Distributed Computing with Imperfect Randomness

Shafi Goldwasser*, Madhu Sudan, and Vinod Vaikuntanathan

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

Abstract. Randomness is a critical resource in many computational scenarios, enabling solutions where deterministic ones are elusive or even provably impossible. However, the randomized solutions to these tasks 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 answer this question in the affirmative, for the problem of Byzantine agreement. We construct protocols for Byzantine agreement in a variety of scenarios (synchronous or asynchronous networks, with or without private channels), in which the players have imperfect randomness. Our solutions are essentially as efficient as the best known randomized protocols, despite the defects in the randomness.

1 Introduction

Randomization has proved useful in many areas of computer science including probabilistic algorithms, 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 theoretic community these days is dedicated to de-randomization: the effort of replacing random string by deterministic "random-looking" strings.

The case of using randomness within the field of distributed computing is, in contrast, unambiguous. There are central distributed computing problems for which it is provably impossible to obtain a deterministic solution, whereas efficient randomized solutions exist. The study of one such problem, the *Byzantine Agreement* problem is the focus of this paper.

* Grant Information

1.1 Byzantine Agreement: Randomized versus Deterministic Protocols

The problem of Byzantine Agreement (BA) defined by Pease, Shostak and Lamport [PSL80] 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: (1) *Consistency:* All non-faulty players agree on the same value. (2) *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 BA protocol 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. 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 [FLP83] shows that BA is impossible to achieve even in the presence of a single faulty player.

Yet, Ben-Or [BO83] 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 [Rab83,Bra84,DSS90,FM97,Fel,CR93]) 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 [CC85]).

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 [1], whereas the best asynchronous BA protocol still requires exponential number of rounds [BO83,Bra84].

1.2 What type of Randomness is Available in the Real World?

The common abstraction used to model the use of randomness by a protocol (or an algorithm), is to assume that each participant's algorithm has access to its own source of unbiased and independent coins. However, this abstraction does not seem to be physically realizable. Instead, physical sources are available whose outcome seem only to be "somewhat random".

This gap between available physical sources and the abstract model has been addressed starting with the work of von Neumann [von63] and Elias [Eli72] in

[1] In recent unpublished work, Ben-Or and Pavlov [BOP] show how to improve this to $O(\log n)$ rounds.

the sixties which deal with sources of independent bits of unknown bias. In more recent works, sources of dependent bits were modeled by Santha-Vazirani [SV84], Chor-Goldreich [CG85], and finally Zuckerman [Zuc90] 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* [CG85] 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 output. Namely, whereas a general 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 [SV84,CG85,Zuc90]. Researchers went on to ask (starting with Vazirani [Vaz85]) whether, given two (or more) weak random sources (all independent from each other), extraction of unbiased random bits is possible. Indeed, it was shown [Vaz85,CG85,Zuc90] that two sources suffice. Whereas original works focus on in-principle results, recent work by Barak, Impagliazzo, and Wigderson [BIW04] and others focuses on constructive protocols aimed at efficient extraction from a constant number of independent weak random sources.

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 [CG85,Zuc90] that a single weak random source can be used to replace a perfect source of randomness for any BPP or RP algorithm. Very recently, Dodis et al [DS02,DOPS04], asked the same question about weak random sources within cryptographic protocols. Namely, is it possible for cryptographic appplications (e.g. encryption, digital signatures, secure protocols) to exist in a world where participants each have access to a *single* weak source of randomness? Surprisingly, they show that even if these sources are independent of each other, many cryptographic tasks are *impossible*. These tasks include private-key and public-key encryption, zero knowledge protocols, two-party secure protocols and more.

We thus are faced with a natural and intriguing question in the context of distributed computing: Are weak random sources suffiently strong to replace perfect random sources within randomized distributed computing protocols ? This is the starting point of our research.

1.3 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 works of Dodis et al in the context of cryptographic protocols. Other models may have been assumed. For example, (1) Each player has two or more independent weak random sources. Clearly, BA can be acheived with no additional effort by each player applying known extractors locally to his own independent sources. However, the assumption of local independence is problematic, for the following practical reason: randomness sampled from two sources located close-by are unlikely to be independent of each other. Therefore, it is not reasonable to assume that each player has multiple independent random sources. Our model looks at the other extreme of this setting – we assume that different players are so well-separated geographically that their random strings are independent. However, the most general, and reasonable model is the following: (2) Each player has a weak random source, but those sources are correlated, in 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. We stress that our model is a natural starting point and a close approximation to the real world. We have partial results on the the more general model, which we plan to investigate further in the future.

1.4 Our Results

We focus on the problem of achieving consensus in a complete network of n participants t of which can be malicious faults as defined by[PSL80]. We address both the settings of synchronous and asynchronous networks, and both the cases of private channels (when each pair of participants have a secret communication channel between them) and the case of a full information network (when no secrecy is assumed for any communication between any pair of participants). We note that by the results mentioned above by Dodis et al [DOPS04], making cryptographic assumptions instead of the assumption of physical private channels, is doomed for failure.

The model of randomness that we assume, is for each player in the protocol to have access to a single source of randomness (which is independent from the other players sources).We will show,

1. In the case of *block sources*: how to obtain the best bounds of fault-tolerance and round complexity currently achieved by randomized distributed protocols. Assuming private channels, we show for both synchronous and asynchronous networks an $O(1)$ expected round protocol for $t < \frac{n}{3}$ faults (matching [FM97,CR93]). In the full-information model, we show for synchronous networks an $O(\frac{t}{\log n})$ expected round protocol for $t < \frac{n}{3}$ (matching [CC85]) and a $O(2^n)$ expected round protocol for $t < \frac{n}{3}$ (matching [Bra84]).
2. In the case of *general weak sources*: We assume private channels. For synchronous networks, we show an $O(1)$ expected round protocol for $t < \frac{n}{3}$ faults (matching [FM97]). For asynchronous networks, we get an $O(1)$ expected rounds protocol for $t < \frac{n}{5}$. The question of finding a BA protocol for the full information model where each player has a weak source is left open.

1.5 Our Methods

To achieve our results we build in various ways on top of the existing distributed algorithms [FM97,CC85,BO83,Bra84]. In general, we follow a 2-step *Extract and Simulate* approach to designing such BA protocols. We utilize first $O(1)$ rounds for a pre-processing protocol, in which the parties interact with each other so that at the end, a large number of them obtain a private uniformly random string. The randomness so obtained is used to run existing randomized BA protocols.

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 they have seen sent by honest players (in the case of full information protocols).

As building blocks, we will use the extractors of [Zuc90,CG85,Raz05] as well as the *strong extractors* of [DO03,Raz05]. A strong extractor is one which ensures that the output of the extraction is random even if one is given *some* of the inputs to the extractor. Our 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 the *all* the non-faulty players, and thus one may fear that when running existing randomized BA protocols with only some of the non-faulty players having perfect randomness, 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 [FM97,CC85], 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 case of [FM97], 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 [CC85] it suffices for $(\frac{1}{2}+\delta)n$ (for arbitrarily small constant $\delta > 0$) players to possess good randomness.

1.6 Future Work

Two questions are left open when each player has a general weak source (rather than a block source): (1) How to achieve BA in the full information model, and (2) How to achieve optimal fault-tolerance in the case of asynchronous networks in the private channels model. We currently achieve $O(1)$ rounds for $t < n/5$.

It is of great interest to study the possibility of other tasks in distributed computing, such as leader election and collective coin-flipping, when the players have imperfect randomness. We briefly note that the results in this paper can be used to show the possibility of both these tasks, in the model where the players have independent block sources.

2 Definitions and the Model

2.1 The Network, Communication and Fault Models

We let n denote the total number of players in the system and t the number of faulty players. We consider various models of communication between the players. In all cases, the n players form a fully-connected communication graph. i.e, each player i can send to every other player j a message in one step. 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 publicly visible.

We consider synchronous and asynchronous communication in the network. In the former case, each processor has access to a global clock, and communication is divided into rounds. Messages sent in a round are received in the beginning of the next round, and the network ensures reliable message delivery. 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 consider Byzantine faults in this paper. Byzantine players can deviate arbitrarily from the prescribed protocol, and co-ordinate with each other so as to mislead the good players into disagreement. We *do not* assume that the Byzantine players are computationally bounded. We refer to Byzantine players variously as bad or faulty players, whereas the rest of the players are referred to as non-faulty or good players. The coalition of Byzantine players is informally referred to as the adversary. 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.

2.2 The model of randomness

Each player has his own source of (imperfect) randomness, and the sources of different players generate mutually independent distributions.

Weak Random Sources Let U_k denote the uniform distribution on k bits. If X is a random variable which has a distribution D, then we write $X \sim D$. The distance between distributions is measured in the ℓ_1 norm. Thus, the distance between distributions D_1 and D_2 (denoted by $\Delta(D_1, D_2)$) is $\sum_a |\Pr_{X_1 \sim D_1}[X_1 = a] - \Pr_{X_2 \sim D_2}[X_2 = a]|$. When $\Delta(D_1, D_2) \leq \epsilon$, we say that D_1 and D_2 are ϵ-close.

A source of randomness X of length k is simply a random variable that takes values in $\{0,1\}^k$. If X is not uniformly distributed, we say that the source X is a weak random source. The randomness contained in a source is usually measured in terms of its *min-entropy*. A source X of k bits has min-entropy δk, if for every $a \in \{0,1\}^k$, $\Pr[X = x] \leq 2^{-\delta k}$. In this case, we call X a (k, δ)-source.

Definition 1 **((k,δ)-source).** *A (k,δ)-source (or a (k,δ)-weak source) is a random variable X that takes values in $\{0,1\}^k$ such that for any $x \in \{0,1\}^k$,* $\Pr[X = x] \le 2^{-\delta k}$.

A block source is a sequence of random variables $(X_1, X_2, \ldots)$ such that each X_i (of length k bits) has min-entropy δk, even if conditioned on any realization of the other blocks. This corresponds to sampling multiple times from a source of random bits, when we are guaranteed that each sample has *some* new entropy. Intuitively, the block-length k specifies how often new entropy is guaranteed to be generated by the source.

Definition 2 **((k,δ)-block Source).** *A (k,δ)-block source is a sequence of random variables $X_1, X_2, \ldots$ (each of length k) such that any X_i has a min-entropy of δk conditioned on all the other random variables. That is,* $\Pr[X_i = a_i \mid X_1 = a_1, \ldots, X_{i-1} = a_{i-1}, X_{i+1} = a_{i+1}, \ldots] \le 2^{-\delta k}$.

We use (X, Y) to denote the joint distribution of the random variables X and Y. In particular, (X, U_m) denotes the joint distribution of X and an independent uniform random variable U_m.

Extractors Many real-world applications require uniform randomness. Given a (k,δ)-source X, our first attempt would be to extract "pure randomness" from X. That is, to construct a deterministic function $Ext : \{0,1\}^k \to \{0,1\}^m$ (for some $m > 0$) such that *for any* (k,δ)-source X, $\Delta(Ext(X), U_m)$ is small. But, it is easy to show that this task is impossible in general. Faced with the impossibility of deterministic extraction from a single source, it is natural to ask if one can extract uniform randomness given *two independent* (k,δ)-sources. The answer to this is yes, as was shown by Chor-Goldreich [CG85] for the case when $\delta > \frac{1}{2}$, and more recently by Raz [Raz05] 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 **((k,δ) two-source Extractor).** *A function $Ext : (\{0,1\}^k)^2 \to \{0,1\}^m$ is a (k,δ) two-source extractor if for any (k,δ)-source X_1 and any independent (k,δ)-source X_2, $Ext(X_1, X_2)$ is ϵ-close to U_m.*

A strong two-source extractor is one in which the output of the extractor is independent of each of the inputs separately. More formally,

Definition 4 **((k,δ) two-source strong extractor).** *A function $Ext : (\{0,1\}^k)^2 \to \{0,1\}^m$ is a (k,δ) two-source* strong *extractor if for any (k,δ)-source X_1 and any independent (k,δ)-source X_2, the distribution $(Ext(X_1, X_2), X_i)$ is ϵ-close to the distribution (U_m, X_i) for $i \in \{1,2\}$.*

Dodis and Oliveira [DO03] consider the notion of a super-strong extractor (which is essentially a two-source strong extractor, according to the definition above) and prove that some well-known constructions of two-source deterministic extractors indeed yield two-source *strong* extractors. More recently, Raz [Raz05] 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 a weak random source or if the players have public randomness.

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.

One of the building blocks 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, but all the sources are independent. We call this a (κ, τ)-immune extractor.

Definition 5 **((κ, τ)-immune extractor).** *Let $X_1, X_2, \ldots, X_{\kappa+1}$ be (k, δ)-block sources. A function Ext that takes as input a finite number of blocks from each of the $\kappa + 1$ block sources is called a (κ, τ)-*immune *(k, δ)-extractor if for any* block sources *$X_1, X_2, \ldots, X_{\kappa+1}$ such that (i) X_1 is a (k, δ)-source, (ii) at least $\kappa - \tau$ among the κ sources $X_2, \ldots, X_{\kappa+1}$ are (k, δ) sources, and (iii) all the X_i's are independent, $Ext(X_1, X_2, \ldots, X_{\kappa+1})$ is ϵ-close to U_m.*

In above definition, we are guaranteed that the τ "bad" sources (those which do not have any randomness) are independent of the $\kappa + 1 - \tau$ "good" sources. We might need to deal with worse situations. In particular, the τ bad sources could be **dependent** on some of the "good" sources. A (κ, τ)-strongly immune extractor extracts uniform randomness even in this adversarial situation.

Definition 6 **((κ, τ)-strongly-immune extractor).** *Let $X_1, X_2, \ldots, X_{\kappa+1}$ be (k, δ)-block sources. A function Ext that takes as input a finite number of blocks from each of the $\kappa + 1$ block sources is called a (κ, τ)-*strongly-immune *(k, δ)-extractor if for any* block sources *$X_1, X_2, \ldots, X_{\kappa+1}$ such that (i) X_1 is a (k, δ)-source independent of all other X_i, and (ii) at least $\kappa - \tau$ among the κ sources $X_2, \ldots, X_{\kappa+1}$ are (k, δ)-sources and are mutually independent, $Ext(X_1, X_2, \ldots, X_{\kappa+1})$ is ϵ-close to U_m.*

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 are publicly visible. In this case, the output of the extraction protocol might depend on the values that were publicly transmitted and is thus not private. Therefore, we need to construct (κ, τ)-strongly immune *strong* extractors to cope with this situation. The constructions are as given below.

I-Ext: A $(t, t-1)$-immune extractor.

– Let Ext be any (k, δ) two-source extractor.
  Let $X_1^2, X_1^3, \ldots, X_1^{t+1}$ denote t distinct blocks of the (k, δ)-block source X_1.
  Let $X_2, \ldots, X_{t+1}$ be one block each from the t other sources.
– I-$Ext(\{X_1^i\}_{i=2}^{t+1}, X_2, \ldots, X_{t+1}) = \bigoplus_{i=2}^{t+1} Ext(X_1^i, X_i)$.

Theorem 1. *I-Ext is a $(t, t-1)$-immune extractor, assuming that Ext is a (k, δ)-two source extractor.*

Proof (Sketch). At least one of the sources (say X_j, $2 \leq j \leq t+1$) has min-entropy δk and X_j is independent of all the X_1^i ($i = 2, \ldots, t+1$). Also, X_1^j has min-entropy δk conditioned on all the blocks $X_1^{j'}$ ($j' \neq j$). That is, the distribution of $(X_1^j | X_1^{j'} = x_1^{j'})$ has min-entropy at least δk. Therefore, $Ext(X_1^j | (X_1^{j'} = x_1^{j'}), X_j)$ is ϵ-close to U_m. Now, consider any $j' \neq j$. The joint distributions $(X_1^j | X_1^{j'}, Y)$ and $(X_1^{j'}, Z)$ are independent. Therefore, $Ext(X_1^{j'}, X_{j'})$ is independent of $Ext(X_1^j, X_j)$, for all $j' \neq j$. This proves that $\bigoplus_{i=2}^{t} Ext(X_1^i, X_i)$ is close to U_m.

We need the following fact, to prove Theorem 2.

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

Theorem 2. *There exists a $(t, t-1)$-strongly immune strong extractor SI-Ext.*

Proof (Sketch). In the construction of I-Ext, using a two-source strong extractor (for instance, those of [DO03,Raz05]) in the place of Ext gives us SI-Ext. We prove the theorem for the case when $t = 2$. The general case follows quite easily from this proof.

Let the distributions under consideration be X_1, Y and Z. Here, the distributions Y and Z could be dependent, but both are independent of X. At least one of Y and Z have min-entropy δk. W.l.o.g, this is Y. Then, since X_1^1 has min-entropy δk conditioned on $X_1^2 = x_1^2$, $Ext((X_1^1 | X_1^2 = x_1^2), Y)$ is ϵ-close to U_m. More importantly, the output of Ext is independent of Y (this is because Ext is a strong extractor).

$$\left[(Ext((X_1^1 | X_1^2 = x_1^2), Y), Y\right] \approx \left[U_m, Y\right]$$

Now, Z is some computable function of Y. Say, $Z = f(Y)$. Then, by Fact 1,

$$\left[(Ext((X_1^1 | X_1^2 = x_1^2), Y), f(Y)\right] \approx \left[U_m, f(Y)\right]$$

$$\big[(Ext((X_1^1|X_1^2 = x_1^2), Y), Z\big] \approx \big[U_m, Z\big]$$

Again, applying Fact 1, and using the fact that Y and X_1^2 are independent, and $(X_1^1|X_1^2 = x_1^2)$ and X_1^2 are independent, we get

$$\big[(Ext((X_1^1|X_1^2 = x_1^2), Y), Ext(X_1^2, Z)\big] \approx \big[U_m, Ext(X_1^2, Z)\big]$$

Thus, $Ext((X_1^1|X_1^2 = x_1^2), Y)$ is ϵ-close to U_m and independent of $Ext(X_1^2, Z)$. Xoring these two values thus gives a value that is ϵ-close to U_m. This proves that SI-Ext is a $(3, 1)$-strongly immune extractor.

Now, we need to prove that SI-Ext is a strong extractor. Let $D_1 \stackrel{def}{=} Ext((X_1^1|X_1^2 = x_1^2), Y)$ and let $D_2 \stackrel{def}{=} Ext(X_1^2, Z)$. We know that

$$\big[D_1, Y\big] \approx \big[U_m, Y\big] \text{ and therefore } \big[D_1, Y, Z\big] \approx \big[U_m, Y, Z\big].$$

This is because $Z = f(Y)$. We also know that $\big[D_1, X_1^2\big] \approx \big[U_m, X_1^2\big]$, since D_1 is the result of extracting from $(X_1^1|X_1^2 = x_1^2)$ and Y, both of which are independent of X_1^2. Since X_1^2 and Z are independent, and so are X_1^2 and Y, $\big[D_1, X_1^2, Y, Z\big] \approx \big[U_m, X_1^2, Y, Z\big]$. Therefore,

$$\big[D_1, X_1^2, Y, Z, Ext(X_1^2, Z)\big] \approx \big[U_m, X_1^2, Y, Z, Ext(X_1^2, Z)\big].$$

Note that the last component of this distribution is precisely D_2. Thus, D_1 is random, given D_2, Y, Z and X_1^2. Thus

$$\big[D_1 \oplus D_2, X_1^2, Y, Z\big] \approx \big[U_m, X_1^2, Y, Z\big].$$

In particular, this means $\big[D_1 \oplus D_2, Y, Z\big] \approx \big[U_m, Y, Z\big]$, which is the definition of the extractor being strong. □

4 Byzantine Agreement Protocols in the case of Block Sources

In this section, we show how to construct randomized Byzantine agreement (BA) protocols that work even when the players have access to block sources (resp. general weak sources), using the extraction protocols of the previous section. Our transformations are fairly generic and they apply to a large class of known randomized BA 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, if the good players have independent block sources.

Protocol Synch-PC-Extract

– 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 an (n, δ) two-source extractor. (Note: Assume for simplicity that n is even. If not, add a dummy player.)
– Each player P_i does the following:
  - 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 $R_i \leftarrow Ext(X_i, X_{i+1})$.
    - Send to P_{i+1} the first $\frac{m}{2}$ bits of R_i and store the remaining bits locally.

Protocol Asynch-Extract

– 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 $R_i \leftarrow Ext(\{X_1^j\}_{j=1}^{t+1}, Y_1, Y_2, \ldots, Y_{t+1})$.

Protocol Synch-FI-Extract

– Group the players $P_1, P_2, \ldots, P_n$ into 4-tuples $(p_1, p_2, p_3, p_4), \ldots, (p_{n-3}, p_{n-2}, p_{n-1}, p_n)$. 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.)
– 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.

Theorem 3 (Synchronous, Private Channels). *If $n \geq 3t+2$, 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 (Sketch). 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 the $O(1)$-round BA protocol of [FM97], with player i using R_i as the randomness to the [FM97] protocol.

Consider the set of all pairs (p_i, p_{i+1}) such that both players in the pair are good. There are at least $\lfloor \frac{n}{2} \rfloor - t \geq \lfloor \frac{t}{2} \rfloor + 1$ such pairs. 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. If both the players p_i and p_{i+1} in a pair are good, then the inputs used to compute R_i are invisible to the adversary. It follows that at least $t+1$ players have private, uniformly random strings. Lemma 1 asserts that, if at least $t+1$ players have private, uniform randomness, then the protocol of [FM97] is an $O(1)$-round BA protocol tolerating any $t < \frac{n}{3}$ faulty players.

Lemma 1. *If $n \geq 3t+1$, then there exists a BA protocol that runs in expected $O(1)$ rounds tolerating t faults in a synchronous network with private channels, even if only $t+1$ (out of $n-t$) good players have private randomness.*

Proof (Sketch). The protocol of Feldman and Micali [FM97] is such a BA protocol. Refer to Appendix A for the BA protocol and the proof of this theorem.

The protocol Asynch-Extract ensures that, *all* the good players get private random strings, at the end of the protocol. This protocol *requires* the players to have *block sources*, informally because the players need multiple samples from the source.

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

Proof (Sketch). 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 [CR93], with player i using R_i as the randomness to the [CR93] 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.

Theorem 5 (Synchronous, Full-Information Model). *If $n \geq 3t+1$, 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 (Sketch). In the first round, the players run the protocol Synch-FI-Extract. Using the randomness so obtained, run the Chor-Coan BA protocol [CC85].

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 pairs. 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 (We need strong immunity of Ext since some of the bad inputs to Ext might be correlated with the good inputs). There are at least $4\lfloor \frac{5t}{12} \rfloor \geq \frac{5}{9}n$ 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, by Lemma 2, if at least $(1+\delta)\frac{n}{2}$ players have private, uniform randomness, then the protocol of [CC85] is an $O(\frac{t}{\log n})$-round BA protocol tolerating any $t < \frac{n}{3}$ faulty players. We are done, since we have $\frac{5}{9}n$ good players who have uniform randomness.

Lemma 2. *If $n \geq 3t + 1$, then 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 $(\frac{1}{2}+\delta)n$ (out of $n-t$) good players have private randomness (for some $\delta > 0$).*

Proof. The protocol of Chor and Coan [CC85] is such a BA protocol. Refer to Appendix C for the protocol and the proof of this lemma.

Theorem 6 (Asynchronous, Full-Information Model). *If $n \geq 3t+1$, then there exists a BA protocol that runs in expected $O(2^n)$ rounds tolerating t faults, assuming the players are connected by an asynchronous network with non-private channels, and have (k, δ) block-sources with $\delta > \frac{1}{2}$.*

Proof (Sketch). Analogous to the proof of Theorem 4. The players run the protocol Asynch-Extract with a $(t + 1, t)$-strongly immune strong extractor in the place of Ext.

4.1 The Case of General Weak Sources

Theorem 7 (Synchronous, Private Channels). *If $n \geq 3t + 2$, 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 weak sources with min-entropy rate $\delta \geq \frac{1}{2}$.*

Proof (Sketch). The protocol used in the proof of Theorem 3 suffices to prove this. This is informally because, the extractor uses at most one sample from each source.

Theorem 8 (Asynchronous, Private Channels). *If $n \geq 5t + 2$, then there exists a BA protocol that runs in expected $O(1)$ rounds tolerating t faults, assuming the players are connected by an asynchronous network with private channels, and have weak sources with min-entropy rate $\delta \geq \frac{1}{2}$.*

Proof. The protocol used in the proof of Theorem 3, with the following slight modification, suffices to prove this. The change is that, each player, after receiving a string from its partner in a pair, sends a message indicating that the extraction protocol is complete. When player i receives such a message from $n - 2t$ players, he stops the extraction protocol and sets $R_i = \phi$. Each player eventually receives such a message from $n - 2t$ players, since at least $n - 2t$ players are in pairs in which both the players are good. When a player i receives such a message, it knows that at least $n - 4t$ players have indeed extracted uniform randomness. Since $n - 4t \geq t + 1$, we are done.

References

[BIW04] Boaz Barak, Russell Impagliazzo, and Avi Wigderson. Extracting randomness using few independent sources. In *FOCS*, pages 384–393, 2004.

[BO83] Michael Ben-Or. Another advantage of free choice: Completely asynchronous agreement protocols (extended abstract). In *PODC*, pages 27–30, 1983.

[BOP] Michael Ben-Or and Elan Pavlov. Byzantine agreement in the full-information non-adaptive model. *unpublished manuscript*.

[Bra84] Gabriel Bracha. An asynchronous [(n-1)/3]-resilient consensus protocol. In *PODC*, pages 154–162, 1984.

[CC85] Benny Chor and Brian A. Coan. A simple and efficient randomized byzantine agreement algorithm. *IEEE Trans. Software Eng.*, 11(6):531–539, 1985.

[CG85] Benny Chor and Oded Goldreich. Unbiased bits from sources of weak randomness and probabilistic communication complexity. *FOCS*, pages 429–442, 1985.

[CR93] Ran Canetti and Tal Rabin. Fast asynchronous byzantine agreement with optimal resilience. In *STOC*, pages 42–51, 1993.

[DO03] Yevgeniy Dodis and Roberto Oliveira. On extracting private randomness over a public channel. In *RANDOM-APPROX*, pages 252–263, 2003.

[DOPS04] 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.

[DS02] Yevgeniy Dodis and Joel Spencer. On the (non)universality of the one-time pad. In *FOCS*, pages 376–, 2002.

[DSS90] Cynthia Dwork, David B. Shmoys, and Larry J. Stockmeyer. Flipping persuasively in constant time. *SIAM J. Comput.*, 19(3):472–499, 1990.

[Eli72] P. Elias. The efficient construction of an unbiased random sequence. *Ann. Math. Statist.*, 43(3):865–870, 1972.

[Fel] Paul Feldman. Asynchronous byzantine agreement in expected constant number of rounds. *unpublished manuscript*.

[FLP83] Michael J. Fischer, Nancy A. Lynch, and Mike Paterson. Impossibility of distributed consensus with one faulty process. In *PODS*, pages 1–7, 1983.

[FM97] Pesech Feldman and Silvio Micali. An optimal probabilistic protocol for synchronous byzantine agreement. *SIAM J. Comput.*, 26(4):873–933, 1997.

[PSL80] M. Pease, R. Shostak, and L. Lamport. Reaching agreement in the presence of faults. *Journal of the ACM.*, 27:228–234, 1980.

[Rab83] Michael O. Rabin. Randomized byzantine generals. *FOCS*, pages 403–409, 1983.

[Raz05] Ran Raz. Extractors with weak random seeds. *STOC*, to appear, 2005.

[SV84] M. Santha and U. V. Vazirani. Generating quasi-random sequences from slightly-random sources. In *FOCS*, pages 434–440, Singer Island, 1984.

[Vaz85] 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.

[von63] 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.

[Zuc90] David Zuckerman. General weak random sources. In *FOCS 1990*, pages 534–543, 1990.

Appendix A – Proof of Lemma 1

We describe the salient features of Feldman-Micali protocol for Byzantine Agreement in a network with private channels, tolerating $t < \frac{n}{3}$ faulty players.

The protocol consists of three building blocks: a graded broadcast protocol, an $n/3$-resilient verifiable secret-sharing protocol and an oblivious common-coin protocol. The graded broadcast protocol is deterministic.

The graded VSS protocol of Feldman-Micali [FM97] has the property that the Sharing protocol is randomized, and the only instructions in Share are executed by the dealer h. The share-verification and recovery are deterministic. It follows that as long as the dealer has uniform randomness, P is a graded VSS protocol.

Lemma 3. *If $n \geq 3t + 1$, there exists an $O(1)$-round protocol, which is a t-resilient graded Verifiable Secret-Sharing (VSS) protocol,* assuming only that the dealer has randomness.

Definition 7 (Oblivious Common Coin). *Let P be a fixed-round protocol in which each player x has no input and is instructed to output a bit r_x. We say that P is an oblivious common coin protocol with fairness p and fault-tolerance t if for all bits b, for every set of t players who are corrupted,* $\Pr[\forall$ *good players* i, $r_i = b] \geq p$. *We refer to an execution of P as a coin. The coin is unanimously good if $r_i = b$ for every good player i.*

Lemma 4. *If $n \geq 3t+1$ and if there exists an $O(1)$-round graded VSS protocol that tolerates t faults and which works assuming only that the dealer has randomness, then there exists an $O(1)$-round oblivious coin protocol, which assumes only that $t+1$ good players have randomness.*

So far, all the subprotocols (Gradecast and Graded VSS) required at most one of the participating players to have uniform randomness. But, the common coin protocol *requires* that $t+1$ players have uniform randomness. The lemma shows that $t+1$ is a tight bound.

Proof. The Oblivious Coin protocol of Feldman-Micali [FM97] is given below. The proof that it is enough for $t+1$ players to have randomness, follows.

<u>Protocol Oblivious Common Coin</u>

1. (for every player i): For $j = 1, \ldots, n$, randomly and independently choose a value $s_{ij} \in [0, \ldots, n-1]$.
   (Note: We will refer to s_{ij} as the secret assigned to j by i.)
   Concurrently run Share and Graded-Verify (of a VSS protocol) n^2 times, once for each pair $(h, j) \in [1 \ldots n]^2$, wherein h acts as a dealer, and shares s_{hj}, the secret assigned by h to j.
   Let $verification_i^{hj}$ be player i's output at the end of Graded-Verify for the execution labeled (h, j).
   (Note: Informally, $verification_i^{hj}$ indicates what player i thinks about the recoverability of the secret assigned by h to j.)
2. (for every player i): Gradecast the value $(verification_i^{1i}, verification_i^{2i}, \ldots, verification_i^{ni})$.
   (Note: The gradecasted value by player i is precisely player i's opinion about the recoverability of the secret assigned by all the other players to itself.)
3. (for every player i): for all j, if
   (a) in the last step, you have accepted player j's gradecast of a vector $\boldsymbol{e_j} \in \{0, 1, 2\}^n$,
   (b) for all h, $|verification_i^{hj} - \boldsymbol{e_j}[h]| \leq 1$, and
   (c) $\boldsymbol{e_j}[h] = 2$ for at least $n - t$ values of h,

   then set $player_{ij} = ok$, else set $player_{ij} = bad$.
4. (for every player i): Recover all possible secrets.
   Concurrently run Recover on all the n^2 secrets shared. Denote by $value_i^{hj}$ your output for execution (h, j). If $player_{ij} = bad$, set $SUM_{ij} = bad$, else set
   $$SUM_{ij} = \{ \sum_{h \text{ such that } \boldsymbol{e_j}[h]=2} value_i^{hj} \} \ mod\ n.$$
   If for some player j, $SUM_{ij} = 0$, output $r_i = 0$, else output $r_i = 1$.

We now sketch the proof that the above protocol is an Oblivious Common Coin protocol, assuming at least $t+1$ good players have uniform randomness, and at most t players are faulty. This follows from the following series of observations.

– In step $3(a)$ of the protocol, all the good players that accept the gradecast of a player i receive the same vector $\boldsymbol{e_i}$, even if it player i is bad. This means, every good player i computes SUM_{ij} as a sum of the same set of values.
– If SUM_{ij} is not set to bad, all the addenda of SUM_{ij} had $\boldsymbol{e_j}[h] = 2$, which means $verification_i^{hj} \geq 1$ (by Step 3(b) of the above protocol). This in turn means, by the property of graded VSS, that there is a unique secret corresponding to the $(h, j)^{th}$ execution of Share, which can be recovered. Thus, for every player j (who may be malicious), there exists a value γ such that, for any good player i, SUM_{ij} is either γ or bad.
– Moreover, if $SUM_{ij} \neq bad$, then SUM_{ij} is a sum of at least $n - t$ values (by Step 3(c) of the above protocol). *At least one of the $n - t$ values is shared by a good player who has randomness (since there are at least $t + 1$ of them).*

Thus, since all the values shared are independent[2], SUM_{ij} is either set to *bad* or a random number γ.
– Given this, we can prove that the coin is sufficiently common and sufficiently random. The proof proceeds identically to that of [FM97]. More precisely, we can prove that for any bit b, $\Pr[\forall \text{ good players } i,\ r_i = b] \geq min(e^{-1}, 1 - e^{-2/3})$.

Lemma 5 ([FM97]). *Given an oblivious-coin protocol as a black-box, there is a deterministic protocol that achieves BA in $O(1)$ rounds.*

Appendix B – Proof of Lemma 2

We describe the Chor-Coan protocol for Byzantine Agreement in a full-information network, tolerating $t < \frac{n}{3}$ faulty players.

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. A more formal description of the protocol follows:
Note: When the protocol begins, each player p_i has an input bit b_i.

1. For $e = 1$ *to* ∞, each player p_i does the following:
   Comment: e is the current phase.
   (a) Sends to every player the message $(e, \mathsf{Phase}_1, b_i)$.
   (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),
      i. then set $b_i \leftarrow v$,
      ii. else set $b_i \leftarrow$ "?"
   (d) If $\mathsf{GROUP}(p_i) \equiv e \ (mod\ \lfloor \frac{n}{g} \rfloor)$
      i. then set $coin \leftarrow b$, {Note : b is a random bit}
      ii. else set $coin \leftarrow 0$
   (e) Send to every player the message $(e, \mathsf{Phase}_2, b_i, coin)$.
   (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 $x \equiv e(mod\ \lfloor n/g \rfloor)$.

Proof of Lemma 2. 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

[2] It turns out that this statement is not precise, and has to be proven by a more careful simulation argument, for which we refer the reader to [FM97].

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). It is clear that

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

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 $\frac{2t}{\log n} + n^{3/4} + O(1)$.