represents i's signature. The second component of Collect_Msg is a sequence if ids. Messages are then passed around the ring, with each agent appending his id to the sequence *IDset* that is the second component of Collect_Msg 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, and the id of the initiator is updated. (The variable *highest* in the protocol in Figure 2 stores the name of the agent who is currently taken to be the initiator.) 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 $(N+1)$st highest id in the set to be the leader. Inappropriate messages are ignored, but if an agent does not receive the messages that he expects, then he will not complete the protocol, so $leader_i$ will remain at $\perp$, and there will be no leader elected. Specifically, this means that an agent who originated a message must get his message back n rounds and $2n$ rounds after he sent it, or get a message from an originator with a higher id. Moreover, for each of the n rounds after the originator with the highest id gets back his message for the second time, each agent i must 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 if j is k steps before i on the ring, with wraparound). Call this protocol for agent i $LEAD_i^{uni}$. The pseudocode for $LEAD_i^{uni}$ is in Figure 2.

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 a signal to initiate leader election. Thus, we are interested in what has been called an *ex post* Nash equilibrium; we want to show that, no matter what nondeterministic choices nature makes, no agent has any incentive to deviate. In the synchronous setting that we consider here, nature's only nondeterministic choice is which agent initiates leader election. So we must show that, no matter which agents initiate leader election, no agent has any incentive to deviate (even if the deviating agent knows which agents initiated the leader election 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. Once we consider asynchronous settings, nature will have yet more scope for making choices. But we will still require that in an ex post equilibrium, no agent has any incentive to deviate, no matter what nature's choice is.

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 initiate leader election, then either it has no impact (if another agent initiates the election) 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 initiates the protocol) or no leader is elected when one could have been

Upon initiating leader election: // *either self-initiated or due to message from predecessor*
$count := n$ // *when you next expect a message*
$highest := id_i$ // *who you expect the message from*
$phase := 1$ // *what kind of message you expect*
if wakened by receiving Collect_Msg$\langle id, IDset\rangle$ and $id > highest$
then $IDset := IDset \cdot id_i$ // id_i *is appended to the list IDset*
send Collect_Msg$\langle id, IDset\rangle$ to successor
$highest_i := id$
else send Collect_Msg$\langle id_i, \langle id_i\rangle\rangle$ to successor // *if election self-initiated*
In round $m > 0$ after being wakened
$count := count - 1$
if received Collect_Msg$\langle id, IDset\rangle$ from predecessor **then**
if $id > highest$ and $phase = 1$ // *message is ignored if id < highest*
then $IDset := IDset \cdot id_i$ // id_i *is appended to the list IDset*
$count := n$
$highest := id$
send Collect_Msg$\langle id, IDset\rangle$ to successor
if $id = highest$ and $|IDset| = n$ and $phase = 1$ and $count = 0$
then $phase := 2$
$IDset_i := IDset$
$count := n$
send Setup_Msg$\langle id_i, n, IDset\rangle$ to successor
if $phase \neq 1$ or ($id = highest$ and ($|IDset| \neq n$ or $count \neq 0$)) **then** halt // *punish*
if received Setup_Msg$\langle id, k, IDset\rangle$ from predecessor **then**
if $id = highest$ and $|IDset| = n$ and $phase = 1$ and $count = 0$
then $phase := 2$
$IDset_i := IDset$
$count := k - 1$
if $count > 1$ **then** send Setup_Msg$\langle id, k-1, IDset\rangle$ to successor
if $id \neq highest$ or $IDset \neq n$ or $phase \neq 1$ or $count \neq 0$ **then** halt // *punish*
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 successor
$phase := 3$
$count := n$
if $phase = 3$ and $count = n - j > 0$ **then**
if received Rand_msg$\langle id, N'\rangle$ from predecessor and id is jth id prior to i in $IDset_i$
then forward Rand_msg$\langle id, N'\rangle$ to successor
$N_{id} := N'$
else halt // *punish*
if $phase = 3$ and $count = 0$ and $leader_i \neq \perp$
$N := \sum_j N_j \pmod n$
set $leader_i$ to the $(N+1)$st highest id in $IDset_i$
halt

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

(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 agents disagree on the set of ids, either it has no impact, or agents will disagree on 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 initiates the leader-election process, 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 phase 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 k-resilient equilibrium as before.

3.4 Asynchronous completely connected network

We now consider an asynchronous setting. With asynchrony, a scheduler gets to choose when agents are scheduled and how long it takes for a message to be delivered. When we talk about ex post equilibrium, we mean that, no matter what the choices of the scheduler are, even knowing these choices (and the choice of which agent initiated the leader election), agents have no incentive to deviate.

As we said in the introduction, in the preliminary version of this paper [Abraham et al. 2013] we claimed to provide a protocol that gave a k-resilient equilibrium in unidirectional and bidirectional rings for $n > 2k$. As Yifrach and Mansour [2018] showed, the protocol was incorrect. But we can provide a protocol that gives a k-resilient equilibrium in completely connected asynchronous networks if $n > 2k$, using ideas from secure multi-party computation and secret sharing [Ben-Or et al. 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.[5] 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 halts with $leader_i = \perp$, so no leader is chosen.) 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

[5] We can get the result for arbitrary n by just choosing a prime $n' > n$ and repeating the protocol below until the number N that is the outcome of the protocol below is between 0 and $n-1$.

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 $(N+1)$st highest id as leader; otherwise, i sets $leader_i$ to $\perp$. Call this protocol $A\text{-}LEAD_i^{cc}$.

The following theorem summarizes the discussion above.

THEOREM 3.3. *If $n > 2k$ then* **A-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 $|H| \leq k$. As in the proof of Theorem 3.1, the agents in H must send the same set of ids to all agents not in H, otherwise no leader will be chosen. 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). The coalition H has only $|H|$ points on the polynomial f_j for $j \notin H$ before it sends *DONE* message, since, for each $j \notin H$, all the agents in H together know only $f_j(i)$ for $i \in H$. Since f_j has degree $k+1$, $|H| \leq k$, and k points on a polynomial of degree $k+1$ do not suffice to determine the polynomial, the agents in H have no idea of what $N_j = f_j(0)$ is when they send a *DONE* message; all values of N_j are compatible with their information and are equally likely.

Clearly agents in H are worse off if they do not send a *DONE* message to an agent $j \notin H$ (since then j will leave $leader_j$ at $\perp$, so there will be no leader elected). Once an agent i receives n *DONE* messages, each agent i can send s_i. Since there are at least $k+1$ agents not in H, there is a unique polynomial that interpolates the values s_i send by agents $i \notin H$. Call this polynomial h. It is easy to see that $h(0)$ is uniformly distributed, since $h(0) = \sum_j f_j(0)$, and the values $f_j(0) = N_j$ are chosen uniformly at random by agents $j \notin H$. Now if an agent $j \in H$ does not send all the agents not in H values from some polynomial, then either there is no leader chosen or the choice does not affect the fact that the value of the outcome is uniformly distributed. So the agents in H cannot gain by deviating from the protocol at any step. ∎

3.5 A matching lower bound

We now show that Theorem 3.3 is the best we can hope for; we cannot find a fair ex post k-resilient strategy if $n \leq 2k$ provided that all agents have *nontrivial preferences*; although they do not need to prefer that they themselves become leader, they cannot be completely indifferent as to who becomes leader. (If agents are completely indifferent as to who is the leader, it is easy to see that there is a fair Nash equilibrium for all values of n. The coalition are perfectly happy with a fair outcome in this case.)

THEOREM 3.4. *If $n \leq 2k$, then there is no fair, ex post k-resilient equilibrium for an asynchronous completely connected network (resp., unidirectional ring, bidirectional ring).*

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 in the full-information model (discussed in the introduction) [Ben-Or and Linial 1985; Boppana and Narayanan 2000; Saks 1989]. In that model, the problems are known not to be solvable if $n = 2$ and $k = 1$. 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. We then extend the argument to arbitrary k.

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. We remark that, although in general the full-information model is quite different from ours, if there are only two players and nature uses strategy ρ, then they are essentially identical.

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, the only state that player 1 could have if player 0's state is h_0 is either state h_1 (where 1 has not yet received the message that 0 sent at time t) or the state that extends h_1 in which 1 has received the message sent by 0 at time t. It follows that 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 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' (since the history up to time $t - 1$ is determined by h_0), 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 history h'' in r and r' using strategy σ_1, it must be the case that σ_1 randomizes. But then it follows from our assumption that all agents have nontrivial preferences that 1 has a profitable deviation. If 1 prefers that he himself becomes leader, then he is 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. Similarly, if 1 prefers that 0 becomes leader, then he is better off deviating when his coin lands as it does in r', and pretending that his coin landed as it did in r.

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 that there must be a Nash equilibrium for 2 players conditional on nature using ρ where player 0 has a probability $\lfloor (n+1)/2 \rfloor / n$ of being the leader, contradicting our earlier argument. Player 0 simply simulates what A does, and 1 simulates what B does. That is, 1 can simulate what the agents in A do when they send message internally, and 1 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, 1 sends a message to 0 in the simulation; similarly, when agent $\lfloor n/2 \rfloor$ sends a message to agent $\lfloor n/2 + 1 \rfloor$, 0 sends a message to 1. 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. ∎

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 and make one further (quite reasonable!) assumption about the utility function, using ideas of

[Boppana and Narayanan 2000; Saks 1989], we can get a stronger result: we cannot even achieve an ϵ–k-resilient equilibrium for sufficiently small ϵ.[6] Say that agents have a *preference for leadership* if each agent i strictly prefers outcomes where he is the leader to outcomes where he is not the leader.

THEOREM 3.5. *If $n \leq 2k$ and each agent has a preference for leadership, then there exists an $\epsilon > 0$ (which depends only on the utility function in the game) such that for all ϵ' with $0 < \epsilon' < \epsilon$, there is no fair, ex post ϵ'–k resilient equilibrium for an asynchronous completely connected network (resp., unidirectional ring, bidirectional ring).*

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 expected utility is $(B_i + G_i)/2$. Let $\epsilon = \min((G_0 - B_0)/2, (G_1 - B_1)/2))$. Since agents have a preference for leadership, we must have $\epsilon > 0$. 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 full-information model. As we observed, in 2-player games with the particular adversary considered in the proof of Theorem 3.4, the the full-information model is essentially equivalent to the asynchronous model we consider in this paper.

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 full-information model if more than half the players are faulty. We now formalize the details.

In Saks' formalization, a distributed protocol in the full-information 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).[7] 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.[8] 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 3.4. We omit details here. ∎

[6] We believe that this result should hold even without the restriction to bounded protocols, but have not proved it yet.
[7] 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.
[8] The Boppana-Narayanan proof assumes that the probability $p_L = 1/2$, but the identical proof works for an arbitrary p_L.

3.6 Doing better with cryptography

In the impossibility result of Section 3.5, 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 one-way functions exist (a standard assumption in cryptography), then we can get a fair ϵ–k-resilient equilibrium in all these topologies, even in the asynchronous settings, as long as $n > k$.

The key idea of the earlier protocol for the completely connected network is that i essentially commits to N_i by sending shares of the polynomial f_i (where $f_i(0) = N_i$ to all the agents), 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 [Goldreich 2001]. In a commitment protocol, an agent Alice (the *sender*) commits to a number m in such a way that another agent Bob (the *receiver*) 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. Roughly speaking, this idea is implemented using one-way functions as follows: When Alice commits to m, she sends Bob $f(m)$ for a one-way function f—a function that is too hard for Bob to invert, since he is polynomially bounded—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.

The basic idea of commitment protocols go back to Blum [1981]. Many commitment protocols are known (see [Goldreich 2005] for discussion). All of these protocols have the property that, given ϵ, we can set the security parameter of the protocol so that the probability of the protocol not satisfying the specification (i.e., the probability that Bob is able to compute m from $f(m)$ or that Alice can reveal a number $m' \neq m$ that Bob accepts) is less than ϵ. Such a commitment protocol suffices for leader election if there are only two agents. However, if $n > 2$, it does not suffice to have a basic commitment protocol.

We need to run $n(n-1)$ commitment protocols concurrently, and we want to make sure that the outcomes of these commitment protocols are independent. Thus, the commitment protocol that we use must be *non-malleable* [Dolev et al. 2000]. To understand the issue, suppose that the agent i just before the originator on the ring knows the value N_j to which j has committed (i.e., i knows $f(N_j)$) before committing to his own random choice. Metaphorically speaking, 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 protocol 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, as we suggested above, since we are running $n(n-1)$ commit protocols concurrently, we want not just non-malleability, but *concurrent* non-malleability, The reason that we need concurrent non-malleability is that we do not want information from one commitment protocol to be used in another one.

Concurrent non-malleability was originally defined by Pass and Rosen [2008], and refined by Lin, Pass, and Venkitasubramanian [2008]. There are a number of concurrently non-malleable commitment protocols known. We focus on one due to Lin and Pass [2015], which runs in a constant number of rounds and provides the Lin-Pass-Venkitasubramanian refinement of concurrent non-malleability. It has a further advantage: it works even in the presence of rushing, that is, even if one agent can delay sending his kth round message until after he has heard the kth round message of all

other agents. We use the Lin and Pass protocol as a black box here, first sketching how it works in an asynchronous unidirectional ring.

As we said, the Lin and Pass commitment protocol (and all others) does not guarantee the commitment properties; rather, it guarantees that the properties hold with all but negligible probability. To make this precise, define a function $\epsilon : \mathbf{N} \to [0, 1]$ to be *negligible* if, for every constant $c \in \mathbf{N}$, $\epsilon(m) < m^{-c}$ for sufficiently large m. Roughly speaking, this says that ϵ is smaller than any inverse polynomial. Lin and Pass provide a protocol for which there is a negligible function ϵ such that, given as input a parameter m, the protocol guarantees that the commitment properties hold with probability at least $1 - \epsilon(m)$.

Before going on, since we are considering computationally bounded agents, we must define more carefully the equilibrium notion that we are interested in. As has been observed before [Dodis et al. 2000; Halpern et al. 2016], when considering computationally bounded agents, new subtleties arise. To talk about "polynomially bounded agents", we must have some parameter m such that the agents are polynomial in m. Thus, Dodis, Halevi, and Rabin [2000] defined a notion of computational Nash equilibrium with respect to what they called an *extended game*, which is essentially a game extended with a security parameter m (a non-negative integer), which players are given as input, and requiring that agents strategies by polynomial in the security parameter. (Halpern, Pass, and Seeman [2016] used a similar approach, defining computational Nash equilibrium with respect to a sequence $\mathcal{G} = (\Gamma_1, \Gamma_2, \ldots)$ of games, all involving the same set of players. We can view Γ_k as an instance of Dodis, Halevi, and Rabin's extended game with security parameter k.)

Definition 1. *A strategy in an extended game is* PPT computable *if there exists a probabilistic Turing machine that given as input the security parameter m and a history, computes an action in time polynomial in m. A strategy profile σ is a* computational k-resilient equilibrium *of an extended game Γ if each strategy σ_i is PPT computable and for all coalitions K of players with $|K| \leq k$ and all PPT computable strategies σ'_i for players $i \in K$, there exists a negligible function ϵ (that may depend on σ'_K) such that $u_i(\sigma'_K(m), \sigma_{-K}(m)) \leq u_i(\sigma(m)) + \epsilon(m)$.*

Thus, in a computational k-resilient equilibrium, all players use PPT computable strategies and a coalition of players of size at most k using PPT computable strategies can gain no more than a negligible amount by deviating. In general, we cannot avoid the use of a negligible function when considering computational equilibria. For example, if the strategy involves encryption, for every PPT computable strategy, there may be another PPT strategy that does a little better by spending a little longer trying to do decryption.

Our goal is to show that there is a computational k-resilient equilibrium for leader election in asynchronous settings as long as $n > k$. We start with a unidirectional ring. We proceed as follows. Given a security parameter m, all agents will use the Lin-Pass protocol with security parameter m. (This is the only role that the security parameter plays in the protocol.) The first agent i that initiates leader election commits to a value N_i by running a separate commitment on N_i with each agent j, using the Lin-Pass protocol. Agent i sends the first-round message according to the Lin-Pass commitment protocol to each agent j around the ring. (This is taken to be one long message of the form $\langle id_{j_1}, m_1, \ldots, id_{j_{n-1}}, m_{n-1} \rangle$, where the message m_k intended for the agent with id j_k is tagged by id_{j_k}.) Thus, there will be $n(n-1)$ commitment protocols proceedings concurrently.[9] Upon receiving such a message, agent j appends his first-round message to each other process j', committing to some value N_j. If more than one agent spontaneously initiates a leader-election protocol, then agent j does not forward messages initiated by agent i if it has already forwarded messages initiated by an agent with a higher id (or if it has a higher id than i and it initiated a message). After the

[9]This presumes that all ids are known. If not, then there is a preliminary phase of the ring where all agents append their ids to a message that goes around the ring. After this phase, all agents know all the ids of agents in the system.

message reaches i again, all agents acting in the role of sender have initiated $n-1$ commitment protocols. The message is then passed around the ring again, with each agent j adding first-round messages for the $n-1$ protocols where it is the receiver. Messages keep being passed around the ring, with each agent appending messages for all the commitment protocols in which he is involved, for M rounds, where M is the (constant) number of rounds required to run the Lin-Pass commitment protocol. As for the other protocols, an agent i does not send its round k message until it has received a round k message from its predecessor (or a round $k-1$ message from its predecessor, in case it is the initiator). This message should include the round k messages sent by all agents from the initiation up to i's predecessor on the ring, and the round $k-1$ messages sent by all agents from i's successor on the ring up to the initiator. Agent i should check that all these messages are appropriate, in the sense that includes all round k and round $k-1$ that should have been to it were indeed sent, and that there were messages that could have been sent in the appropriate round according to the Lin-Pass protocol in all the protocols that i is running concurrently. If i receives an inappropriate message, it decides $\perp$. Note that, when running on the ring, i receives some round k messages before it has sent its own round k message. This is not a problem, because the Lin-Pass protocol tolerates rushing.

In running the protocol, all agents first commit to a value and then reveal it to each agent (since this is a property of the Lin-Pass protocol). Thus, after M circuits of the ring (Mn rounds), all agents will know all values of N_i (assuming that all agents follow the protocol). They then continue as in protocol $\mathbf{LEAD}^{uni}$, computing $N = \sum_{i=1}^{n} N_i$ and choosing as leader the agent with the $(N+1)$st highest id. Of course, if the protocol does not complete (because the some agent does not forward messages), then no leader is chosen; similarly, if agent notices another agent not following the protocol, then no leader is chosen. Call the resulting protocol $\mathbf{A\text{-}LEAD}^{com,\,uni}(\epsilon)$.

Theorem 3.6. *If one-way functions exist, then* $\mathbf{A\text{-}LEAD}^{com,\,uni}$ *is a fair, computational 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. Suppose that we have a coalition H with $|H| < n$. Clearly agents in H do not gain by not forwarding messages along the ring, because then no leader is chosen. Similarly, they do not gain by sending inappropriate messages. The properties of the Lin-Pass protocol guarantee that no matter how a group of at most k agents deviate, there is a negligible probability ϵ (that may depend on the deviation) such that in the game with security parameter m, all the commitment properties hold with probability at least $1-\epsilon(m)$. This is true even if the agents in K collude (since collusion does not affect the non-malleability) and even if the agents learn all the round-h messages that other agents sent before sending their own round-h messages. Thus, with probability at least $1-\epsilon(m)$, a deviation in the game with security parameter m results in either no leader or a leader being chosen fairly. While it is possible that agent $i \in H$ will commit to a different values N_{ij} and $N_{ij'}$ to agents j and j', unless $\sum_{i\in H} N_{ij} = \sum_{i\in H} N_{ij'}$ for all j and j' not in H, j and j' will choose different leaders, so i will be worse off. It easily follows that all agents $i \in H$ maximize their utility by sending the same value N_i to all agents. And, as in all other protocols, as long as at least one agent chooses a value at random, the final choice will be random, so the agents in H do not gain by not choosing a value at random. Thus, it follows the the coalition cannot gain more than $M\epsilon(m)$ by deviating, where M is the difference between the best utility that can get the leader-election protocol and the worst utility that they can get with a fair outcome. Since $M\epsilon$ is negligible if ϵ is, it follows that deviating agents can gain at most negligible utility by deviating. It follows that $\mathbf{A\text{-}LEAD}^{com,\,uni}$ is an k-resilient ex post computational equilibrium. ∎

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-LEAD**$^{com,\,uni}$. Thus, we have the following result:

Theorem 3.7. *If one-way functions exist, then there is a a fair, computational k-resilient ex post equilibrium in an asynchronous bidirectional ring (resp., completely connected network) for all $k < n$.*

4 SEQUENTIAL EQUILIBRIUM

Our Nash equilibrium 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 play such a punishment strategy? Note that agents punish only off the equilibrium path—punishment 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 et al. 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 et al. 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.[10] 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 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 $\mathcal{I}_K(h) = \cap_{i \in K} \mathcal{I}_i(h)$.[11]

[10] 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.

[11] Note that if we took $\mathcal{I}_i$ to be a partition of the histories where only i moves, and we assume (as is standard in game theory, and is also typically the case in asynchronous systems) that only one agent moves in each history, then $\mathcal{I}_K$ would be empty for coalitions consisting of more than one agent. This is exactly why we took $\mathcal{I}_i$ to be a partition of all histories.

Definition 2. *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 in a straightforward way on the equilibrium path; 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 3. *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 4. 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* Γ*, all* K *such that* $|K| \leq k$*, and all information sets* $I_K \in \mathcal{I}_K$*, we have*

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

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

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].

To get a k-sequential equilibrium, we seem to need an additional constraint on the utility function. Up to now, we have not made any assumptions about the agents' utility for outcomes where there is no leader. For this section, we assume that all outcomes where there is no leader have equal utility for each agent i. While this does not seem unreasonable, note that it means that i is indifferent as to how much computation is expended. For example, i would have to give equal utility to an outcome where $leader_i = \perp$ and i sends many messages to one where $leader_i = \perp$ and i sends no messages. As we point out in the proof, our sequential equilibrium would also work if we assumed that, among outcomes where there was no leader, i preferred to minimize computation.

To define the k-belief system that we need, we modify each protocol P defined thus far to get a sequence of completely mixed protocols P^m, $m = 1, 2, \ldots$ that converges to a protocol P^*. Not only can we use this sequence of protocols to define the k-belief system, the limit protocol P^* will be a fair ϵ–k-resilient sequential equilibrium in the setting that P is a fair ϵ–k-resilient NE. For ease of exposition, we assume that each agent can perform only finitely many possible actions at each information set (although the proof can be modified in a straightforward way to deal with the case that agents can perform countably many actions). We proceed as follows.

Consider an information set I for player i. If I is an information set that includes histories on the equilibrium path, then when running P^m, with probability $1 - \frac{1}{m}$, i performs the same action a as he does with P; with probability

$\frac{1}{m} - \frac{1}{m^2}$, performs some action that where i sets $leader_i$ to $\perp$, all such actions being performed with equal likelihood;[12] the remaining probability $\frac{1}{m^2}$ is apportioned equally over all remaining possible actions at I. If I is disjoint from the equilibrium path, then i knows that there has been a deviation. In this case, with probability $1 - \frac{1}{m}$ i halts (note that if i detects a deviation in any of our protocols, i also halts); the remaining probability $1/m$ is again split uniformly over all the remaining possible actions at I.

It is not hard to see that this sequence of strategies generates k-beliefs with the properties we need. Specifically, if a rational player i is a member of a coalition H of size at most k (where $n > 2k$ if $P = \textbf{A-LEAD}^{cc}$ and $n > k$ in all other cases), and i is at an information set I that is consistent with equilibrium play (i.e., there is a history in I on the equilibrium path, 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 agents j not in H have been fact playing σ_j; as we have shown, in this case his best response is σ_i. On the other hand, if I is an information set where i detects a deviation from equilibrium play, then i will assign probability 1 to some other agent j having set $leader_j$ to $\perp$ (since i believes that the first agent j to deviate from equilibrium play, when using P_j^m, is much more likely to set $leader_j$ to $\perp$ than not to do so so). Since we have assumed that all outcomes where there is no leader have equal utility for i, what i does at I does not affect his utility, so halting is a best response. (Note that it is also the response that minimizes computation, so if i's utility is such that, among outcomes where there is no leader, he prefers to minimize computation, this strategy is also a sequential equilibrium.)

Summarizing this discussion, we have the following result.

Theorem 4.1. *Given the assumptions on utility above,*

(a) *in a synchronous systems if $n > k$, then there is a fair k-resilient sequential equilibrium in a completely connected network, a unidirectional ring, and a bidirectional ring;*
(b) *in an asynchronous systems if $n > 2k$, then there is a fair k-resilient sequential equilibrium in a completely connected network.*

We believe that we can also get an analogue to Theorem 3.6 for sequential equilibrium. However, to do this formally, we would need to define the notion of computational k-resilient sequential equilibrium. This turns out to be rather subtle. For example, there are information sets in the extended game where an agents is able to break the commitment scheme. Clearly, in these information sets, the agent is better off deviating. However, intuitively, these information sets occur with only negligible probability. We essentially want to ignore them. Halpern, Pass, and Seeman [2016] (HPS from now on) provide a definition of computational sequential equilibrium that allows us to do this. Going through all the details of the formal definition was take us too far afield, so we just sketch the ideas here. (In our sketch, we slightly modify the definitions of HPS to be consistent with the presentation of this paper.)

HPS define computational sequential equilibrium in extended games.[13] They assume that the extended game Γ^+ of interest represents an *underlying game* Γ. For example, if the extended game involves a commitment protocol (as ours does), in the underlying game, players just put the numbers that they are committing to in an envelope; that is, they have an action that corresponds to putting a number m in an envelope. All the histories that agree except on the number that Alice put in the envelope are in the same information set for Bob in the underlying game. Opening the envelope to reveal the number is another move in the underlying game. Thus, the underlying game essentially describes an "ideal"

[12] (Note that there may be many actions that involve setting $leader_i$ to $\perp$, since actions involve not just setting $leader_i$, but also possibly sending messages and setting the values of other variables. Since we have assume that there are only finitely many actions, we can assign them all equal probability.
[13] Actually, as mentioned earlier, they consider a sequence of games rather than an extended game, but we can identify the sequence with an extended game.

version of what the computational commitment protocol is intended to achieve. There is a surjective mapping f from histories in the extended game to histories in the underlying game (intuitively, the history $f(h)$ in the underlying game is an idealized version of the history h in the extended game), and a mapping $\mathcal{F}$ from strategies in the underlying game to PPT strategies in the extended game. The mappings f and $\mathcal{F}$ must satisfy a number of natural conditions. If they do, then Γ^+ is said to $(f, \mathcal{F})$-*represent* Γ.

The next step in defining computational sequential equilibrium is to consider a definition equivalent to the standard definition of sequential equilibrium.

Definition 5. *Given a partition $\vec{\mathcal{I}}$, $\vec{\sigma}$ is an $\vec{\mathcal{I}}$-*sequential equilibrium *of Γ if there exists a sequence of completely mixed strategy profiles $\sigma^1, \sigma^2, \ldots$ converging to σ and a sequence $\delta_1, \delta_2, \ldots$ of nonnegative real numbers converging to 0 such that, for each player i and each set $I \in \mathcal{I}_i$, σ_i^n is a δ_n-best response to σ_{-i}^n conditional on I.*

It is well known [Kreps and Wilson 1982] that Definition 5 is equivalent to the standard definition of sequential equilibrium if $\mathcal{I}$ consists of all the information sets in the game. In an extended game, we do not want to take $\mathcal{I}$ to consist of the information sets. Two histories where Alice has committed to different numbers will be in different information sets in the extended game, but Bob may still not be able to distinguish them computationally. Intuitively, we want to take $\mathcal{I}$ to be a "computational information partition", so histories in the same cell should be computationally indistinguishable. Roughly speaking, the HPS definition of sequential equilibrium says that if the extended Γ^+ represents an underlying game Γ with respect to functions f and $\mathcal{F}$ in an appropriate sense, then if σ is a sequential equilibrium of Γ^+ if it is an $\mathcal{I}'$-sequential equilibrium in the sense of Definition 5, where $\mathcal{I}' = (\mathcal{I}_1', \ldots, \mathcal{I}_n')$ and $\mathcal{I}_i'$ consists of sets of the form $f^{-1}(I)$ where I is an i-information set in the underlying game Γ, and, as in the definition of computational Nash equilibrium, deviations are restricted to PPT strategies. We can extend this definition to computational k-resilient sequential equilibrium by consider K-information sets (with respect to $\mathcal{I}'$), for $|K| \leq k$.

HPS show that if Γ^+ $(f, \mathcal{F})$-represents Γ and and σ is a sequential equilibrium in Γ, then $\mathcal{F}(\sigma)$ is a computational sequential equilibrium in Γ^+. This result can be extended to k-sequential equilibrium. It is straightforward to show that we can get a k-sequential equilibrium in the underlying game corresponding to the extended game with the commitment scheme described in Section 3.6. It follows that we have a computational k-sequential equilibrium in $\mathcal{G}^+$.

5 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. 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 to consider when applying game-theoretic ideas to distributed computing is 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. Indeed, as shown by

Halpern and Vilaca [2016], it is impossible to obtain a fair ex post Nash equilibrium for the consensus problem, even in synchronous systems, if there can be even one crash failure. But if we cannot get a fair ex post equilibrium, what are reasonable alternatives? Halpern and Vilaca put some arguably reasonable probabilistic constraints on nature's behavior, specifically, constraints on the probability of failures and the failure pattern (e.g., how likely is it that a particular subset of agents get a message from agent i in the round that agent i fails). It would be good to get a more general understanding of the effect of various probabilistic constraints on nature's behavior on the existence of NE, especially in an asynchronous setting. More generally, are there other reasonable solution concepts that should be considered?

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 et al. 2006, 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. As we said earlier, Altman and Tennenholtz [2008], Antonakopoulos [2006], and Gradwohl [2010] show that we can indeed handle faulty agents in leader election if there is one rational agent. It would be interesting to see under what conditions a (k, t)-robust equilibrium could be obtained.
  Afek et al. [2014] and Bei, Chen, and Zhang [2012] provide protocols for consensus with crash failures and rational agents. However, Afek et al.'s protocol works only under strong assumptions about agents' preferences, such as an agent having a strict preference for outcomes where it learns the input of other agents, while Bei, Chen, and Zhang require that their protocol be robust to deviations (that is, it achieves agreement even if rational agents deviate), a requirement that seems unreasonably strong.[14] As the work of Halpern and Vilaca [2016] shows, dealing with failures seems to add significant complexity.
- 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. But this only works in the absence of failures. As mentioned above, Halpern and Vilaca [2016] provide some results for consensus with crash failures, but there is clearly much more that can be done here. Afek et al. [2014], building on our preliminary work [2013], provide resilient protocols for some basic building blocks, *knowledge sharing* and *wake-up*, and use them to get protocols consensus, renaming, and leader election. Gradwohl [2010] extended the ideas for his leader election to the problem of *random sampling*, where players must jointly sample from a set of m items, but may have preferences regarding which items are chosen. However, we believe that these results just scratch the surface; much more can be done in terms of analyzing standard distributed computing protocols from a game-theoretic perspective.
- As we mentioned in the introduction, Yifrach and Mansour [2018] provide a protocol that for rings of size n that can tolerate coalitions of size $O(\sqrt{n})$. The only lower bound we have is that to achieve k-resiliency in a ring, we

[14] There seem to be other problems with Bei, Chen, and Zhang's protocol; see [Halpern and Vilaca 2016] for discussion.

require $n > 2k$. We conjecture that that we can achieve in k-resiliency in a ring of size $O(k)$, but proving this (or establishing a better lower bound) seems nontrivial.

- In [Abraham et al. 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 et al. 2008]) as long as $n > k$. Thus, we beat the lower bound of [Abraham et al. 2008]. There is no contradiction here. The lower bound of [Abraham et al. 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 et al. 2006, 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.

ACKNOWLEDGMENTS

We thank Rafael Pass for numerous discussion regarding concurrent non-malleability and for pointing out that the utility function can depend only on who is chosen as leader. Dolev's work was supported in part by the HUJI Cyber Security Research Center in conjunction with the Israel National Cyber Bureau in the Prime Minister's Office and by AFOSR under grant FA9550-09-1-0266. Halpern's work was supported in part by NSF under grants IIS-0534064, IIS-0812045, IIS-0911036, and CCF-1214844, by AFOSR under grants FA9550-08-1-0438, FA9550-09-1-0266, and FA9550-12-1-0040, by ARO under grants W911NF-09-1-0281 and W911NF-17-1-0592, and by a grant from the Open Philanthropy Foundation.

REFERENCES

I. Abraham, 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*. 53–62.

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

I. Abraham, D. Dolev, and J. Y. Halpern. 2011. On the Power of Cheap Talk. (2011). Unpublished manuscript.

I. Abraham, D. Dolev, and J. Y. Halpern. 2013. Distributed protocols for leader election: a game-theoretic perspective. In *Proc. 27th International Symposium on Distributed Computing*. 61–75.

Y. Afek, Y. Ginzberg, S.L. Feibish, and M. Sulamy. 2014. Distributed computing building blocks for rational agents. In *Proc. 33rd ACM Symposium on Principles of Distributed Computing*. 406–415.

A. S. Aiyer, 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)*. 45–58.

A. Altman and M. Tennenholtz. 2008. Strategyproof deterministic lotteries under broadcast communication. In *Proc. Seventh International Joint Conference on Autonomous Agents and Multiagent Systems*. 1549–1552.

S. Antonakopoulos. 2006. Fast leader-election protocols with bounded cheaters' edge. In *Proc. 38th ACM Symposium on Theory of Computing*. 187–196.

I. Barany. 1992. Fair distribution protocols or how the players replace fortune. *Mathematics of Operations Research* 17, 2 (1992), 327–340.

X. Bei, W. Chen, and J. Zhang. 2012. Distributed consensus resilient to both crash failures and strategic manipulations. (2012). Available at http://arxiv.org/abs/1203.4324; version 3.

M. Ben-Or, S. Goldwasser, and A. Wigderson. 1988. Completeness theorems for non-cryptographic fault-tolerant distributed computation. In *Proc. 20th ACM Symp. Theory of Computing*. 1–10. https://doi.org/10.1145/62212.62213

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

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

M. Blum. 1981. Coin flipping by telephone. In *CRYPTO 81*. 11–15.

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

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

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

Y. Dodis, S. Halevi, and T. Rabin. 2000. A Cryptographic Solution to a Game Theoretic Problem. In *CRYPTO 2000: 20th International Cryptology Conference*. Springer-Verlag, 112–130.

D. Dolev. 1982. The Byzantine Generals Strike Again. *Journal of Algorithms* 3, 1 (1982), 14–30.

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

D. Dolev, 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 (1982), 245–260.

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

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

E. Friedman and S. Shenker. 1998. Learning and implementation on the internet. Working paper.

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

O. Goldreich. 2001. *Foundations of Cryptography, Vol. 1.* Cambridge University Press, Cambridge, U.K.

O. Goldreich. 2005. *Foundations of Cryptography: A Primer.* Now Publishers, Inc.

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

R. Gradwohl. 2010. Rationality in the full-information model. In *Proc. 7th Theory of Cryptography Conference (TCC '10)*. 401–418.

J. Y. Halpern, R. Pass, and L. Seeman. 2016. Computational extensive-form games. In *Proc. 17th ACM Conference on Electronic Commerce (EC '16)*. 681–698.

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

J. Y. Halpern and X. Vilaca. 2016. Rational consensus: extended abstract. In *Proc. 35th ACM Symposium on Principles of Distributed Computing*. 137–146.

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

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

J. Katz and C.-Y. Koo. 2006. On Expected Constant-Round Protocols for Byzantine Agreement. In *CRYPTO 2006: 26th International Cryptology Conference*. 445–462.

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

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

M. Lepinski, 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*. 1–10.

H. Lin and R. Pass. 2015. Constant-round nonmalleable commitments from any one-way function. 62, 1 (2015).

H. Lin, R. Pass, and M. Venkitasubramian. 2008. Concurrent non-malleable commitments from any one-way function. , 571–588 pages.

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

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

R. McGrew, 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)*. 59–51.

D. Monderer and M. Tennenholtz. 1999. Distributed Games. *Games and Economic Behavior* 28 (1999), 55–72.

T. Moscibroda, 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*. 35–44.

M. J. Osborne and A. Rubinstein. 1994. *A Course in Game Theory.* MIT Press, Cambridge, MA.

R. Pass and A. Rosen. 2008. Concurrent non-malleable commitments. 37 (2008), 1891–1925.

G. L. Peterson. 1982. An $O(n \log n)$ unidirectional distributed algorithm for the circular extrema problem. 4, 4 (1982), 758–762.

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

Y. Shoham and M. Tennenholtz. 2005. Non-Cooperative Computing: Boolean Functions with Correctness and Exclusivity. *Theoretical Computer Science* 343, 1–2 (2005), 97–113.

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

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

E. Wong, 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)*. 80–95.

A. Yifrach and Y. Mansour. 2018. Fair leader election for rational agents in asynchronous rings and networks. In *Proc. 37th ACM Symposium on Principles of Distributed Computing*.