Fault-Tolerant Distributed Computing in Full-Information Networks

Shafi Goldwasser[*]
CSAIL, MIT
Cambridge MA, USA

Elan Pavlov
MIT
Cambridge MA, USA

Vinod Vaikuntanathan[*]
CSAIL, MIT
Cambridge MA, USA

September 23, 2006

Abstract

In this paper, we use random-selection protocols in the full-information model to solve classical problems in distributed computing. Our main results are the following:

- An $O(\log n)$-round randomized Byzantine Agreement (BA) protocol in a synchronous full-information network tolerating $t < \frac{n}{3+\epsilon}$ faulty players (for any constant $\epsilon > 0$). As such, our protocol is asymptotically optimal in terms of fault-tolerance.
- An $O(1)$-round randomized BA protocol in a synchronous full-information network tolerating $t = O(\frac{n}{(\log n)^{1.58}})$ faulty players.
- A compiler that converts any randomized protocol Π_{in} designed to tolerate t *fail-stop* faults, where the *source of randomness of* Π_{in} *is an SV-source*, into a protocol Π_{out} that tolerates $\mathsf{min}(t, \frac{n}{3})$ *Byzantine* faults. If the round-complexity of Π_{in} is r, that of Π_{out} is $O(r \log^* n)$.

Central to our results is the development of a new tool, "audited protocols". Informally "auditing" is a transformation that converts *any* protocol that assumes built-in broadcast channels into one that achieves a slightly weaker guarantee, *without assuming broadcast channels*.

We regard this as a tool of independent interest, which could potentially find applications in the design of simple and modular randomized distributed algorithms.

[*]Supported by NSF grants CNS-0430450 and CCF0514167.

1 Introduction

The problem of how n players, some of who may be faulty, can make a common random selection in a set, has received much attention. The challenge is that the faulty players may form a coalition and deviate arbitrarily from the prescribed protocol. Despite this malicious behavior of some of the players, we want to select an element of a set "as randomly as possible". This problem was studied in various network models: the *private channels* model where players can communicate via perfectly private pairwise communication channels; the *computational model* where the faulty players are assumed to be computationally bounded and cryptographic primitives are assumed to exist; and the *full information* model where no assumptions are made on the existence of private channels nor are the faulty players computationally restricted.

When achievable, random selection is not only an end in itself, but is a useful building block in solving other distributed tasks. Generally speaking, the paradigm of design is to first construct by random selection protocols a source of randomness and then prove correctness and security of protocols under the assumption that all players use this source. The most striking example of this paradigm is the progression of works by Ben-Or, Rabin, Bracha, Dwork-Shmoys-Stockmeyer, and Feldman-Micali [2, 26, 8, 13, 15] on the Byzantine agreement problem in the computational and the private channels model. Ultimately, [15] achieved expected *constant-round* randomized protocols for Byzantine agreement, by first constructing global common coins in a constant number of rounds, and then applying the round preserving reduction by [26, 13] of Byzantine agreement to constructing global common coins. This paradigm was also used in the context of secure computation [6, 19, 18, 3].

The focus of our work will be, in contrast, the **full information** model. We will show how to use a variety of random selection protocols, to address classical questions in fault tolerant distributed computing in the full information model.

We remark that achieving results in the full information model is important, as these results hold unconditionally. Currently, all results in the computational model hold only under intractability assumptions such as the existence of one-way functions. The results in the private channels model are conditioned on the availability of private physical communication channels between pair of players. This elegant abstraction is implemented by resorting to secure encryption, the existence of which is again based on intractability assumptions.

There is an extensive body of work [17, 20, 14, 27, 6] on efficient random selection protocols in the full information model, which we could potentially take advantage of. The challenge, however, in using these protocols, is that in *all of these works, an additional assumption is made*: **reliable broadcast channels exist for free**. Namely, when an honest player receives a message that is 'broadcast', he is guaranteed that all other honest players received the same message even if it was sent by a faulty player. Thus, no part of the protocol needs to be dedicated to ambiguity. Indeed, both the correctness and efficiency (including the round complexity) of [17, 20, 14, 27, 4] hold only under the additional assumption that broadcast is an atomic unit-cost operation.

In our work, we do not assume that broadcast channels exist. In fact, one of our main results will be a protocols to achieve broadcast (i.e, Byzantine Agreement) in the full information model.

We focus on the case of a point-to-point synchronous network. Messages are sent at the end of a round and are delivered at the beginning of the next round. Delivery of messages is in the *rushing* model. Namely, a player may possibly see all messages sent in round i by other players, before he sends his own round i messages.

The fault model we address is a coalition of t faulty (corrupted) players, whose identity is decided by an adversary before the protocol begins. The adversary can be a *Byzantine t-adversary*, in which case he decides *which* messages will be sent by the t faulty players in every round, deviating from the protocol in the worst possible manner. The adversary can be a *fail-stop t-adversary*, in which case every faulty player follows the protocol, until the adversary instructs it to stop sending messages all together. This may happen anytime, including in the middle of a round, after a faulty player has sent messages to a subset of the players.

We consider a computationally unbounded adversary who makes his decisions based on the information about the state of all players (faulty and honest) including their coin tosses up to and including the current round, and the entire history of communications between them. Such adversary was called *intrusive* by Chor and Dwork [10]. An intrusive adversary was also assumed in [2, 9, 5].[1]

Main Results

- The Byzantine agreement problem is how n players, each of whom has a single bit input, can agree on a common output bit, such that if all non-faulty players start with the same input b, then the output is b as well. We show

  (1) An $O(\log n)$-round randomized Byzantine Agreement (BA) protocol in a synchronous full-information network tolerating $t < \frac{n}{3+\epsilon}$ faulty players (for any constant $\epsilon > 0$). This achieves *asymptotically optimal fault tolerance* as Pease, Shostak and Lamport [25] and Karlin and Yao [22] show that BA is not possible if $n \leq 3t$. Our protocol improves on the fault-tolerance of the protocol of Ben-Or, Pavlov and Vaikuntanathan [5] who show an $O(\log n)$ round Byzantine agreement for $t \leq \frac{n}{4+\epsilon}$.

  (2) An $O(1)$-round randomized BA protocol tolerating $O(\frac{n}{\log^{1.58} n})$ faulty players. The best round complexity known for this value of t was $O(\log n)$ [5].

- A fail-stop adversary models a benign fault whereas a Byzantine adversary models a much more severe fault. We show a compiler that takes any randomized protocol Π_{in} designed to tolerate a Fail Stop t-adversary, where the *source of randomness of all players in* Π_{in} *is an SV-source* [28],[2] into a protocol Π_{out} that tolerates a Byzantine $\mathsf{min}(t, \frac{n}{3})$-adversary. If the round-complexity of Π_{in} is r, that of Π_{out} is $O(r \log^* n)$. Previously, Hadzilacos, Neiger-Toueg, and Bracha [21, 24, 7] constructed such a compiler for deterministic protocols, and [7] raised as an open question, whether such a compiler exists for randomized protocols.

- The results above are derived via new leader election protocols in the full-information model that do not assuming broadcast channels. For $t \leq \frac{n}{3+\epsilon}$ faults, we achieve leader election in $O(\log n)$ rounds.

1.1 A New Tool – Audited Protocols: How to Remove Broadcast Assumption

Given any distributed protocol Π that possibly assumes broadcast channels, we would like to execute Π "as well as possible" when no reliable broadcast channels are given.

Of course, one could simulate any protocol assuming broadcast channels, by replacing each broadcast instruction of the protocol with the execution of a sub-protocol for implementing reliable broadcast.

Note that reliable broadcast is trivially solved given any protocol for BA, simply by setting the inputs of all non-faulty players in the Byzantine agreement protocol to be the message to be reliably broadcast. This naive approach, however, runs into trouble, as it may increase the round-complexity of the simulated protocol prohibitively. Moreover, even if one were to design a Byzantine agreement protocol with expected round-complexity $k = o(\log n)$, it would not merely imply an $O(k)$ factor slow-down in the number of

[1]We remark that even within the full-information setting, an intrusive adversary is especially powerful. Conceivably, one can get more efficient protocols in the full-information model by taking advantage of weaker adversaries, such as one who is computationally unbounded and can see all messages exchanged between non-faulty players but does *not* have access to their private inputs and coin tosses.

[2]Namely, given all coins tossed by all players thus far, the probability that the next coin is "heads" is bounded between $\frac{1}{2} - \gamma$ and $\frac{1}{2} + \gamma$ for some $\gamma > 0$.

rounds. As already observed by Chor and Rabin [12], the probability that all executions of a probabilistic protocol halt within the expected number of rounds proved for a single execution can be exponentially small in the number of executions.

Thus, we introduce a new protocol transformation called Audit. This tool applies to *any* protocol designed with the simplifying assumption that automatic broadcast is available and is at the *heart of all of our work*.

Audit allows us to take any protocol Π designed assuming reliable automatic broadcast, and produce a protocol $\mathsf{Audit}(C, \beta, \Pi)$ which *does not assume automatic broadcast*. The *auditing* committee C, is a subset of the n players and $\beta > 0$. We say that C is good when the fraction of corrupted players in C is smaller than β. When C is a good auditing committee the output distribution of non-faulty players in $\mathsf{Audit}(C, \beta, \Pi)$ will be as in Π. Whenever C is not a good committee, the output of the non-faulty players in $\mathsf{Audit}(C, \beta, \Pi)$ will be as in Π *except* that some non-faulty players may output $\perp$. The round-complexity of $\mathsf{Audit}(C, \beta, \Pi)$ is $|C|$ times that of Π.

Informally, the role of the committee C is to "audit" the execution of $\mathsf{Audit}(C, \beta, \Pi)$ and ensure that in *each round* all honest players get the same messages, thus simulating the reliable broadcast functionality. If the auditing committee contains a sufficient fraction of honest players this will be ensured. Otherwise the worst that may happen is that some honest player will receive a $\perp$ message instead of what was sent by honest players.

An interesting special case of 'Audit' used to derive the results in sections 5 and 6, is when the auditing committee consists of a single player. In this case, the round-complexity of the audited protocol is essentially the same as that of the original protocol. Recently, in independent work, Katz and Koo [23] introduced the notion of moderated Verifiable Secret Sharing (VSS). The idea of a moderator is very reminiscent of the idea of an auditing committee which consists of a single auditor. In contrast with our work, [23] obtain their results in the private channels model. We remark, that although our primary interest in this paper is the full information model, the Audit transformation introduced here will apply to the private channels model as well. Another interesting usage of 'Audit' is when the execution of a protocol is audited not by a single committee, but by a collection of committees. Namely, the same execution is in parallel audited by different committees (See Section 4 for details).

We proceed to outline the ideas behind our results.

1.2 $O(\log n)$-Round Byzantine Agreement in Full Information Network

Since its introduction in [25], the problem of Byzantine Agreement (BA) has been the source of enormous attention. The BA protocol of [25] had a round complexity of $t + 1$ rounds, which was shown to be optimal for deterministic protocols by Fischer and Lynch [16].

Researchers quickly resorted to randomization as one of the ways to overcome this limitation. Ben-Or, Rabin and Bracha [2, 26, 8] started this line of work, putting forth the idea of a *common coin* as the correct notion of randomization to achieve BA. In particular, Rabin [26] followed by Dwork, Shmoys and Stockmeyer [13] distilled the notion of a common coin and showed that if there is an r-round common-coin protocol, then there is an expected $O(r)$-round Byzantine Agreement protocol.

In the private channels model (and the computational model under intractability assumptions), Feldman and Micali [15] showed how to construct a common coin in in $O(1)$ rounds tolerating $t < \frac{n}{3}$ faulty players. Consequently, they achieved BA in expected $O(1)$ rounds.

Feige [14] and Russell and Zuckerman [27] construct a common-coin in the full-information model with a round-complexity of $\log^* n + O(1)$ rounds, but this bound is proved under the assumption that *reliable broadcast channels exist*.

The precise notion of a coin necessary to make the Rabin reduction go through is that of an (ϵ, δ)-common coin [13]. An (ϵ, δ)-common coin is a coin with bias δ which all players agree on with probability ϵ. Thus, the protocols of [14, 27] construct an $(1, \delta)$-common coin for some $\delta > 0$. In other words, at the end of the coin-flipping protocol, all the players output the same semi-random bit *with probability* 1. In this work, we will construct an (ϵ, δ)-common-coin protocol for some $\epsilon, \delta > 0$, *without assuming broadcast channels*. We then show,

Main Theorem 1. *For any constant $\epsilon > 0$, there exists a BA protocol BA_ϵ in a synchronous full-information network tolerating $t < (\frac{1}{3} - \epsilon)n$ Byzantine faults, and runs for expected $O(\frac{\log n}{\epsilon^2})$ rounds.*

Prior to our work, in the full information model, the best known BA protocol was due to Ben-Or, Pavlov, and Vaikuntanathan [5] who construct an $O(\log n)$-round protocol that tolerates $t < (\frac{1}{4} - \epsilon)n$ faults, with quasi-polynomial communication complexity.

Whereas the basic paradigm outlining the result of [5] was to use (in a complex fashion) the leader election protocol of Feige [14], the basic building block outlining our new protocol is the random selection protocol of Russell and Zuckerman (RZ) [27]. Informally, we will use the committees defined by the RZ protocol as auditors for an execution of the RZ protocol itself. This is possible, since in the RZ protocol, the committees are defined in advance. This idea is not (at least directly) applicable to Feige's protocol since it constructs the committees on-the-fly.

1.3 Transforming Fail-Stop Fault Resilient Protocols into Byzantine Faults Resilient Protocols

Designing distributed algorithms that tolerate Byzantine failures is a complex task. The task of the protocol-designer would be simpler, were there a *compiler* that takes as input a distributed protocol Π that tolerates *benign failures*, and outputs a robust distributed protocol Π' that tolerates the most *severe* failures.

The problem of designing a compiler that automatically converts any *deterministic* protocol that tolerates fail-stop faults to one that tolerates Byzantine faults was considered by Hadzilacos[21] and Neiger and Toueg [24]. They provide a procedure that converts any deterministic protocol Π_{in} that tolerate fail-stop faults into Π_{out} whose output in the presence of Byzantine faults, is identical to the output of Π_{in} in the presence of fail-stop faults. We remark that their transformation explicitly assumes that in Π_{in} the inputs of honest players is sent to all other players in the first round, however when the adversary is intrusive this restriction is unnecessary.

Bracha [7] explicitly raised the question, which we partially address here, whether such a transformation is possible for randomized protocols as well. The additional challenge over the deterministic case is that a fail-stop fault in a randomized protocol will flip coins fairly as prescribed by the protocol, but a Byzantine failure could potentially use any biased coins it likes (or none at all, in particular).

Our main result of this section is:

Main Theorem 2. [3] *Suppose there is an r_{cc}-round $(1, \gamma)$ common-coin protocol (possibly using reliable broadcast channels). Then there is a compiler that converts any r_Π-round protocol Π in which the randomness source is a γ-SV-source and which tolerates a fail-stop t-adversary, to a $r_{\Pi'}$-round protocol Π' that uses a uniformly random source and tolerates a Byzantine t'-adversary where $t' = \mathsf{min}(t, \frac{n}{3})$, and $r_{\Pi'} = O(r_{\mathsf{cc}} r_\Pi)$.*

[3]For a non-intrusive adversary, the theorem statement would hold for those protocols Π in which all players send their inputs in the first round, and in which all coins tossed by players are public-coin, i.e in each round the honest players send the outcome of their coins along with their messages. All random selection protocols we know of are of this type. Indeed, within this work, we only apply the above theorem to random selection protocols in full information model which have no initial inputs and which use public coins. It is an interesting question whether one can design better (with less rounds and/or better bias) random selection protocols in which the players use private-coins.

It is an interesting question whether the condition on in the above theorem on the coins of the players in the input protocol can be removed.

1.4 $O(1)$-Round BA Against $O(\frac{n}{\log^\beta n})$ Faults

Ben-Or et al [5] construct an expected $O(1)$-round Byzantine Agreement protocol that tolerates $O(\frac{n}{\log^2 n})$ Byzantine faults, using the *one-round* collective coin-flipping protocol of Ajtai and Linial [1]. This is the best fault-tolerance for which we know how to construct an expected $O(1)$-round BA protocol. In particular, the best known BA protocol that tolerates $t = \omega(\frac{n}{\log^2 n})$ faults has a round-complexity of $O(\log n)$ [5]. We show the following theorem.

Main Theorem 3. *There exists an $O(1)$-round BA protocol in a synchronous full-information network of n players tolerating $t = O(\frac{n}{\log^{1.58} n})$ malicious faults.*

This $O(1)$-round BA protocol works in two steps. First, we construct a new common-coin protocol in the full information model with broadcast channels with bias $O(\frac{1}{\log n})$ tolerating $t = O(\frac{n}{\log^{1.58} n})$ faults. Interestingly, one cannot use [14, 27] directly for this task, as they do not achieve such small bias even when the number of faults is small. Then, we use the compiler from the previous section, applied to a very simple one-round Byzantine Agreement protocol designed to tolerate fail-stop adversary due to Chor, Merritt and Shmoys [11]. It is an easy calculation to show that the [11] protocol works even when the randomness source is a γ-SV-source with $\gamma = O(\frac{1}{\log n})$. In fact, we prove a more general theorem than above (See Section 6 for details).

2 Technical Preliminaries

Notation. For a vector $\vec{x} = (x_1, x_2, \ldots, x_n)$ and a predicate $P(i)$ on indices i, $\vec{x}\big|_{i:P(i)}$ denotes the vector consisting of all x_i's such that $P(i)$ is true.

2.1 Formal Model of a Synchronous Distributed System

We will let n denote the number of players, and $\mathcal{P} = \{P_1, P_2, \ldots, P_n\}$ the set of players. The communication network consists of reliable communication channels between every pair of players. We *do not* assume the existence of built-in broadcast channels.

Protocols. A distributed protocol $\vec{\Pi}$ is specified by an n-tuple of interacting randomized programs $(\Pi_1, \Pi_2, \ldots, \Pi_n)$, where we think of Π_i as the program executed by player P_i. In each round, each program Π_i receives messages from the other programs, *tosses coins*, sends messages to the other programs and changes state.

Randomness. We model the randomness used by the components in a distributed protocol by a random source $\mathcal{R}$ that generates a sequence of bits according to some distribution. Whenever one of the programs Π_i requests a random bit ("tosses a coin"), the next bit from the source is returned. We consider two specific sources $\mathcal{R}$ in this paper. The first one, which is traditionally assumed by randomized distributed protocols, is a perfectly random source (that is, when $\mathcal{R}$ is just a sequence of unbiased and independent bits). The second one is a γ-SV-source [28]. A γ-SV-source (where $0 \leq \gamma \leq \frac{1}{2}$) is a sequence of bits $b_1 b_2 \ldots$ such that for any i, $\frac{1}{2} - \delta \leq \Pr[b_{i+1} = 0 \mid b_1, \ldots, b_i] \leq \frac{1}{2} + \delta$.

Adversaries. We consider static adversaries in this paper. A static adversary corrupts players *before* the protocol execution begins, as opposed to a dynamic (or adaptive) adversary which can corrupt players throughout the execution of the protocol.

An t-adversary corrupts at most t players, by replacing the program Π of each corrupted player with a program $\widetilde{\Pi}$. The adversaries we consider are of three types – Byzantine, omission and fail-stop. In the case of a *Byzantine (malicious)* t-adversary, $\widetilde{\Pi}$ is an arbitrary program. In the case of an omission t-adversary, $\widetilde{\Pi}$ is the same as Π except that in every round, $\widetilde{\Pi}$ can *omit to send or receive* an arbitrary subset of the outgoing messages. In the case of a fail-stop adversary, $\widetilde{\Pi}$ is the same as Π except that it can *halt*, possibly in the middle of a round and send an arbitrary subset of the messages it is supposed to send.

In each round, the adversary can decide what the corrupted players send in a round (resp. decide when the corrupted players halt) *after* seeing the messages sent by the honest players in *the same round*, in addition to the messages from the previous rounds. Such an adversary is called a *rushing* adversary. Note that the adversary can observe *all* the messages exchanged in the network, even the ones that honest players exchange amongst themselves. In other words, no communication is assumed to be secret and no computational restrictions are imposed on the adversary (this models the "full-information" aspect of the network). In addition, the adversary has access to the state of all the players, including their past (but not future) random coins (this stronger model has been called an "intrusive adversary")[4].

We let $\mathcal{O}_i(\vec{\Pi}, \vec{x}, \mathcal{R}, A)$ denote the output distribution of the player P_i in the execution of the protocol $\vec{\Pi}$ against the adversary A, when the input vector of the players is $\vec{x}$ and the randomness is taken from a source (with distribution) $\mathcal{R}$. We will let $\mathcal{O}(\vec{\Pi}, \vec{x}, \mathcal{R}, A)$ denote the joint distribution of the outputs of *all* the players.

Definition 1 (**Simulation of a Protocol Π by a Protocol Π'**)**.** *Protocol Π' with randomness source $\mathcal{R}'$ is said to* perfectly simulate Π *with randomness source $\mathcal{R}$ if, for every adversary A', there exists an adversary A such that for every input vector $\vec{x}$, the output distribution of Π' under the influence of A' is identical to the output distribution of Π under A. That is, $\forall A', \exists A$ such that $\mathcal{O}(\Pi', \vec{x}, \mathcal{R}', A') \equiv \mathcal{O}(\Pi, \vec{x}, \mathcal{R}, A)$. In such a case, we write $\Pi' \sim \Pi$.*

If the output distribution of Π' is identical to that of Π except that some players output $\perp$, we say that Π' $\perp$-simulates Π. We write $\Pi' \sim_\perp \Pi$.

2.2 Byzantine Agreement and Reliable Broadcast

In its most basic form, the problem of Byzantine Agreement [25] is as defined below.

Definition 2 (Byzantine Agreement)**.** *Let Π be a protocol among n players, in which each player P_i starts with an input bit b_i, and P_i outputs a bit c_i at the end of the protocol. Π is a Byzantine Agreement protocol, if the following conditions hold: (1)* Agreement: *For any two non-faulty players P_i and P_j, $c_i = c_j$. (2)* Validity: *If all the non-faulty players start with the same input bit b, then the output of every non-faulty player is b. (3)* Termination: *Protocol Π terminates with probability* 1.

Note that, if Π is randomized, the Agreement, Validity and Termination conditions are required to hold *with probability* 1 over the coin-tosses of the processors. The principal complexity measure of interest is the *expected* running time of the protocol.

We also define reliable broadcast, which is easily seen to be equivalent to Byzantine Agreement problem.

Definition 3 (Reliable Broadcast)**.** *Let Π be a protocol among n players, in which a designated player $D \in \mathcal{P}$ starts with input b and every player P_i outputs a bit c_i. Π is a reliable broadcast protocol, if the following conditions hold: (1)* Agreement: *For any two non-faulty players P_i and P_j, $c_i = c_j$. (2)* Validity: *If D is honest, then the output of every non-faulty player is b. (3)* Termination: *Protocol Π terminates with probability* 1.

[4]Since we show positive results in this paper, it is only better to work against a stronger adversary

2.3 Graded Broadcast

An appropriate formalization of a semi-reliable broadcast channel is the following notion of graded broadcast (introduced and implemented by Feldman and Micali [15]).

A protocol $\mathcal{P}$ is said to be achieve graded broadcast if, at the beginning of the protocol, one of the players (the dealer D) holds a value v, and at the end of the protocol, every player P_i outputs a pair $(\mathsf{v}_i, \mathsf{conf}_i)$ such that the following properties hold: $(\forall i, \mathsf{conf}_i \in \{0, 1, 2\})$ **(1)** If the dealer D is honest, then $\mathsf{v}_i = v$ and $\mathsf{conf}_i = 2$ for every honest player P_i. **(2)** For any two honest players P_i and P_j, $|\mathsf{conf}_i - \mathsf{conf}_j| \leq 1$. **(3)** For any two honest players P_i and P_j, if $\mathsf{conf}_i > 0$ and $\mathsf{conf}_j > 0$, then $\mathsf{v}_i = \mathsf{v}_j$. The following lemma is proven in [15].

Lemma 1 (Feldman-Micali [15])**.** *There exists a protocol Π_{FM} among n players which achieves graded broadcast as long as $t < \frac{n}{3}$ players are corrupted by a Byzantine adversary. Π_{FM} runs in $O(1)$ rounds.*

2.4 Common Coin

The precise notion of a "common-coin" necessary for the Ben-Or and Rabin reduction from Byzantine Agreement to coin-flipping was distilled by Dwork, Shmoys and Stockmeyer [13] and is as given below.

Definition 4 (Common-coin)**.** *A protocol Π is said to be a (ϵ, δ)-common-coin protocol if, at the end of the protocol, each player P_i outputs a bit b_i, and there exist constants $\epsilon, \delta > 0$ such that the following hold:*
(1) Commonality*: With probability at least ϵ,* all *the players output the same bit b. That is,* $\Pr[\,\exists b \text{ such that } \forall i, b_i = b] \geq \epsilon$*.*
(2) Randomness*: Given that all the players output the same bit b, the bias of b is at most δ. That is,* $\frac{1}{2} - \delta \leq \Pr[\, b = 0 \mid \exists b \text{ such that } \forall i, b_i = b] \leq \frac{1}{2} + \delta$.

2.5 Combinatorial Tools

Construction of Committees. Let $\mathcal{C} \subseteq \{C_i \mid C_i \subseteq [n] \text{ such that } |C_i| = n'\}$ be a collection of m subsets of the n players, each subset of size n'. Each such subset is referred to as a *committee*. Thus, $\mathcal{C}$ is a collection of m committees.

Assuming that βn of the n players are corrupt, a natural property to desire on the part of the collection of committees is that, *very few* of the committees have much more than $\beta n'$ bad players. More formally, a committee C is said to be ϵ-bad with respect to a set $B \subseteq [n]$ if $|B \cap C| > (1 + \epsilon)\beta n'$. The collection of committees $\mathcal{C}$ is good if, *for any* $B \subseteq [n]$, the number of bad committees in $\mathcal{C}$ with respect to B is at most $3n$.

The following lemma shows that such committees with appropriate parameters exist.

Lemma 2. *Let n denote the number of players and t denote the number of bad players. Then, there exists a good collection of m committees $\mathcal{C}$ such that $m = n^2$ and $n' = O(\frac{\log n}{\epsilon^2})$.*

Hitting Sets. To define the notion of a hitting set for combinatorial rectangles, we first need a few definitions.

Fix an $a > 0$. A *combinatorial rectangle* $R \subseteq [a]^n$ is *defined to be* $R = R_1 \times R_2 \times \ldots \times R_n$ where each $R_i \subseteq [a]$, and $|R_i| = a - 1$. The volume of R in $[a]^n$ is $\mathsf{vol}(R) = \frac{1}{a^n} \prod_{i=1}^{n} |R_i|$.

A set $D \subseteq [a]^n$ is called an (a, n, m)-hitting set if (1) $|D| = m$, and (2) for every combinatorial rectangle $R \subseteq [a]^n$, $|D \cap R| > 0$. The following lemma shows that hitting sets exist for a certain choice of parameters.

Lemma 3. *There exists an (a, n, m)-hitting set D for every n, $m = n^2$ and $a = \frac{2n}{\log n}$.*

2.6 Other Tools

Lemma 4 ([4]). *There exists a one-round $(1, \frac{t}{n^{0.63}})$-common-coin protocol among n players tolerating a Byzantine t-adversary.*

Lemma 5 ([9]). *There exists a randomized BA protocol in the synchronous full-information model that tolerates a Byzantine t-adversary for any $t < \frac{n}{3}$ and runs in expected $O(\frac{t}{\log n})$ rounds.*

3 Audited Protocols

Given any distributed protocol Π that possibly assumes broadcast channels, we would like execute Π "as well as possible" when no reliable broadcast channels are given. For any distributed protocol Π and for any player P (the "auditor"), we define an "audited" protocol $\Pi' = \mathsf{Audit}(P, \Pi)$. Informally, $\mathsf{Audit}(P, \Pi)$ provides the following guarantees: If the auditor P is honest, Π' works exactly like Π. In particular, the outputs of the honest players in Π' are distributed exactly as they are in Π. Even if the auditor is dishonest, he can do only minimal damage – the worst he can do is set the outputs of some of the players to $\perp$. The players that do get a legal output (an output different from $\perp$) get one that is distributed according to Π.

More generally, we could have a set (a "committee") of players C (rather than a single player P) be the auditor. We define Π' to be an $\mathsf{Audit}(C, \beta, \Pi)$ if, Π' behaves exactly like Π whenever C is a good committee – that is, when the number of bad players in C is smaller than β. Even when C is bad (that is, the fraction of bad players in C is larger than β), the worst that C can do is to set the outputs of some of the honest players to $\perp$; the honest players that do get some legal output get a correctly distributed one. More formally,

Definition 5 ($\mathsf{Audit}(C, \beta, \Pi)$). *Define a committee $C \subseteq \mathcal{P}$ to be good if the number of bad players in C is at most $\beta|C|$. Given any protocol Π among n players, and a designated committee $C \subseteq \mathcal{P}$, a protocol $\Pi' = \mathsf{Audit}(C, \beta, \Pi)$ is called a (C, β)-audited Π if,*

- *When C is good, the output vector of the players in Π' is identically distributed as in Π. Thus, $\Pi \sim \Pi'$.*
- *Even when C is bad, the honest players get a correctly distributed output, or $\perp$. Thus, $\Pi' \sim_{\perp} \Pi$.*

Note that auditing by a single player is a special case of auditing by committees. In particular, $\mathsf{Audit}(P, \Pi)$ is exactly $\mathsf{Audit}(\{P\}, 0, \Pi)$.

Transforming Any Protocol into an Audited Protocol

We provide a transformation that converts any protocol Π that assumes broadcast to a (C, β)-audited Π, as defined above (See Table 1). We get a P-audited Π as a corollary of this transformation.

Theorem 6. *There exists a transformation* Audit *that takes any distributed protocol Π, the description of a committee C and a number $0 \leq \beta < \frac{1}{3}$, and outputs a protocol $\Pi' = \mathsf{Audit}(C, \beta, \Pi)$ such that Π' is a (C, β)-audited Π (as in Definition 5) If the fault-tolerance of Π is t, that of Π' is $\mathsf{min}(t, \frac{n}{3})$. If the round-complexity of Π is r_{Π}, the round-complexity of Π' is at most $|C| r_{\Pi}$.*

Proof. (Sketch.) We will show that the protocol $\Pi' = \mathsf{Audit}(C, \beta, \Pi)$ (given in Table 1) satisfies Definition 4. The proof will proceed in two parts: first, we show that Π' is an audited version of Π, whenever $|C| = 1$. That is, when the auditor is a single player. Secondly, we will show that a committee "behaves like" a player, in a sense to be made precise below.

First of all, assume that $|C| = 1$ (Notice that this considerably simplifies Steps 2(b) and 2(c)). If all the honest players set $\mathsf{fail}_j = 1$ at the end of Π', then all of them output $\perp$, and there is nothing to prove. Our goal will be to show that if some honest player P_j sets $\mathsf{fail}_j = 0$ (that is, his output is not $\perp$) then for every

Input: Protocol Π. **Output:** Protocol $\Pi' = \mathsf{Audit}(C, \beta, \Pi)$
Each Round of Protocol Π is simulated as follows. **(1)** If Π instructs P_i to send a message to P_j, Π' instructs the same. **(2)** If Π instructs P_i to broadcast a message m to all the players, the following subroutine is invoked. (a) **(Player P_i)** Gradecast m to all the players. (b) **(The Auditor C)** Let $C = \{Q_1, Q_2, \ldots, Q_{\lvert C \rvert}\}$. Let m'_k denote the message that Q_k received as a result of P_i's gradecast, in Step 2(a). The players in C run a deterministic Byzantine Agreement among themselves, with Q_k's input to the BA being m'_k. Each player $Q_k \in C$ gradecasts the output m''_k of the BA to all the players. (c) **(Every player P_j)** Receive $\lvert C \rvert$ pairs (m''_k, conf_k) from each player $Q_k \in C$. If there are *more than* $\frac{\lvert C \rvert}{2}$ pairs of the form $(\mu, 2)$, then set $m_C = \mu$ and $\mathsf{conf}_C = 2$. Else, if there are *more than* $\frac{\lvert C \rvert}{2}$ pairs of the form $(\mu, \geq 1)$, then set $m_C = \mu$, and $\mathsf{conf}_C = 1$. Else, set $m_C = \perp$ and $\mathsf{conf}_C = 0$. (d) **(Every player P_j)** Let (m_i, conf_i) and (m_C, conf_C) denote the outputs of P_j from the gradecast in step (a) and the computation in step (c), respectively. P_j will take m_C as the message broadcast by P_i in the underlying execution of Π. (e) **(Every player P_j)** Set a bit $\mathsf{fail} = 1$ if either (1) $m_C \neq m_i$ and $\mathsf{conf}_i = 2$, or (2) $\mathsf{conf}_C \neq 2$.
Finally, each player P_i gets an output O_i from the underlying execution of Π. P_i outputs O_i if $\mathsf{fail} = 0$, and $\perp$ otherwise.

Table 1: The transformation Audit

time a broadcast instruction in the protocol is simulated by Step (2) of Audit, all the honest players in fact receive the same message (and thus, the functionality of broadcast is achieved).

Suppose some honest player P_j sets $\mathsf{fail}_j = 0$. This means that, in all steps of the simulation (that is, every time a player P_i broadcasts a message m), in the view of P_j, both (1) $\mathsf{conf}_C = 2$, **and** (2) $m_C = m_i$ or $\mathsf{conf}_i \neq 2$. This follows from Step $2(e)$ of Audit. Since $\mathsf{conf}_C = 2$, every other honest player receives the same message from the committee C, with confidence at least 1. Thus, it follows that all the players receive the same value for the broadcast of player P_i. Moreover, if P_i is good, then $\mathsf{conf}_i = 2$. Then, by condition (ii) above, $m_C = m_i$. This means that, if P_i is good, all players accept the message that P_i "broadcast" (regardless of whether the committee is good or not). Thus, we showed that every player gets the same message as a result of P_i's broadcast *and* moreover if P_i is good, they accept the message P_i sent. This simulates the functionality of reliable broadcast perfectly, in each round.

To finish the proof, we show that a committee "behaves like" a player in the following sense: Suppose a committee gradecasts a message m (i.e, all the honest players in the committee gradecast m), and two honest players P_i and P_j receive the committee's gradecast with confidences (m_i, conf_i) and (m_j, conf_j) (as in Step 2(c) of the transformation). Then, if the committee is good (i.e, has more than $\frac{1}{3}$ fraction of honest players), $m_i = m_j = m$ and $\mathsf{conf}_i = \mathsf{conf}_j = 2$. Even if the committee is bad, $\lvert \mathsf{conf}_i - \mathsf{conf}_j \rvert \leq 1$ and if conf_i and $\mathsf{conf}_j \geq 1$, then $m_i = m_j$. This follows by inspection of Steps 2(b) and 2(c), a formal proof is omitted. The claim about the fault-tolerance of $\mathsf{Audit}(C, \beta, \Pi)$ and round complexity are easy to check and are omitted. $\square$

4 $O(\log n)$-round Byzantine Agreement

Theorem 7 (Main Theorem 1, Restated). *For any constant $\epsilon > 0$, there exists a protocol BA_ϵ that achieves Byzantine Agreement in a synchronous full-information network of n players tolerating $t < (\frac{1}{3} - \epsilon)n$ faults. The round complexity of BA_ϵ is $O(\frac{\log n}{\epsilon^2})$.*

Our BA protocol uses as a key tool the Audit transformation from Section 3, applied to the committee-selection protocol of Russell and Zuckerman [27].

4.1 Committee Selection Protocol

The goal of a committee-selection protocol is to select, from among n players, a set of committees, each consisting of n' players, such that with probability at least $1 - \frac{1}{n}$, *all the selected committees are good.*

Russell and Zuckerman use the tools developed in Subsection 2.5 (committee construction and hitting sets) to build a committee selection protocol Π_{RZ}. In particular, they prove the following theorem.

Theorem 8 (Russell-Zuckerman [27]). *Let n be the number of players and t be the number of bad players. Then, there exists a protocol Π_{RZ} which outputs a set of $k \geq 1$ committees C_i, where each committee has size $\frac{n \log n}{t}$, such that with probability at least $1 - \frac{1}{n}$,* all *the committees that Π_{RZ} outputs are good.*

Proof Sketch: The protocol Π_{RZ} proceeds as follows. The proof of correctness is as in [27] and is omitted for lack of space. The public setup for the protocol consists of (1) a collection of committees $\mathcal{C} = \{C_1, C_2, \ldots, C_m\}$, (2) A hitting set $D \subseteq [a]^n$ with $|D| = m$ and (3) A bijection $h : \mathcal{C} \mapsto D$.

(1) Each Player P_j chooses $r_j \in [a]$, and *broadcasts* r_j to all the players.
(2) Locally compute and output a vector $\vec{e} = \langle e_1, e_2, \ldots, e_m \rangle$ where $e_i = 1$ if and only if for some $1 \leq j \leq n$, $h(C)|_j = r_j$. Committee C_i is said to be eliminated if $e_i = 1$. $\square$

4.2 Overview of the Proof of Theorem 7

The classical paradigm for designing randomized BA protocols, pioneered by Ben-Or and Rabin [2, 26], reduces the problem of BA to the problem of constructing a common-coin. A common-coin is a bit b which has a constant bias, and is seen by all the players with a constant probability. Rabin shows that if there is an r-round common-coin protocol, then there is an expected $O(r)$-round Byzantine Agreement protocol. For the precise definition of a "common-coin protocol", see Subsection 2.4.

Suppose we could select a "good committee" (i.e, a committee with more than $2/3$ fraction of honest players) of size $O(\log n)$ *without using broadcast channels*, then we will immediately get an $O(\log n)$-round Byzantine Agreement protocol by the following strategy: We first choose the committee, and then run an $O(\log^* n)$-round *coin-flipping protocol* within the selected committee. These coin-flipping protocols themselves require reliable broadcast channels among the players in the committee. However, since the number of players in the committee is small, we can simulate the reliable broadcast channels by the Chor-Coan protocol (Lemma 5), resulting in a $o(\log n)$ rounds coin-flipping. After the players in the committee agree on a coin, they *will send* the value of the coin to all the players. Each player, in turn, computes the majority of all the coins he receives from the committee members. Since the committee is good, the majority will indeed be the coin-flip of the honest players.

It remains to specify how we could select a good committee of size $O(\log n)$ without broadcast channels. The committee-selection protocol Π_{RZ}, described above, assumes the existence of reliable broadcast channels, over which the players can announce their choice of which committees to eliminate. Take $\mathcal{C}$ to be the set of m committees defined before the beginning of the committee sampling protocol Π_{RZ}. We construct an *audited* committee-selection protocol, where *the same execution* of Π_{RZ} is audited by all the committees $C \in \mathcal{C}$. Thus, a committee acts both as a committee in Π_{RZ} as well as an auditor for Π_{RZ}. As opposed to auditing by a single committee, this structure will allow us to eliminate the use of broadcast channels in Π_{RZ}. For details, see the formal proof below.

4.3 Auditing by a Set of Committees

We make precise the notion of a set of committees auditing the same execution of a protocol (alluded to in the overview above). We define $\mathsf{Audit}(\mathcal{C}, \frac{1}{3}, \Pi)$ (refer to Table 1) to be the protocol obtained by applying the Audit transformation to protocol Π, except for the following changes: Steps 1 and 2(a) of the Audit transformation remain exactly the same. Steps 2(b)–2(e) are executed by each committee $C \in \mathcal{C}$ separately. The output of each player is a vector of m values $\vec{O} = \langle O_1, O_2, \ldots, O_m \rangle$, corresponding to the m auditing committees.

In the following, we focus on $\mathsf{Audit}(\mathcal{C}, \frac{1}{3}, \Pi)$ for a specific, simple protocol Π, namely that of a player Q broadcasting a message μ to all the players. Call this functionality $\mathsf{BCast}(Q, \mu)$. We show the following lemma about the output of the players in $\mathsf{Audit}\big[\mathcal{C}, \frac{1}{3}, \mathsf{BCast}(Q, \mu)\big]$.

Lemma 9. *Let P and P' be any two honest players. Let the output of P in $\mathsf{Audit}\big[\mathcal{C}, \frac{1}{3}, \mathsf{Broadcast}(Q, \mu)\big]$ be the vector $\langle O_1, O_2, \ldots, O_m \rangle$ and that of P' be $\langle O'_1, O'_2, \ldots, O'_m \rangle$.*
(1) *If C_i is a good auditor, then $O_i = O'_i$.*
(2) *If the broadcaster Q is honest, then for every good auditor C_i, $O_i = \mu$. Even if Q is dishonest, there is a* unique *message μ' such that for every good auditor C_i, $O_i \in \{\mu', \perp\}$.*

Proof Sketch: Part (1) follows from Theorem 6 and the fact that C_i is a good auditor. Part (2) essentially follows by the property of gradecast. Observe that after Q gradecasts his message, each player gets either the same message μ, or $\perp$. Thus, each good auditor will send either μ or $\perp$ in Step 2(b) of the Audit transformation. If Q is honest, each player gets μ as a result of Q's gradecast, and thus each good auditor will send μ. $\square$

4.4 Formal Proof of Theorem 7.

For the full description of the protocol, see Table 2. By Lemma 10 below, Π_{select} selects a good committee with probability $1 - \frac{1}{n}$ and all the honest players know the identity of the chosen committee. By the argument in the overview, once we have a good committee, Byzantine Agreement follows (by running a coin-flipping protocol within the committee, and using Rabin's reduction from BA to coin-flipping). The claims about fault-tolerance and round-complexity follow from the corresponding claims in Lemma 10. $\square$

Lemma 10 (Committee-selection without Broadcast Channels)**.** *For every $\epsilon > 0$, there is an $O(1)$-round n-player protocol Π_{select} to select a committee $C \subseteq [n]$, consisting of $\frac{\log n}{\epsilon^2}$ players, such that with probability $1 - \frac{1}{n}$, all honest players output the same committee C. Π_{select} does not assume reliable broadcast channels.*

Proof Sketch: The protocol Π_{select} is given in Table 2. Lemma 12 shows that all the honest players *agree* on the list of committees that have not been eliminated. Lemma 11 shows that at the end of Π_{select}, all the bad committees are eliminated and Lemma 13 shows that at least one committee is not eliminated. Thus the committee C_i output at the end of Π_{select} is good, and all the honest players agree on which committee is chosen. $\square$

First, we will show that at the end of Π_{select}, all the bad committees are eliminated.

PROTOCOL Π_{select}
Public Setup. As in Theorem 8 1. **(Each Player** P_j**)** Choose r_j randomly from $[a]$. Run $\mathsf{Audit}\big[\mathcal{C}, \frac{1}{3}, \mathsf{BCast}(P_j, r_j)\big]$. The output of player P from $\mathsf{Audit}\big[\mathcal{C}, \frac{1}{3}, \mathsf{BCast}(P_j, r_j)\big]$ is a vector $\langle r^P_{j1}, r^P_{j2}, \ldots, r^P_{jm} \rangle$. 2. **(Each Player** P**, locally)** Construct an $m \times n$ matrix $R^P = [r^P_{ij}]$, where each row corresponds to an auditor and each column corresponds to a player. The j^{th} column of R^P is the output of $\mathsf{Audit}\big[\mathcal{C}, \frac{1}{3}, \mathsf{BCast}(P_j, r_j)\big]$ from Step (1). 3. **(Each Player** P**, locally)** From R^P, construct an $m \times m$ matrix $D^P = [d^P_{ik}]$, where $d^P_{ik} = 1$ iff there exists a j such that $h(C_k)\|_j = r^P_{ij}$. 4. **(Each Committee** C_i**)** Run a deterministic BA among the players in C_i, where a player P's input is the matrix D^P from Step (3). Let the resulting matrix (after the BA) of player P be denoted $\tilde{D}^P$. If the i^{th} column of $\tilde{D}^P$ contains more than n 1's, then P sends a message, "Eliminate C_i" to all the players. 5. **(Each Player** P**, locally)** Construct a vector $\vec{e}^P$ such that $\vec{e}^P_i = 1$ if and only if either, (1) the i^{th} column of D^P contains more than $4n$ 1's, or (2) P received "Disqualify C_i" messages from more than half the members of C_i. C_i is said to be *eliminated* if $\vec{e}^P_i = 1$. Output the lexicographically smallest C_i that is *not eliminated*.

Table 2: Committee-Selection Without Broadcast

Lemma 11. *With probability* $1 - \frac{1}{n}$ *over the coin-tosses of the honest players, all bad committees are eliminated after Step (5) of* Π_{select}. *More precisely, for every bad committee* C_k *and every honest player* P, $\vec{e}^P_k = 1$.

Proof. With probability at least $1 - \frac{1}{n}$, every bad committee is eliminated at the end of Π_{RZ} (See Theorem 8). That is, for every bad committee C_k, there is an honest player P_j (who broadcasts r_j) such that $h(C_k)|_j = r_j$. Because of the property of $\mathsf{Audit}\big[\mathcal{C}, \frac{1}{3}, \mathsf{BCast}(P_j, r_j)\big]$ (Lemma 9, Part (2)), the j^{th} column of R_P contains $n^2 - 3n$ r_j's (one for each good auditing committee). This means that, by the construction of D_P, the k^{th} column of D_P contains $n^2 - 3n$ 1's. Since $n^2 - 3n > 4n + 1$, C_k is eliminated by player P. Since the above argument holds for every honest player P, $\vec{e}_k = 1$ for every bad committee C_k and honest P. $\square$

The next lemma shows that if an honest player P thinks that C_j has been eliminated, then every other honest player Q will think that C_j has been eliminated too.

Lemma 12. *Let* $\vec{e}^P$ *and* $\vec{e}^Q$ *be the vectors constructed by honest players* P *and* Q *in Step (5) of* Π_{select}. *Then,* $\vec{e}^P = \vec{e}^Q$.

Proof Sketch: It is sufficient to show that for every i such that $\vec{e}^P_i = 1$, $\vec{e}^Q_i = 1$ too. First of all, for every bad committee C_i, $\vec{e}^P_i = \vec{e}^Q_i = 1$ (be Lemma 11). In the rest of the proof, we let C_i be a good committee. If $\vec{e}^P_i$ is 1, then it is because of one of the following reasons (See Step (5) of Table 2).

(1) P received "Disqualify C_i" messages from more than half the members of C_i: In this case, since C_i runs a BA protocol before sending the "Disqualify C_i" messages and since C_i is good, all honest players in C_i send "Disqualify C_i" messages too. This means that every other honest player Q receives a "Disqualify C_i" message from more than half the members of C_i. This results in $\vec{e}^Q_i = 1$.

(2) The i^{th} column of D^P contains more than $4n$ 1's: Consider the k^{th} row in R^P corresponding to a good auditor C_k. By Part (1) of Lemma 9, the k^{th} row of R^Q will be the same. Thus, the k^{th} rows of D^P and D^Q are the same too, by construction. Of the $4n$ 1's in the i^{th} column of D^P, at most $3n$ belong to bad auditors. Thus, there are more than n 1's in the i^{th} column of D^Q. Since this is true for every honest player Q, in Step 4 of Π_{select}, the matrix $\tilde{D}^Q$ will contain more than n 1's. Thus all the honest players in C_i will send a "Disqualify C_i" message to all the players. This makes $\vec{e}_i^Q = 1$. ☐

Finally, we show that at least one committee is not eliminated.

Lemma 13. *There exists an i such that $\vec{e}_i^P = 0$ for every honest player P.*

Proof Sketch: Fix the collection of messages r_j sent by all the players (including the faulty ones). By Theorem 8, there is at least one committee C_i such that $h(C_i)|_j \neq r_j$ for any j. This means that the i^{th} column of D^P has at most n 1's, for any honest player P. P will set $\vec{e}_i^P = 0$, and thus, C_i is not eliminated. ☐

Corollary 14. *For every $\epsilon > 0$, there exists a leader election protocol among n players that runs in $O(\frac{\log n}{\epsilon^2})$ rounds tolerating $t < \frac{n}{3+\epsilon}$ faulty players and does not assume reliable broadcast channels.*

5 Transformation from Fail-stop to Byzantine

In this section, we will construct a compiler that takes as input any full-information protocol $\Pi_{\mathsf{fs}}^{\gamma\text{-SV}}$ that tolerates a fail-stop t-adversary and uses a γ-SV-source as the source of randomness, and outputs a protocol Π_{Byz} that realizes the same functionality as $\Pi_{\mathsf{fs}}^{\gamma\text{-SV}}$, tolerates a Byzantine $\min(t, \frac{n}{3})$-adversary and uses a uniformly random source.

To understand our compiler, we first focus on the difference between a fail-stop adversary and a Byzantine one. Informally, a Byzantine fault is more malicious than a fail-stop fault in two aspects: (1) a Byzantine player can use arbitrary coins as the source of randomness, whereas a fail-stop player uses the prescribed source of randomness, (2) a Byzantine player can send arbitrary messages to players in every round, whereas a fail-stop player follows the prescribed protocol, and can only halt (possibly in the middle of a round).

Thus, to transform a protocol tolerating fail-stop faults to one tolerating Byzantine faults, we should (1) force the Byzantine player to flip "good coins" and (2) restrict her to follow the prescribed protocol. The compiler we construct reflects the above intuition.

Overview of the Compiler. The compiler proceeds in three steps.
Step 1. Convert the input protocol $\Pi_{\mathsf{fs}}^{\gamma\text{-SV}}$ to a protocol $\Pi_{\mathsf{omit}}^{\gamma\text{-SV}}$ that tolerates *omission* faults, where in both $\Pi_{\mathsf{fs}}^{\gamma\text{-SV}}$ and $\Pi_{\mathsf{omit}}^{\gamma\text{-SV}}$, *all* the players use a γ-SV source as the source of randomness.
Step 2. Convert $\Pi_{\mathsf{omit}}^{\gamma\text{-SV}}$ to a protocol Π_{omit}, which tolerates *omission* faults, and uses a uniformly random source, but allows the faulty players to flip their own coins. Recall that $\Pi_{\mathsf{omit}}^{\gamma\text{-SV}}$ worked only against adversaries that used the prescribed source of randomness (which is an SV-source).
Step 3. Convert Π_{omit} to Π_{Byz} that tolerates Byzantine faults.

Steps 1 and 3 exactly follow the compiler constructed by Neiger and Toueg [24]. The proof of correctness, however, is slightly more complex than [24], since we deal with randomized protocols.

The main novelty is in Step 2, which we proceed to describe. Observe that since the adversary is intrusive, we can without loss of generality assume that all the players in Π_{omit} send their coin-tosses in every round. Step 2 will proceed by running a $(1, \gamma)$-common-coin subroutine Π_{cc} for player P, whenever the underlying protocol $\Pi_{\mathsf{omit}}^{\gamma\text{-SV}}$ asks P to sample a coin from the γ-SV random source. In other words, we force all the players in Π_{omit} to use the outcome of the common-coin subroutine as the source of randomness. If the common-coin protocol Π_{cc} assumes reliable broadcast channels, we will run a P-audited Π_{cc} (For details, see Lemma 15).

Lemma 15. *Assume that Π_{cc} is an a r_{cc}-round $(1,\gamma)$-common-coin protocol tolerating an omission t-adversary. Then, there are compilers that convert:*
(1) *Any r-round protocol $\Pi_{\mathsf{fs}}^{\gamma\text{-}\mathsf{SV}}$ that works against a* fail-stop *t-adversary, to a protocol $\Pi_{\mathsf{omit}}^{\gamma\text{-}\mathsf{SV}}$ that works against an omission $\min(t, \frac{n}{2})$-adversary and runs in $O(r)$ rounds.*
(2) *Any r-round protocol $\Pi_{\mathsf{omit}}^{\gamma\text{-}\mathsf{SV}}$ to a protocol Π_{omit}. If the fault-tolerance of $\Pi_{\mathsf{omit}}^{\gamma\text{-}\mathsf{SV}}$ is t, that of Π_{omit} is $\mathsf{min}(t, \frac{n}{2})$. The round-complexity of Π_{omit} is $O(r_{\mathsf{cc}}r)$.*
(3) *Any r-round protocol Π_{omit} that works against an omission t-adversary to a protocol Π_{Byz} that works against a Byzantine $\min(t, \frac{n}{3})$-adversary and runs in $O(r)$ rounds.*

Proof Sketch: We omit the proofs for parts (1) and (3). We proceed to describe the compiler for part (2). Observe that since the adversary is intrusive, we can without loss of generality assume that in every round, each player send its coin-tosses to every other player.

Π_{omit} works exactly like $\Pi_{\mathsf{omit}}^{\gamma\text{-}\mathsf{SV}}$, except for the following: *Whenever a player P_i in $\Pi_{\mathsf{omit}}^{\gamma\text{-}\mathsf{SV}}$ is instructed to toss a coin, all the players together execute a P_i-audited $(1,\gamma)$-common-coin protocol Π_{cc}. Let the output (the coin) be b_i.*[5] *P_i uses b_i as the coin in the underlying protocol.*

When an honest player P_i is instructed to toss a coin in $\Pi_{\mathsf{omit}}^{\gamma\text{-}\mathsf{SV}}$, all the players in Π_{omit} see the outcome of the P_i-audited Π_{cc}, since the auditor P_i is honest. When a faulty player P_i is instructed to toss a coin, the outcome of the P_i-audited Π_{cc} is a coin with bias at most γ, however a subset S of players do not see the outcome of the coin-toss (since the auditor P_i is dishonest). This corresponds to P_i flipping a coin with bias γ and sending it only to the players in $\mathcal{P} \setminus S$ in the underlying protocol $\Pi_{\mathsf{omit}}^{\gamma\text{-}\mathsf{SV}}$. Thus, Π_{omit} constructed as above simulates the functionality of $\Pi_{\mathsf{omit}}^{\gamma\text{-}\mathsf{SV}}$. □

Proof of Main Theorem 2. Follows by putting together the compilers in Lemma 15.

6 $O(1)$-Round BA Against $O(\frac{n}{\log^{\beta} n})$ Faults

In this section, we construct an expected $O(1)$-*round* BA protocol that tolerates a Byzantine t-adversary for any $t = O(\frac{n}{\log^{1.58} n})$. In fact, we prove the following general theorem, which will allow us to construct a BA protocol tolerating $\frac{n}{\log^{1+\epsilon} n}$ faults, if a one-round coin-flipping protocol with the appropriate resilience is given.

Theorem 16. *Suppose there exists a* one-round *collective coin-flipping protocol Π_{coin} such that for any Byzantine t-adversary A, Π_{coin} generates a coin with bias at most $\frac{t}{n^{\alpha}}$. Then, there exists a BA protocol in a synchronous full-information network of n players tolerating $t = O(\frac{n}{\log^{\beta} n})$ Byzantine faults, for any $\beta > \frac{1}{\alpha}$. The protocol runs in expected $O(1)$ rounds.*

Proof. (of Theorem 16) By Main Theorem 2, it suffices to construct (1) a BA protocol against a fail-stop t-adversary that uses a γ-SV source as the source of randomness (for some $\gamma > 0$) *and* (2) a method of generating the γ-SV-source (that is, by running a collective coin-flipping protocol that outputs a coin with bias at most γ). (1) follows from Lemma 18 and (2) follows from Lemma 17. □

The proof of Main Theorem 3 is a simple corollary.

Proof of Main Theorem 3. Use the coin-flipping protocol from Lemma 4 as Π_{coin} in Theorem 16.

[5] Note that, if Π_{cc} does not assume physical broadcast channels, a P_i-audited Π_{cc} is Π_{cc} itself.

A Coin-flipping Protocol with Bias $O(\frac{1}{\log n})$. The coin-flipping protocols of Feige [14] and Russell-Zuckerman [27] do not guarantee that the bias of the coin generated is as small as $O(\frac{1}{\log n})$ (*even when* the number of faults is small). Thus, we construct a new collective coin-flipping protocol (assuming broadcast channels) that generates a coin with bias $\frac{1}{2\log n}$, when the number of faults $t = O(\frac{n}{\log n})$. More precisely,

Lemma 17. *Suppose there exists a* one-round *collective coin-flipping protocol* Π_{coin} *such that for any Byzantine t-adversary,* Π_{coin} *generates a coin with bias* $\frac{t}{n^{\alpha}}$. *Let* $\gamma = \frac{1}{2\log n}$. *Then, there exists a* $(1,\gamma)$*-common-coin protocol* Π_{cc} *against* $t < \frac{n}{\log^{\beta} n}$ *faults, for any* $\beta > \frac{1}{\alpha}$ *(*Π_{cc} *uses reliable broadcast channels). The round-complexity of* Π_{cc} *is* $O(1)$.

Proof. Π_{cc} selects a committee (much like the Russel-Zuckerman committee selection), but it does so in three rounds. In the *first* round, select a committee C_1 of size $\log^{1+\beta} n$ by running the Russell-Zuckerman committee-selection Π_{RZ} among n players (see Theorem 8). In the second round, elect a committee C_2 of size $\log^{\beta} n \log\log n$ by running the Russell-Zuckerman committee-selection Π_{RZ} among the players in C_1. In the third round, run the one-round protocol Π_{coin} among the players in C_2. By Theorem 8, the probability that C_2 is a bad committee is at most $\frac{1}{n} + \frac{1}{\log^{2+\beta} n} = o(\frac{1}{\log n})$. If C_2 is good, then running Π_{coin} generates a coin with bias at most $\frac{\log\log n}{(\log^{\beta} n \log\log n)^{\alpha}} \leq \frac{1}{\log n}$. Thus, the total bias of the coin is at most $\frac{1}{\log n} + o(\frac{1}{\log n})$. $\square$

Byzantine Agreement Protocol Using an SV-source Against Fail-stop Faults. Chor, Merritt and Shmoys [11] constructed a simple *one-round* BA protocol against fail-stop faults, which uses a uniformly random source. Below, we show that the same protocol achieves BA even if the source of randomness is a $\frac{1}{2\log n}$-SV-source.

Lemma 18. *There exists a BA protocol in a synchronous full-information network of* n *players tolerating a fail-stop* t*-adversary for* $t < (1-\epsilon)n$ *(for any* $\epsilon > 0$*). The protocol runs in expected* $O(1)$ *rounds, even if the randomness for the protocol is drawn from a* γ*-SV-source with* $\gamma = O(\frac{1}{\log n})$.

Proof. As usual, we focus on constructing a *common-coin* protocol. The protocol proceeds as follows.
(1) (Every player P_i) Sample $\log n$ random bits, and sends it to every other player.
(2) (Every player P_i) If there is a unique P_j from which P_i received message 0, elect P_j as leader, else output $\perp$.
(3) The leader flips a coin and sends it to everybody.

Let $\gamma = \frac{1}{2\log n}$. Let samp_i denote the $\log n$-bit string sampled by P_i from a γ-SV-source. Then, $\frac{1}{en} \leq (\frac{1}{2} - \frac{1}{2\log n})^{\log n} \leq \Pr[\mathsf{samp}_i = 0] \leq (\frac{1}{2} + \frac{1}{2\log n})^{\log n} \leq \frac{e}{n}$.

If *exactly* one of the *good* players P_i obtains $\mathsf{samp}_i = 0$ and *all* the bad players P_j obtain $\mathsf{samp}_j \neq 0$, then it is easy to see that all the honest players will choose an honest player as the leader. The probability that this happens is at least $(1-\frac{e}{n})^t \cdot \binom{n-t}{1}\frac{1}{en}(1-\frac{e}{n})^{n-t} \geq \kappa$ for some constant $\kappa > 0$. Thus, the protocol generates a (κ,γ)-common-coin in one round. $\square$

Acknowledgments. We gratefully acknowledge discussions with Ran Canetti, Rafael Pass, Guy Rothblum, Salil Vadhan and David Zuckerman.

References

[1] Miklós Ajtai and Nathan Linial. The influence of large coalitions. *Combinatorica*, 13(2):129–145, 1993.

[2] Michael Ben-Or. Another advantage of free choice: Completely asynchronous agreement protocols (extended abstract). In *PODC*, pages 27–30, 1983.

[3] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In *STOC*, pages 1–10, 1988.

[4] Michael Ben-Or and Nathan Linial. Collective coin flipping, robust voting schemes and minima of banzhaf values. In *FOCS*, pages 408–416, 1985.

[5] Michael Ben-Or, Elan Pavlov, and Vinod Vaikuntanathan. Byzantine agreement in the full-information model in $O(\log n)$ rounds. In *STOC*, 2006.

[6] Manuel Blum. Coin flipping by telephone. In *CRYPTO*, pages 11–15, 1981.

[7] Gabriel Bracha. An asynchronous $\lceil (n-1)/3 \rceil$-resilient consensus protocol. In *PODC*, pages 154–162, 1984.

[8] Gabriel Bracha. An $\mathrm{O}(\log n)$ expected rounds randomized byzantine generals protocol. *J. ACM*, 34(4):910–920, 1987.

[9] Benny Chor and Brian A. Coan. A simple and efficient randomized byzantine agreement algorithm. *IEEE Trans. Software Eng.*, 11(6):531–539, 1985.

[10] Benny Chor and Cynthia Dwork. Randomization in byzantine agreement. 5:443–497, 1989.

[11] Benny Chor, Michael Merritt, and David B. Shmoys. Simple constant-time consensus protocols in realistic failure models. *J. ACM*, 36(3):591–614, 1989.

[12] Benny Chor and Michael O. Rabin. Achieving independence in logarithmic number of rounds. In *PODC*, pages 260–268, 1987.

[13] Cynthia Dwork, David B. Shmoys, and Larry J. Stockmeyer. Flipping persuasively in constant time. *SIAM J. Comput.*, 19(3):472–499, 1990.

[14] Uriel Feige. Noncryptographic selection protocols. In *FOCS*, pages 142–153, 1999.

[15] Pesech Feldman and Silvio Micali. An optimal probabilistic protocol for synchronous byzantine agreement. *SIAM J. Comput.*, 26(4):873–933, 1997.

[16] Michael J. Fischer and Nancy A. Lynch. A lower bound for the time to assure interactive consistency. *Inf. Process. Lett.*, 14(4):183–186, 1982.

[17] Oded Goldreich, Shafi Goldwasser, and Nathan Linial. Fault-tolerant computation in the full information model. *SIAM J. Comput.*, 27(2):506–544, 1998.

[18] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or a completeness theorem for protocols with honest majority. In *STOC*, pages 218–229, 1987.

[19] Shafi Goldwasser and Silvio Micali. Probabilistic encryption and how to play mental poker keeping secret all partial information. In *STOC*, pages 365–377, 1982.

[20] Ronen Gradwohl, Salil Vadhan, and David Zuckerman. Random selection with an adversarial majority. *ECCC Report TR06-026*, 2006.

[21] Vassos Hadzilacos. Byzantine agreement under restricted types of failures. *Technical Report 18-83, Department of Computer Science, Harvard University*, 1983.

[22] Anna Karlin and Andrew Chi-Chih Yao. Probabilistic lower bounds for byzantine agreement. *Manuscript*, 1986.

[23] Jonathan Katz and Chiu-Yuen Koo. On expected constant-round protocols for byzantine agreement. *ECCC Report TR06-028*, 2006.

[24] Gil Neiger and Sam Toueg. Automatically increasing the fault-tolerance of distributed algorithms. *J. Algorithms*, 11(3):374–419, 1990.

[25] M. Pease, R. Shostak, and L. Lamport. Reaching agreement in the presence of faults. *Journal of the ACM.*, 27:228–234, 1980.

[26] Michael O. Rabin. Randomized byzantine generals. *FOCS*, pages 403–409, 1983.

[27] Alexander Russell and David Zuckerman. Perfect information leader election in $\log^* n + O(1)$ rounds. *JCSS*, 63(4):612–626, 2001.

[28] M. Santha and U. V. Vazirani. Generating quasi-random sequences from slightly-random sources. In *FOCS*, pages 434–440, Singer Island, 1984.