TBD manuscript No.
(will be inserted by the editor)

Shafi Goldwasser · Madhu Sudan · Vinod Vaikuntanathan

Distributed Computing with Imperfect Randomness

Abstract Randomness is a critical resource in many computational scenarios, enabling solutions where deterministic ones are elusive or even provably impossible. However, randomized solutions typically assume access to a source of unbiased, independent coins. Physical sources of randomness, on the other hand, are rarely unbiased and independent although they do seem to exhibit somewhat imperfect randomness. This gap in modeling questions the relevance of current randomized solutions to computational tasks. Indeed, there has been substantial investigation of this issue in complexity theory in the context of the applications to efficient algorithms and cryptography.

In this paper, we seek to determine whether imperfect randomness, modeled appropriately, is "good enough" for distributed algorithms. Namely, can we do with imperfect randomness all that we can do with perfect randomness, and with comparable efficiency?

We examine two natural models of imperfect randomness in a distributed setting – namely, where each player has an (independent) block source (resp. weak source). For the first model (that of block sources), we construct a general-purpose compiler that converts distributed algorithms that assume perfect randomness, to ones that work even with imperfect coins. The compiled protocol has the same fault-tolerance and round-complexity as the original protocol. For the second model (that of weak sources), we construct Byzantine Agreement protocols in a variety of scenarios (synchronous or asynchronous networks, with or without private channels). Our protocols have comparable fault-tolerance and round complexity as the best known Byzantine Agreement protocols that expect perfect randomness.

Shafi Goldwasser's work was supported in part by NSF CNS-0430450, a Minerva Grant 8495, and a Cymerman-Jakubskind award. Vinod Vaikuntanathan's work was supported in part by NSF CNS-0430450.

MIT CSAIL, Cambridge MA 02139, USA.
E-mail: {shafi,madhu,vinodv}@theory.csail.mit.edu

1 Introduction

Randomization has proved useful in many areas of computer science including algorithm design, cryptography, and distributed computing. In algorithm design, randomness has been shown to reduce the complexity requirments for solving problems, but it is unclear whether the use of randomization is inherently necessary. Indeed, an extensive amount of research in the complexity theory is dedicated to de-randomization : the effort to replace random strings by deterministic "random-looking" strings.

In contrast, the case of using randomness within the field of distributed computing is unambiguous. There are central distributed computing problems for which there are *provably* no deterministic solutions, whereas efficient randomized solutions exist (Byzantine agreement in an asynchronous network, for instance) [16].

The common abstraction used to model the use of randomness by a protocol is to assume that each participant's algorithm has access to its own source of unbiased and independent coins. This abstraction, however, does not seem to correspond exactly to the physically available sources of randomness. Physical sources instead generate an output that seems only to be "somewhat random".

1.1 Models of Randomness in the Real World

The gap between the abstract model and the physically available sources of randomness has been addressed beginning with the work of von Neumann [23] and Elias [12] which deal with sources of *independent bits of unknown bias*. In later works, sources of dependent bits were modeled by Santha and Vazirani [21], Chor and Goldreich [7], and finally Zuckerman [24] who presented the *weak random source* generalizing all previous models.

Informally, for a weak random source, no sequence of bits has too high a probability of being output. A weak random source is a *block source* [7] if this is guaranteed

for every output block (for a block size which is a parameter of the source) regardless of the values of the previous blocks. Namely, whereas a weak random source guarantees some minimum amount of entropy if sampled *exactly once*, a block source guarantees a minimum amount of entropy each time a sample is drawn (where a sample corresponds to a block).

Two natural questions arise.

1. Can weak random sources be used to extract a source of unbiased and independent coins?
2. Even if not, can weak random sources be used within applications instead of perfect random sources, with the same guarantee of correctness and complexity?

The first question was addressed early on, in conjuction with introducing the various models of imperfect randomness. It was shown that it is impossible to extract unbiased random coins with access to a single weak random source [21,7,24]. Researchers went on to ask (starting with Vazirani [22]) whether, given two (or more) weak random sources (all independent from each other), extraction of unbiased random bits is possible. Indeed, it was shown [22,7,24] that two sources suffice. Whereas original works focus on in-principle results, recent work by Barak, Impagliazzo, and Wigderson [1] and others focuses on constructive protocols.

The second question is the type we will we focus on in this work. In the context of probabilistic algorithms, it was shown early on in [7,24] that a single weak random source can be used to replace a perfect source of randomness for any BPP algorithm. Very recently, Dodis et al [10,9] initiated an investigation of the same question in the context of cryptographic protocols. Namely, is it possible for cryptographic appplications (e.g. encryption, digital signatures, secure protocols) to exist in a world where each participant has access to a *single* weak source of randomness? Surprisingly, they show that even if these sources are block sources which are independent of each other, many cryptographic tasks such as encryption and zero-knowledge protocols are *impossible*.

We thus are faced with a natural and intriguing question in the context of distributed computing:

> *Are weak random sources strong enough to replace perfect sources in distributed computing protocols?*

This is the starting point of our research.

1.2 The Choice of our Randomness Model

The model of randomness we assume in this work is that each player has its own weak source (or block source) that is independent of the sources of all the other players, as was assumed in the work of [9] in the context of cryptographic protocols. We feel that this model is a natural starting point for the study of randomness in distributed computation. We note however that there is a spectrum of models that may be assumed, and one such alternative is discussed in the section on future directions (Section 5.1.1).

1.3 Byzantine Agreement: Randomized versus Deterministic Protocols

We briefly review the problem of Byzantine Agreement and the best randomized protocols for this problem in various settings.

The problem of Byzantine Agreement (BA) defined by Pease, Shostak and Lamport [18] is for n players to agree on a value, even if some t of them are faulty. Informally, for any set of initial values of the players, a BA protocol should satisfy the following:

- *Consistency:* All non-faulty players agree on the same value.
- *Non-triviality:* If all the players started with some value v, they agree on v at the end of the protocol.

The faulty players might try to force the non-faulty players to disagree. The good players, in general, do not know who the faulty players are. A protocol for Byzantine Agreement should ensure that the good players agree, even in the presence of such malicious players.

The possibility of BA depends crucially on the model of communication among the players. When the players communicate via a synchronous network with point-to-point channels, there are $(t+1)$-round deterministic BA protocols (one in which no player tosses coins) even in the presence of $t < \frac{n}{3}$ faults [17]. A lower bound of $t+1$ communication rounds is known for every deterministic protocol. When the players communicate via an asynchronous network, the celebrated result of Fischer, Lynch and Paterson [15] shows that BA is impossible to achieve even in the presence of a single faulty player.

Yet, Ben-Or [2] in 1983 showed how to achieve Byzantine agreement in an asynchronous network tolerating a linear number of faults via a randomized protocol with expected exponential round complexity. More efficient randomized protocols in asynchronous as well as synchronous networks followed, some of which (due to [19, 4,11,14,13,5]) assume the existence of private communication channels between pairs of participants (or alternatively cryptographic assumptions), and some do not require secret communication (notably Chor-Coan [6] and Ben-Or et al. [3]).

To summarize these works, both synchronous and asynchronous BA can be achieved via a randomized protocol in expected $O(1)$ number of rounds tolerating an optimal number of faults, assuming private channels of communication exist. Without any secret communication requirements, for $t < n/3$ a randomized protocol exists for synchronous BA using $O(\frac{t}{\log n})$ rounds, whereas the best asynchronous BA protocol still requires exponential number of rounds [2,4]. Table 1.3 summarizes these results.

	Private Channels	Non-Private Channels
Synchronous	$O(1)$ rounds [14] ($t < \frac{n}{3}$)	$O(\frac{n}{\log n})$ rounds [6] ($t < \frac{n}{3}$) $O(\log n)$ rounds [3] ($t < \frac{n}{4}$)
Asynchronous	$O(1)$ rounds [5] ($t < \frac{n}{3}$)	$2^{O(n)}$ rounds [4] ($t < \frac{n}{3}$)

Table 1 The Best Known Byzantine Agreement Algorithms

1.4 Overview of our Results and Techniques

We focus on the case where a distributed algorithm is run on a complete network of n participants t of which can be Byzantine faults. The case of an arbitrary network can be handled assuming sufficient connectivity. We address the settings of both synchronous and asynchronous networks, and the cases of private channels (when each pair of participants have a secret communication channel between them) and of a full information network (when no secrecy is assumed for any communication). We note that by the results of Dodis et al. [9], making cryptographic assumptions is doomed for failure.

1. In the case of *block sources*: Let Π be any randomized distributed algorithm. We construct a compiler $\mathcal{C}$ that takes Π as input and outputs a protocol $\Pi' \stackrel{\mathsf{def}}{=} \mathcal{C}(\Pi)$ that simulates Π in a strong sense. Π' works even when the players are given independent block sources (as opposed to perfect randomness that Π expects). Π' runs for just one extra round compared to Π, and has the same fault-tolerance as Π.
2. In the case of *weak sources*: We focus on the specific problem of Byzantine Agreement. For the various settings of network and communication models, we design protocols for Byzantine Agreement that work with weak sources and obtain the bounds on fault-tolerance and round-complexity achieved by the best randomized Byzantine Agreement protocols that work with perfect randomness. In the private channels case, we show for both synchronous and asynchronous networks an $O(1)$ expected rounds protocol for $t < \frac{n}{3}$ faults (matching [14,5]). In the full-information model, we show a protocol with $O(\frac{t}{\log n})$ expected rounds for synchronous networks (with $t < \frac{n}{3}$, matching [6]) and a $O(2^n)$ expected round protocol for $t < \frac{n}{3}$ (matching [4]).

The compiler constructed in the case of block sources follows a 2-step *Extract and Simulate* approach. $\mathcal{C}$ utilizes the first $O(1)$ rounds for a pre-processing protocol, in which the parties interact with each other so that at the end, *all of them* obtain a *private, uniformly random* string. The randomness so obtained is used to simulate Π.

We construct various extraction protocols, in which the players interact to obtain unbiased and independent random bits. The problem that we will need to overcome is naturally that when a player receives a sample from another player (which may be faulty), he cannot assume that the sample is good and not constructed to correlate with other samples being exchanged. We construct extraction protocols that work *even if* some of the players contribute bad inputs which may depend on samples that have seen sent by honest players (in the case of full information protocols).

As building blocks, we will use the extractors of [24, 7,20] as well as the *strong extractors* of [8,20]. A strong extractor ensures that the output of the extraction is random even if one is given *some* of the inputs to the extractor.

In the case of weak sources, we build in various ways on top of the existing distributed algorithms for Byzantine Agreement. Our extraction procedures will guarantee that a certain fraction of the non-faulty players obtain perfectly unbiased and independent coins. However, this will not necessarily be the case for *all* the non-faulty players, and thus one may fear that now when running existing randomized BA protocols with perfect randomness only available to some of the non-faulty players, the fault-tolerance of the final protocol may go down. Luckily this is not the case, due the following interesting general observation.

When we analyze the current usage of randomness in [14,6], we find on closer look that one may distinguish between how many non-faulty players truly need to have access to perfectly unbiased and independent sources of random coins, and how many non-faulty players merely need to follow the protocol instructions. The number of non-faulty players which need to have access to perfect coins is drastically lower than the total number of non-faulty players. In the Feldman-Micali protocol[14], it suffices for $t+1$ players to posses good randomness whereas we need all the $n - t$ non-faulty players to follow the protocol to prove correctness and expected $O(1)$ termination. In the case of the Chor-Coan protocol [6], it suffices for $(\frac{1}{2} + \delta)n$ (for arbitrarily small constant $\delta > 0$) players to possess good randomness.

Organization of this paper In Section 2 we describe the communication models as well as the models of imperfect randomness considered in this paper. In Section 2.5 we descibe the technical ingredients used to extract pure randomness from imperfect randomness. In particular, we construct some new, though simple, extractors that use a mix of imperfect randomness and adversarial inputs from multiple sources to extract pure randomness. In Section 4 we construct the compiler that converts an arbitrary distributed algorithm to one that works even when the randomness comes from a block source. In Section 5, we construct optimal Byzantine agreement protocol in various settings, that work with weak sources.

2 Definitions and the Model

2.1 Preliminaries

Throughout this paper, we use letters X and Y to denote random variables. Probability distributions are denoted by letters $\mathcal{C}$ and $\mathcal{D}$. We sometimes identify a random variable X with its distribution $\mathcal{D}$.

The Statistical Distance between distributions $\mathcal{C}$ and $\mathcal{D}$ on a set S,

$$\| \mathcal{C} - \mathcal{D} \| \stackrel{\mathsf{def}}{=} \frac{1}{2} \sum_{s \in S} |\mathcal{C}(s) - \mathcal{D}(s)|.$$

When $\| \mathcal{C} - \mathcal{D} \| \leq \epsilon$, we say that the distributions $\mathcal{C}$ and $\mathcal{D}$ are ϵ-close, and denote it by $\mathcal{C} \approx_\epsilon \mathcal{D}$ (or sometimes simply $\mathcal{C} \approx \mathcal{D}$).

We denote by U_k the uniform distribution on the set $\{0,1\}^k$. We let $X \circ Y$ denote the joint distribution of the random variables X and Y. In particular, $X \circ U_k$ denotes the joint distribution of the random variable X and an *independent* uniformly distributed random variable U_k.

2.2 Imperfect Random Sources

A k-bit source of randomness is simply a random variable X whose support is the set $\{0,1\}^k$. The randomness contained in a source is quantified by its *min-entropy*. The min-entropy of a random variable X, supported on the set $\{0,1\}^k$ is defined as

$$\mathbf{H}_\infty(X) \stackrel{\mathsf{def}}{=} \min_{x \in \{0,1\}^k} \{ -\log_2(\Pr[X = x]) \}.$$

Definition 1 (Weak Source) A (k, δ)**-weak source** (or simply a (k, δ)-source) is a random variable X supported on the set $\{0,1\}^k$ such that for any $x \in \{0,1\}^k$, $\Pr[X = x] \leq 2^{-\delta k}$.

A sequence of random variables $X_1 \circ X_2 \circ \ldots \circ X_\ell$ is called a (k, δ)-block source if each block X_i has support $\{0,1\}^k$ and has min-entropy δk, even conditioned on any values of all the other blocks. This notion corresponds to sampling multiple times from a source of random bits, wherein we are guaranteed that each sample has *some fresh* entropy. The block-length k specifies how often new entropy is generated by the source. Let $\overline{X}_i$ denote the concatenation of all blocks excluding X_i.

Definition 2 (Block Source) A sequence of random variables, $X_1 \circ X_2 \circ \ldots \circ X_\ell$ is called a (k, δ)**-block source** if for all i, x_i and $\overline{x}_i$, $\Pr[X_i = x_i \mid \overline{X}_i = \overline{x}_i] \leq 2^{-\delta k}$.

2.3 The Network, Communication and Fault Models

A distributed system is a set of processors joined by bidirectional communication channels. For simplicity, we always assume that the processors are fully connected to each other. Let n be the number of processors and let $\mathbf{P} \stackrel{\mathsf{def}}{=} \{P_1, P_2, \ldots, P_n\}$ be the set of processors.

The communication channels in the network could be either synchronous and asynchronous. In the *synchronous* case, the communication proceeds in rounds. At the beginning of round i, all the processors send messages to each other. The messages are delivered at the end of round i, after which the processors perform local computation and change state. In the case of *asynchronous* communication, however, the only guarantee is that the messages sent are *eventually* received by the recipient. Messages can be arbitrarily re-ordered, and arbitrarily delayed. We present a more formal model of asynchronous systems in Section 4.

A further characterization of the channels depends on whether they allow for private communication between the pair of processors involved. In the *private channels* model, the communication between players i and j is invisible to all the players but i and j. In contrast, in the *full-information model*, the communication between any two players is visible to the adversary.

In this paper, we exclusively deal with Byzantine faults. Byzantine players can deviate arbitrarily from the prescribed protocol, and co-ordinate with each other so as to mislead the good players into disagreement. The coalition of Byzantine players is informally referred to as the adversary. We *do not* assume that the adversary is computationally bounded. In addition, in the synchronous case, we allow the adversary to be *rushing*. i.e, the adversary can see all the messages sent by the good players in a round r, before deciding what to send in round r. In other words, all the good players send messages in the beginning of a round, whereas the bad players can delay sending messages until later in the round, when they (possibly) have had the chance to learn about the good players' messages.

2.4 Our Model of Randomness

Each player P_i has its own source of imperfect randomness X_i (which is either a block source or a weak source). We assume that the random sources of any two players are *independent*. That is, the random variables X_i and X_j are independent for any i and j.

2.5 Extracting Pure Randomness

Naturally, the first thing to attempt, given a (k, δ)-source X, would be to extract "pure randomness" from X. That is, we would like to construct a deterministic function $\mathsf{Ext} : \{0,1\}^k \to \{0,1\}^m$ (for some $m > 0$) such that *for any* (k, δ)-source X, $\| \mathsf{Ext}(X) - U_m \|$ is small. The condition that the extractor works for *any* (k, δ)-source corresponds to the natural limitation that we cannot assume anything about the structure of the source except that it has a certain min-entropy.

Unfortunately, it is easy to show that this task is impossible in general. Thus, it is natural to ask if one can extract uniform randomness given *two independent* (k, δ)-sources. Chor-Goldreich [7] answered this in the affirmative for the case when $\delta > \frac{1}{2}$. In particular, they built a function $\mathsf{Ext} : (\{0,1\}^k)^2 \to \{0,1\}$ such that for some $\epsilon > 0$, $\| \mathsf{Ext}(X, Y) - U_1 \| \leq \epsilon$. We call such an Ext a two-source (deterministic) extractor.

More recently, Raz [20] showed this for the case when one of the two sources has min-entropy at least $\frac{k}{2}$ and the other has min-entropy at least $\log k$. Below, we formally define the notion of a deterministic two-source extractor, which is a key tool in our constructions.

Definition 3 A function $\mathsf{Ext} : (\{0,1\}^k)^2 \mapsto \{0,1\}^m$ is called a (k, δ)**-two-source extractor** if for any two independent (k, δ)-sources X and Y,

$$\| \mathsf{Ext}(X, Y) - U_m \| \leq \epsilon.$$

A strong extractor is a generalization of this notion. A strong two-source extractor is one in which the output of the extractor is independent of each of the inputs separately. In other words, the output of the extraction is random even given one of the inputs. More formally,

Definition 4 A function $\mathsf{Ext} : (\{0,1\}^k)^2 \mapsto \{0,1\}^m$ is a (k, δ)**-two-source *strong* extractor** if for any two independent (k, δ)-sources X and Y,

$$\| \mathsf{Ext}(X, Y) \circ X - U_m \circ X \| \leq \epsilon.$$

Dodis and Oliveira [8] show that some well-known constructions of two-source deterministic extractors indeed yield two-source *strong* extractors. In a recent result, Raz [20] shows how to construct very general two-source *strong* extractors.

3 Extracting Randomness in a Network

Each player participating in a randomized distributed protocol is traditionally assumed to have a uniformly distributed string that is independent of the random strings of the other players. In addition, some protocols assume that the randomness of each player is *private*. i.e, the faulty players have no information on the randomness of the good players. There is no guarantee on the behavior of the protocol if the players use an imperfect random source or if the randomness is public.

Our goal would be to run a distributed extraction protocol among the players such that the good players help each other extract a uniform random string collectively from their (mutually independent) weak random sources, even in the presence of some malicious parties. The malicious colluding parties could each contribute an arbitrary string, possibly correlated with what they see in the network, as input to the extraction protocol.

The main building block in our randomness extraction *protocols* is a multi-source extractor whose output is random even if an arbitrary subset of the input sources do not have any min-entropy.

Definition 5 (Immune Extractor) Let X be a (k, δ)-block source, and let $Y_1, Y_2, \ldots, Y_p$ be random variables, at least q of which are (k, δ)-block sources. Assume that all the sources are independent. Then, a function $\mathsf{Ext} : (\{0,1\}^*)^{p+1} \mapsto \{0,1\}^m$ is called a (p, q)-immune (k, δ)-extractor if $\| \mathsf{Ext}(X, Y_1, \ldots, Y_p) - U_m \| \leq \epsilon$ for some $\epsilon > 0$.

In the above definition, we are guaranteed that the $p - q$ "bad" sources are independent of the $q + 1$ "good" sources. We will need an extractor that works even if all the q sources Y_i are not independent of each other (We still need to assume that all the Y_i's are independent of X). A (p, q)-strongly immune extractor extracts uniform randomness under these more stringent conditions.

Definition 6 (Strongly Immune Extractor) Let X be a (k, δ)-block source, and let $Y_1, Y_2, \ldots, Y_p$ be random variables, at least q of which are (k, δ)-block sources. Assume that, $\forall i$, Y_i is independent of X. Then, a function $\mathsf{Ext} : (\{0,1\}^*)^{p+1} \mapsto \{0,1\}^m$ is called a (p, q)-strongly immune (k, δ)-extractor if $\| \mathsf{Ext}(X, Y_1, \ldots, Y_p) - U_m \| \leq \epsilon$ for some $\epsilon > 0$.

We can define the notions of a (p, q)-immune *strong* extractor and a (p, q)-strongly immune strong extractor by requiring the stronger condition that

$$\| \mathsf{Ext}(X, Y_1, \ldots, Y_p) \circ Y_1 \circ \ldots \circ Y_p - U_m \circ Y_1 \circ \ldots \circ Y_p \| \leq \epsilon.$$

The motivation for these definitions is as follows. Some distributed protocols might require the players to have private randomness. But, if the players are connected by non-private channels, most of the inputs to the extraction protocols will be visible to the adversary. If the output of the extraction protocol indeed depends on the values that were publicly transmitted, then the extracted randomness is not private. We need to construct (p, q)-strongly immune strong extractors to ensure that the randomness extracted is indeed private. Table 3 gives the construction of the immune extractor. We show that,

- If Ext is a two-source extractor, then IExt is a $(p, 1)$-immune extractor, and
- If Ext is a two-source strong extractor, then IExt is a $(p, 1)$-strongly immune strong extractor.

Theorem 1 *Suppose* Ext *is any* (k, δ)*-two source extractor. Then,* IExt *is a* $(p, 1)$*-immune* (k, δ)*-extractor.*

Proof Without loss of generality, let Y_1 be the (k, δ)-source. Denote by X_1' the distribution of X conditioned on all the X_i $(i > 1)$. Since X is a block source, it follows that $\mathbf{H}_\infty(X_1') \geq \delta k$. Thus $\mathsf{Ext}(X_1', Y_1)$ is ϵ-close to U_m.

The Construction of $\mathsf{IExt}(\mathsf{Ext}, X, Y_1, \ldots, Y_p)$

Inputs:

- An extractor $\mathsf{Ext}(\cdot,\cdot)$.
- Let $X_1, X_2, \ldots, X_p$ denote p distinct blocks of the (k,δ)-block source X.
- Let $Y_1, \ldots, Y_p$ be one block each from p other sources.

$\mathsf{IExt}(\{X_i\}_{i=1}^p, Y_1, \ldots, Y_p) = \bigoplus_{i=1}^p \mathsf{Ext}(X_i, Y_i)$.

Table 2 Construction of an Immune Extractor

Now, all the Y_i are independent of Y_1. Therefore, for any i, $\mathsf{Ext}(x_i, Y_i)$ is independent of $\mathsf{Ext}(X_1', Y_1)$. A straightforward consequence is that $\bigoplus_{i=1}^p \mathsf{Ext}(X_i, Y_i)$ is ϵ-close to U_m.

Now, suppose the Y_i are allowed to depend on each other. Using a simple (not necessarily strong) extractor in the construction of IExt is not guaranteed to work, since a "'bad"' Y_i could be chosen such that $\mathsf{Ext}(X_i, Y_i)$ is non-trivially correlated with $\mathsf{Ext}(X_1', Y_1)$. However, using a strong extractor eliminates this difficulty – informally, a strong extractor guarantees that $\mathsf{Ext}(X_1', Y_1)$ is independent of Y_1. Thus, even though $\mathsf{Ext}(X_j, Y_j)$ depends on Y_1, it has no influence on the output of IExt.

Theorem 2 *Suppose* Ext *is any* (k,δ)*-two source strong extractor. Then,* IExt *is a* $(p,1)$*-strongly immune* (k,δ)*-strong extractor.*

Proof Without loss of generality, let Y_1 be the (k,δ)-source. Denote by X' the distribution of X_1 conditioned on arbitrary values of the other blocks. That is, let $X_1' \stackrel{\mathsf{def}}{=} [X_1 | X_2 = x_2, \ldots, X_p = x_p]$. Then, since X is a block source with min-entropy δk, it follows that the distribution X_1' has min-entropy δk. Thus $\mathsf{Ext}(X_1', Y_1)$ is ϵ-close to U_m.

Let $D_1 = \mathsf{Ext}(X_1', Y_1)$ and let $D_i = \mathsf{Ext}(X_i, Y_i)$ for $i > 1$. From the fact that Ext is a strong extractor, we have that $\big[D_1, Y_1\big] \approx \big[U_m, Y_1\big]$.

Now, an adversary generates Y_i $(i > 1)$ as a function of Y_1. Thus, by Proposition 1,

$$\big[D_1, Y_1, Y_2, \ldots, Y_p\big] \approx \big[U_m, Y_1, Y_2, \ldots, Y_p\big].$$

Recall that $D_1 = \mathsf{Ext}(X_1', Y_1)$, and that both X_1' and Y_1 are independent of $X_2, \ldots, X_p$. Thus,

$$\big[D_1, X_2, X_3, \ldots, X_p\big] \approx \big[U_m, X_2, X_3, \ldots, X_p\big].$$

Since the X_i are independent of the Y_i,

$$\big[D_1, Y_1, \ldots, Y_t, X_2, \ldots, X_t\big] \approx \big[U_m, Y_1, \ldots, Y_t, X_2, \ldots, X_t\big].$$

Therefore, again by Proposition 1,

$$\big[D_1, Y_1, Y_2, \ldots, Y_t, \mathsf{Ext}(X_2, Y_2), \ldots, \mathsf{Ext}(X_t, Y_t)\big] \approx \big[U_m, Y_1, Y_2, \ldots, Y_t, \mathsf{Ext}(X_2, Y_2), \ldots, \mathsf{Ext}(X_t, Y_t)\big].$$

Construction of the Compiler $\mathcal{C}(\Pi)$

Input: A Protocol Π.
Output: A Protocol Π', whose code is as below:

1. (Each player P_i) Sample $n-1$ blocks of randomness Y_i^j and send the j^{th} block Y_i^j to player P_j.
2. (Each player P_i) Wait to receive messages from $t+1$ players.
3. (Each player P_i) Denote by Y_j^i the block received from player P_j (If no message is received from P_j, set $Y_j^i = 0$). Sample n blocks from the source and denote the sample as $X = (X_1, \ldots, X_n)$.
4. – Private Channels Case: Let Ext be some two-source extractor. Set $R_i = \mathsf{IExt}(\mathsf{Ext}, X, Y_1, \ldots, Y_n)$.
   – Non-private Channels Case: Let Ext be some two-source *strong* extractor. Set $R_i = \mathsf{IExt}(\mathsf{Ext}, X, Y_1, \ldots, Y_n)$.
5. Run Π with R_i as the random tape of player P_i.

Table 3 The Compiler $\mathcal{C}$ for the case of Block Sources

Note that $D_i \stackrel{\mathsf{def}}{=} \mathsf{Ext}(X_i, Y_i)$. This means that, $D_1 \approx U_m$, even given D_i $(i > 1)$ and the Y_i. Thus,

$$\big[\bigoplus_{i=1}^{t} D_i, Y_1, \ldots, Y_t\big] \approx \big[U_m, Y_1, \ldots, Y_t\big].$$

Proposition 1 *Assume* X_1*,* X_2*,* Y *and* Z *are random variables such that* Z *is independent of* X_1 *and* X_2*. If* $(X_1, Y) \approx (X_2, Y)$*, then* $(X_1, f(Y,Z)) \approx (X_2, f(Y,Z))$*.*

4 Simulating Distributed Algorithms with Block Sources

We are ready to show how to execute randomized distributed algorithms, assuming that the players have independent block sources. We exhibit a compiler $\mathcal{C}$ which, when given any distributed algorithm Π as input, outputs an algorithm Π' that "behaves exactly like" Π, but works even when the randomness comes from a block source.

The compiler first runs a 1-round pre-processing protocol, at the end of which *all* the honest players obtain perfectly random and independent strings. Moreover, the randomness extracted by a player is independent of the views of *all* the other players (and is thus, private). This randomness can now be used to simulate any distributed protocol Π. See Table 4 for the construction of the compiler $\mathcal{C}$.

First of all, we need to define what it means for the protocol Π' to simulate protocol Π. The view of a player P in a protocol Π, denoted as $\mathsf{view}_\Pi(P)$ is the set of all messages received by P, the inputs of P in the protocol Π and the random tape of P. The view of a set $\mathcal{S}$ of players is simply $\cup_{P \in \mathcal{S}} \mathsf{view}_\Pi(P)$.

The following definition captures what it means for a protocol to have the same outcome as another protocol.

Informally, we say that Π' simulates Π, if for any adversary that produces some effect on the protocol Π', there is another adversary (simulator) that produces the same effect in protocol Π. Formally,

Definition 7 (Π' simulating Π) A protocol Π' is said to simulate a protocol Π, if for any adversary $\mathcal{A}$ for the protocol Π, there exists an adversary $\mathcal{A}'$ for the protocol Π' such that $\mathsf{view}_{\Pi'}(\mathbf{P}) \approx \mathsf{view}_{\Pi}(\mathbf{P})$.

Theorem 3 (Block Sources) *Suppose Π is a distributed protocol. Then, the protocol $\Pi' = \mathcal{C}(\Pi)$ simulates Π.*

The key fact we use is that, for each honest player P_i, the randomness extracted R_i is independent of the messages passed in the network before Step 5 (See Table 4). This fact follows trivially from Theorems 1 and 2.

Proof Given any adversary $\mathcal{A}' = (\mathcal{A}_1, \mathcal{A}_2)$ for Π', we construct an adversary $\mathcal{A}$ for Π as follows. $\mathcal{A}$ generates, for every honest player, n blocks of randomness, and sends t of them to the $\mathcal{A}_1$. It then runs $\mathcal{A}_2$ with the output of $\mathcal{A}_1$ as its input.

Denote by R_h the joint distribution of the random tapes of all the honest players after Steps 1–4 of protocol Π', and by $R_{\mathcal{A}}$ the view of the adversary in the protocol Π'. We first observe that the joint distribution $(R_h, R_{\mathcal{A}}) \stackrel{s}{\equiv} (U_\ell, R_{\mathcal{A}})$ for some $\ell > 0$. Thus, since the input distribution of the players in both cases is ϵ-close, the views at the end of the protocols are ϵ-close too, by Proposition 1.

As a corollary, we get that the best known Byzantine Agreement protocols can be executed even when the randomness of each player comes from a block source, with an overhead of one round, and with the same fault-tolerance.

Corollary 1 *If $n \geq 3t+1$, then there exist BA protocols that tolerate t faults, when the players have (k, δ) block-sources with $\delta > \frac{1}{2}$, and*

- *run in $O(1)$ rounds, in a synchronous network with private channels,*
- *run in $O(1)$ rounds, in an asynchronous network with private channels,*
- *run in $O(\log n)$ rounds, in a synchronous network with non-private channels,*
- *run in $O(2^n)$ rounds, in an asynchronous network with non-private channels.*

5 Simulating Distributed Algorithms with Weak Sources

This section deals with simulating distributed algorithms using a weak random source. We would not be able to simulate arbitrary distributed algorithms this way. Instead, we show how to construct randomized *Byzantine agreement* (BA) protocols that work even when the players have access to weak sources. The protocols we construct have running time and fault-tolerance that match the best known randomized BA protocols that use perfect randomness.

Our transformations are fairly generic and we describe a general class of distributed algorithms on which they can be applied. First of all, we need the following proposition that shows us how to get a block source with a small number of blocks from a weak source with a high enough min-entropy. Proposition 2 says exactly that.

Proposition 2 *Fix a constant $c > 0$. Assume X is a (k, δ)-weak source, where $\delta > 1 - \frac{1}{2c}$. Then, we can deterministically construct a (k', δ')-block source $Y = Y_1 \circ \ldots \circ Y_c$ from X, where $k' = \frac{k}{c}$ and $\delta' > \frac{1}{2}$.*

We run a pre-processing protocol, just as in the case of block sources, but among a small number of players. This will guarantee that a sufficiently large fraction of the honest players get truly random strings at the end of the pre-processing. We conclude by showing that some Byzantine Agreement protocols do not require *all* the honest players to have random strings – it is sufficient that a large fraction of the honest players have random strings.

Protocol Private-Channels-Extract

1. Group the players $P_1, P_2, \ldots, P_n$ into pairs $(P_1, P_2), \ldots, (P_{n-1}, P_n)$. Let Ext be a two-source extractor.
2. (Each player P_i)
   - If i is even, sample a k-bit string X_i from the source, and send it to P_{i-1}.
   - If i is odd, sample a k-bit string X_i from the source, and receive a k-bit string X_{i+1} from P_{i+1}. Compute an m-bit string $R_i \leftarrow \mathsf{Ext}(X_i, X_{i+1})$. Send to P_{i+1} the first $\frac{m}{2}$ bits of R_i and store the remaining bits.

Table 4 Pre-processing protocol for the Synchronous, Private Channels case

5.1 Byzantine Agreement Protocols

The protocol Synch-PC-Extract ensures that, in the presence of at most t faults, at least $2\lfloor \frac{n}{2} \rfloor - 2t$ good players get private random strings.

Theorem 4 (Synchronous, Private Channels) *Let $n \geq 3t + 2$. then there exists a BA protocol that runs in expected $O(1)$ rounds tolerating t faults, assuming the players are connected by a synchronous network with private channels, and have (k, δ) block-sources with $\delta > \frac{1}{2}$.*

Proof In the first round, the players run the protocol Synch-PC-Extract. Let R_i denote the output of player i after running Synch-PC-Extract. Now, the players run

Protocol Asynch-Extract

1. Each player p_i does the following: (Note: Ext is either a $(t+1,t)$-immune extractor or a $(t+1,t)$-strongly immune strong extractor).
   - Wait to receive $t+1$ strings $Y_1, Y_2, \ldots, Y_{t+1}$ from $t+1$ different players.
   - Sample blocks $X_1^1, X_1^2, \ldots, X_1^{t+1}$ from the random source.
   - Compute and Store $R_i \leftarrow \mathsf{Ext}(\{X_1^j\}_{j=1}^{t+1}, Y_1, Y_2, \ldots, Y_{t+1})$.

Table 5 Pre-processing protocol for the Asynchronous case

Protocol Synch-FI-Extract

1. Group the players into 4-tuples (p_1,p_2,p_3,p_4), ..., $(p_{n-3},p_{n-2},p_{n-1},p_n)$. Let SI-*ext* be a $(3,2)$-strongly immune strong extractor. (Note: Assume for simplicity that n is a multiple of four. If not, add at most two dummy players.)
2. Each player p_i does the following: (Assume that p_i is in a 4-tuple with p_{i+1}, p_{i+2} and p_{i+3}.)
   - Samples six blocks X_1^j $(j=1,\ldots,6)$ from its random source.
   - Send X_1^j to p_{i+j} (for $j=1,\ldots,3$). Store X_1^j $(j=4,\ldots,6)$.
   - Receive k-bit strings Y_j from p_{i+j} $(j=1,\ldots,3)$.
   - Compute $R_i \leftarrow$ SI-$ext(\{X_1^4, X_1^5, X_1^6\}, Y_1, Y_2, Y_3)$ and store R_i.

Table 6 Pre-processing protocol for the Synchronous, Full-Information case

the BA protocol of Feldman and Micali [14] using R_i as randomness.

There are at least $\lfloor \frac{n}{2} \rfloor - t \geq \lfloor \frac{t}{2} \rfloor + 1$ pairs such that both the players in the pair are good. In each pair, the players extract uniform and independent random strings. Thus, there are at least $2(\lfloor \frac{t}{2} \rfloor + 1) \geq t+1$ players at the end of the protocol with m-bit strings that are ϵ-close to uniform. Because of the private channels assumption, the inputs used to compute R_i are invisible to the adversary, and therefore, the randomness extracted is private.

Theorem 5 (Synchronous, Full-Information) *Assume $n \geq 3t+1$. Then, there exists a BA protocol that runs in expected $O(\frac{t}{\log n})$ rounds tolerating t faults, assuming the players are connected by a synchronous network with non-private channels, and have (k,δ) block sources with $\delta > \frac{1}{2}$.*

Proof In the first round, the players run the protocol Synch-FI-Extract. Using the randomness so obtained, run the BA protocol guaranteed by Lemma 1.

Consider the set of 4-tuples of players such that at most two players in the 4-tuple are bad. There are at least $\lfloor \frac{n}{4} \rfloor - \lfloor \frac{t}{3} \rfloor \geq \lfloor \frac{5t}{12} \rfloor$ such tuples. In each such pair, the good players extract uniform and independent random strings, since there are at least two good players in such a 4-tuple and Ext is a $(3,2)$-*strongly immune* extractor. There are at least $4\lfloor \frac{5t}{12} \rfloor \geq \frac{5}{9}n = (\frac{1}{2} + \Theta(1))n$ players at the end of the protocol with m-bit strings that are ϵ-close to uniform. Moreover, the random strings R_i of these players are private, since Ext is a strong extractor. Now, invoke Lemma 1 to complete the proof.

Lemma 1 *If $n \geq 3t+1$, there exists a BA protocol that runs in expected $O(\frac{t}{\log n})$ rounds tolerating t faults in a synchronous network with non-private channels, even if only $(\frac{1}{2}+\delta)n$ good players have private randomness (for some $\delta > 0$).*

Proof The protocol of Chor and Coan [6] is such a BA protocol. We first sketch the protocol and then prove that it indeed has the claimed property.

The players are divided into fixed disjoint groups of size g. The i^{th} group consists of the set of players $\{p_{(i-1)g+1}, \ldots, p_{ig}\}$. For any player p_i, let $\mathsf{GROUP}(p_i)$ denote the group that p_i belongs to. The protocol proceeds in phases where, in each phase, the players try to reach agreement on their values. In each phase, one of the groups is said to be *active*. The purpose of the players in the active group is (among other things) to toss coins and send it to all the other players.

1. For $e = 1$ *to* ∞, each player p_i does the following: (Note: e is the current phase.)
   (a) Sends the message $(e, \mathsf{Phase}_1, b_i)$ to every player.
   (b) Receive messages from every other player of the form $(e, \mathsf{Phase}_1, *)$.
   (c) If for some v, there are $\geq n-t$ messages of the form (e, Phase_1, v), then set $b_i \leftarrow v$, else set $b_i \leftarrow$ "?"
   (d) If $\mathsf{GROUP}(p_i) \equiv e \ (mod \ \lfloor \frac{n}{g} \rfloor)$ then set $coin \leftarrow b$, else set $coin \leftarrow 0$ {Note : b is a random bit}
   (e) Send the message $(e, \mathsf{Phase}_2, b_i, coin)$ to all players.
   (f) Receive messages of the form $(e, \mathsf{Phase}_2, c, coin)$ from every player.
   { Note: Let $\mathsf{NUM}(c)$ be the number of messages received that contain c. }
   (g) If $\mathsf{NUM}(c) \geq n-t$ for some bit c, **decide** c.
   (h) Else, if $\mathsf{NUM}(c) \geq t+1$ and $\mathsf{NUM}(c) > \mathsf{NUM}(\bar{c})$, set $b_i \leftarrow c$.
   (i) Else, set $b_i \leftarrow$ majority of the $coin_j$'s from the group x, where $x \equiv e(mod \ \lfloor n/g \rfloor)$.

The following properties of the protocol are easily verified: (a) If a player p_i decides at the end of a phase, all players decide by the end of the next phase. (b) If a player sets $b_i \leftarrow c$ at the end of a phase (instruction h, above), then no player p_j sets $b_j \leftarrow \bar{c}$. Given this, it is easy to see that agreement is reached when all the remaining players (ones who set b_i to be the coin-toss from a group) set b_i to c (in instruction i). It remains to analyze the expected number of rounds in which this event happens.

Set the size of a group to be $g = 2m = \log n$. Call a group e *good* if more than $m+1$ players in the group are

non-faulty. Call a coin-toss good if at least $m+1$ good players in a group tossed the same coin (with a fixed value – 0 or 1).

$$\Pr[\text{group } e\text{'s coin is good} \mid e \text{ is a good group}] \geq \frac{1}{2^{m+1}}.$$

Now, lets analyze how many bad groups there can be. There are at most $t < (\frac{1}{2} - \epsilon)n$ players who have no randomness, and these players can make at most $\frac{t}{m+1} < (\frac{1}{2} - \epsilon)\frac{2n}{\log n} = (1-2\epsilon)\frac{n}{\log n}$ groups bad. Since there are $\frac{n}{\log n}$ groups in total, the number of good groups is at least $\frac{2\epsilon n}{\log n}$.

The protocol terminates as soon as there is a good coin-toss. The expected number of good groups that have to toss coins before they get a good coin is precisely $2^{m+1} \leq 2\sqrt{n}$. The probability that a good coin is not formed after $n^{3/4}$ groups tossing coins is negligible, by a Chernoff Bound. Thus, the expected number of rounds to each agreement is $\frac{2t}{\log n} + n^{3/4} + O(1)$.

Theorem 6 (Asynchronous Network) *If $n \geq 3t+1$, then there exist BA protocols that tolerate t faults in an asynchronous network, when the players have (k, δ) block-sources with $\delta > \frac{1}{2}$, and*

- *run in $O(1)$ rounds, with private channels, and*
- *run in $O(2^n)$ rounds, with non-private channels.*

Proof

In the private channels case: In the first round, the players run the protocol Asynch-Extract with a $(t+1, t)$-immune extractor in the place of Ext. Let R_i denote the output of player i after running Asynch-Extract. Now, the players run the $O(1)$-round BA protocol of [5], with player i using R_i as the randomness to the [5] protocol.

Each player p_i gets $t+1$ strings, eventually. This is because $n \geq 2t+1$ and there are at most t faulty players. At least one of the $t+1$ strings is "good". i.e, it comes from a (k, δ) block-source which is independent from p_i's source. By the $(t+1, t)$-immunity of Ext, this means that the output R_i of player i is ϵ-close to uniform. Further, the output R_i of p_i is private, informally because one of the inputs to Ext is unknown to the faulty players.

In the non-private channels case: The players run the protocol Asynch-Extract with a $(t+1, t)$-strongly immune strong extractor in the place of Ext.

5.1.1 Discussion and Future Work

An intriguing question is whether one can achieve a general simulation of an arbitrary distributed algorithm, even when one is given weak sources. In this paper, we answer this question for some specific type of distributed algorithms (such as the best known Byzantine Agreement protocols in some settings).

Models of randomness other than what we chose to focus on in this paper may have been assumed. The one we find particularly appealing is where each player has a weak random source, but the sources are *correlated*. Namely, the only guarantee is that the randomness sampled by player i has a large min-entropy even conditioned on the values for random strings sampled by all other players. The model considered in this paper is a first approximation to this more general model. Note that the correlation is adversarial and cannot be used to our advantage in any way.

Acknowledgments. The authors wish to acknowledge initial conversations on this topic with Adi Akavia and Oded Goldreich.

References

1. Boaz Barak, Russell Impagliazzo, and Avi Wigderson. Extracting randomness using few independent sources. In *FOCS*, pages 384–393, 2004.
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, Elan Pavlov, and Vinod Vaikuntanathan. Byzantine agreement in the full-information model in $o(\log n)$ rounds. *unpublished manuscript*.
4. Gabriel Bracha. An asynchronous [(n-1)/3]-resilient consensus protocol. In *PODC*, pages 154–162, 1984.
5. Ran Canetti and Tal Rabin. Fast asynchronous byzantine agreement with optimal resilience. In *STOC*, pages 42–51, 1993.
6. Benny Chor and Brian A. Coan. A simple and efficient randomized byzantine agreement algorithm. *IEEE Trans. Software Eng.*, 11(6):531–539, 1985.
7. Benny Chor and Oded Goldreich. Unbiased bits from sources of weak randomness and probabilistic communication complexity. *FOCS*, pages 429–442, 1985.
8. Yevgeniy Dodis and Roberto Oliveira. On extracting private randomness over a public channel. In *RANDOM-APPROX*, pages 252–263, 2003.
9. Yevgeniy Dodis, Shien Jin Ong, Manoj P, and Amit Sahai. On the (im)possibility of cryptography with imperfect randomness. In *FOCS*, pages 196–205, 2004.
10. Yevgeniy Dodis and Joel Spencer. On the (non)universality of the one-time pad. In *FOCS*, pages 376–, 2002.
11. Cynthia Dwork, David B. Shmoys, and Larry J. Stockmeyer. Flipping persuasively in constant time. *SIAM J. Comput.*, 19(3):472–499, 1990.
12. P. Elias. The efficient construction of an unbiased random sequence. *Ann. Math. Statist.*, 43(3):865–870, 1972.
13. Paul Feldman. Asynchronous byzantine agreement in expected constant number of rounds. *unpublished manuscript*.
14. Pesech Feldman and Silvio Micali. An optimal probabilistic protocol for synchronous byzantine agreement. *SIAM J. Comput.*, 26(4):873–933, 1997.
15. Michael J. Fischer, Nancy A. Lynch, and Mike Paterson. Impossibility of distributed consensus with one faulty process. In *PODS*, pages 1–7, 1983.
16. Michael J. Fischer, Nancy A. Lynch, and Mike Paterson. Impossibility of distributed consensus with one faulty process. *J. ACM*, 32(2):374–382, 1985.
17. Juan A. Garay and Yoram Moses. Fully polynomial byzantine agreement for $>$ processors in $+1$ rounds. *SIAM J. Comput.*, 27(1):247–290, 1998.

18. M. Pease, R. Shostak, and L. Lamport. Reaching agreement in the presence of faults. *Journal of the ACM.*, 27:228–234, 1980.
19. Michael O. Rabin. Randomized byzantine generals. *FOCS*, pages 403–409, 1983.
20. Ran Raz. Extractors with weak random seeds. *STOC*, to appear, 2005.
21. M. Santha and U. V. Vazirani. Generating quasi-random sequences from slightly-random sources. In *FOCS*, pages 434–440, Singer Island, 1984.
22. Umesh V. Vazirani. Towards a strong communication complexity theory or generating quasi-random sequences from two communicating slightly-random sources (extended abstract). In *STOC*, pages 366–378, 1985.
23. J. von Neumann. Various techniques for use in connection with random digits. In *von Neumann's Collected Works*, volume 5, pages 768–770. Pergamon, 1963.
24. David Zuckerman. General weak random sources. In *FOCS 1990*, pages 534–543, 1990.