Shared vs Private Randomness in Distributed Interactive Proofs

Pedro Montealegre[1]
Facultad de Ingeniería y Ciencias, Universidad Adolfo Ibáñez, Santiago, Chile
p.montealegre@uai.cl

Diego Ramírez-Romero
Departamento de Ingeniería Matemática, Universidad de Chile, Santiago, Chile
dramirez@dim.uchile.cl

Ivan Rapaport
DIM-CMM (UMI 2807 CNRS), Universidad de Chile, Santiago, Chile
rapaport@dim.uchile.cl

Abstract

In distributed interactive proofs, the nodes of a graph G interact with a powerful but untrustable prover who tries to convince them, in a small number of rounds and through short messages, that G satisfies some property. This series of interactions is followed by a phase of distributed verification, which may be either deterministic or randomized, where nodes exchange messages with their neighbors.

The nature of this last verification round defines the two types of interactive protocols. We say that the protocol is of Arthur-Merlin type if the verification round is deterministic. We say that the protocol is of Merlin-Arthur type if, in the verification round, the nodes are allowed to use a fresh set of random bits.

In the original model introduced by Kol, Oshman, and Saxena [PODC 2018], the randomness was private in the sense that each node had only access to an individual source of random coins. Crescenzi, Fraigniaud, and Paz [DISC 2019] initiated the study of the impact of shared randomness (the situation where the coin tosses are visible to all nodes) in the distributed interactive model.

In this work, we continue that research line by showing that the impact of the two forms of randomness is very different depending on whether we are considering Arthur-Merlin protocols or Merlin-Arthur protocols. While private randomness gives more power to the first type of protocols, shared randomness provides more power to the second. Our results also connect shared randomness in distributed interactive proofs with distributed verification, and new lower bounds are obtained.

2012 ACM Subject Classification Theory of computation → Distributed computing models; Theory of computation → Interactive proof systems; Theory of computation → Distributed algorithms

Keywords and phrases Distributed interactive proofs, Distributed verification, Shared randomness, Private randomness

Digital Object Identifier 10.4230/LIPIcs.ISAAC.2020.51

Related Version A full version of the paper is available at `https://arxiv.org/abs/2006.16191`.

Funding Partially supported by CONICYT via PIA/ Apoyo a Centros Científicos y Tecnológicos de Excelencia AFB 170001 (P.M. and I.R.), FONDECYT 1170021 (D.R. and I.R.), FONDECYT 11190482 (P.M.) and PAI + Convocatoria Nacional Subvención a la Incorporación en la Academia Año 2017 + PAI77170068 (P.M.).

[1] Corresponding author

1 Introduction

Distributed decision refers to the task in which the nodes of a connected graph G have to collectively decide whether G satisfies some graph property [19]. For performing any such task, the nodes exchange messages through the edges of G. The input of distributed decision problems may also include labels given to the nodes and/or to the edges of G. For instance, the nodes could decide whether G is properly colored, or decide whether the weight of the minimum spanning tree lies below some threshold.

Acceptance and rejection are defined as follows. If G satisfies the property, then all nodes must accept; otherwise, at least one node must reject [15]. This type of algorithms could be used in distributed fault-tolerant computing, where the nodes, with some regularity, must check whether the current network configuration is in a legal state for some Boolean predicate [9]. Then, if the configuration becomes illegal at some point, the rejecting node(s) raise the alarm or launch a recovery procedure.

Deciding whether a given coloring is proper can be done locally, by exchanging messages between neighbors. These types of properties are called *locally decidable*. Nevertheless, some other properties, such as deciding whether G is a tree, are not. As a remedy, the notion of *proof-labeling scheme* (PLS) was introduced [15]. Similar variants were also introduced: non-deterministic local decisions [7], locally checkable proofs [12], and others.

Roughly speaking, in all these models, a powerful prover gives to every node v a certificate $c(v)$. This provides G with a global distributed-proof. Then, every node v performs a local verification using its local information together with $c(v)$. PLS can be seen as a distributed counterpart to the class NP, where, thanks to nondeterminism, the power of distributed algorithms increases.

Just as it happened in the centralized framework [10, 11], a natural step forward is to consider a model where the nodes are allowed to have more than one interaction with the prover. In fact, with the rise of the Internet, prover-assisted computing models are more relevant than ever. We can think of asymmetric applications like Facebook, where, together with the social network itself, there is a very powerful central entity that stores a large amount of data (the topology of the network, preferences, and activities of the users, etc.). Or we can consider Cloud Computing, where computationally limited devices delegate costly computations to a cloud with tremendous computational power. The central point lies in the fact that these devices may not trust their cloud service (as it may be malicious, selfish, or buggy). Therefore, the nodes must regularly verify the correctness of the computation performed by the cloud service.

Interestingly, there is no gain when interactions are all deterministic. When there is no randomness, the prover, from the very beginning, has all the information required to simulate the interaction with the nodes. Then, in just one round, he could simply send to each node the transcript of the whole communication, and the nodes simply verify that the transcript is indeed consistent. A completely different situation occurs when the nodes have access to some kind of randomness [2, 9]. In that case, the exact interaction with the nodes is unknown to the prover until the nodes communicate the realization of their random variables. Adding a randomized phase to the non-deterministic phase gives more power to the model [2, 9].

Two model variants arise in this new randomized scenario, regarding the order of the phases. Assume that we have two phases. When the random phase precedes the non-deterministic phase, we refer to *distributed Arthur-Merlin protocols*, and we denote them by dAM (following the terminology and notation of [13]). Conversely, when nodes access randomness only after receiving the certificates, we refer to *distributed Merlin-Arthur protocols*, and we denote them by dMA. Note that Merlin is the powerful but untrustable prover of the PLS model, while Arthur represents the nodes, which are simple and limited verifiers that can flip coins.

In a dMA protocol, the prover does not see the nodes' randomness when choosing the certificates. Instead, only once the prover assigns certificates to the nodes, each node randomly selects a message that broadcasts to its neighbors. Then, each node decides whether to accept or reject, based on its randomness, input, certificate, and the messages it received from its neighbors.

These definitions can be easily extended to a more general setting [5], where the number of interactions between Arthur and Merlin is constant but not fixed to only one interaction per player. This model was introduced in [13] and further studied in [5, 8, 18]. For instance, a dMAM protocol involves three interactions: Merlin provides a certificate to Arthur, then Arthur queries Merlin by sending a random string. Finally, Merlin replies to Arthur's query by sending another certificate. Recall that this series of interactions is followed by a phase of distributed verification performed between every node and its neighbors. When the number of interactions is k we refer to $\mathsf{dAM}[k]$ protocols (if the last player is Merlin) and $\mathsf{dMA}[k]$ protocols (otherwise). For instance, $\mathsf{dAM}[2] = \mathsf{dAM}$, $\mathsf{dMA}[3] = \mathsf{dAMA}$, etc. Also, the scenario of distributed verification, where there is no randomness and only Merlin interacts, corresponds to $\mathsf{dAM}[1]$, which we denote by dM. In other words, dM is the PLS model.

In distributed interactive proofs, Merlin tries to convince the nodes that G satisfies some property in a small number of rounds and through short messages. We say that an algorithm uses $\mathcal{O}(f(n))$ bits if the messages exchanged between the nodes (in the verification round) and also the messages exchanged between the nodes and the prover are upper bounded by $\mathcal{O}(f(n))$. We include this *bandwidth bound* in the notation, which becomes $\mathsf{dMA}[k, f(n)]$ and $\mathsf{dAM}[k, f(n)]$ for the corresponding protocols.

In this article we cope with an important issue, well-studied in the context of communication complexity, but much less considered in distributed computing, related to the visibility of the coins: they can be either shared or private [1, 3, 6, 16, 20]. The theory of distributed decision has restricted itself to private randomness, in the sense that each node has only access to a private source of random coins. These coins are shared with the prover but remain private to the other nodes. We explore the role of *shared randomness*, that is, the situation in which the same set of random bits is produced on every node. The issue of shared randomness in distributed interactive proofs was explicitly formulated by Naor, Parter, and Yogev [18]. It is also expressly addressed in Crescenzi, Fraigniaud, and Paz [5].

For distinguishing the two types of randomness, we denote the private randomness setting by $\mathsf{dAM}^{\mathrm{p}}[k, f(n)]$, and the shared randomness setting by $\mathsf{dAM}^{\mathrm{s}}[k, f(n)]$. Also, as explained before, we omit the number of interactions k when they are 2. For instance, we denote $\mathsf{dAM}^{\mathrm{p}}[2, f(n)]$ simply by $\mathsf{dAM}^{\mathrm{p}}[f(n)]$.

Some distributed problems are hard, even when a powerful prover provides the nodes with certificates. It is the case of SYMMETRY, the language of graphs having a non-trivial automorphism (i.e., a non-trivial one-to-one mapping from the set of nodes to itself preserving edges). Any proof labelling scheme recognizing SYMMETRY requires certificates of size $\Omega(n^2)$ [12].

Many problems requiring $\Omega(n^2)$-bit certificates in any PLS, such as SYMMETRY, admit distributed interactive protocols with small certificates, and very few interactions. In fact, SYMMETRY is in both $\mathsf{dMAM}^{\mathrm{p}}[\log n]$ and $\mathsf{dAM}^{\mathrm{p}}[n \log n]$ [13]. Moreover, $\overline{\text{SYMMETRY}}$ (i.e. the languages of graphs not having a non-trivial automorphism) belongs to $\mathsf{dAMAM}^{\mathrm{p}}[\log n]$ [18].

In [5], the authors explore the role of shared randomness in distributed interactive proofs. They prove that private randomness does not limit the power of Arthur-Merlin protocols compared to shared randomness, up to a small additive factor in the certificate size. Roughly, they show that, if $\mathcal{L} \in \mathsf{dAM}^{\mathrm{s}}[k, f(n)]$, then $\mathcal{L} \in \mathsf{dAM}^{\mathrm{p}}[k, f(n) + \log n]$.

We deepen this study by finding explicit inclusions and separations between models.

1.1 Our Results

In Section 3 we show that any interactive protocol using shared randomness can be derandomized into a non-interactive proof, with an exponential-factor overhead in the bandwidth. Roughly, we prove that, if $\mathcal{L} \in \mathsf{dAM}^{\mathrm{s}}[k, f(n)]$, then $\mathcal{L} \in \mathsf{dM}(2^{O(k\, f(n))} + \log n)$. From this we conclude many lower bounds. For instance, we can conclude that SYMMETRY $\in \mathsf{dAM}^{\mathrm{s}}[k, \Omega(\log n)]$, for any fixed k. This result is tight, because it is already known that SYMMETRY $\in \mathsf{dMAM}^{\mathrm{s}}[\log n]$ (in fact, it is known that SYMMETRY $\in \mathsf{dMAM}^{\mathrm{p}}[\log n]$ [13], but the private coin protocol can be easily adapted to work with shared randomness).

Later, in Section 4, we separate the models with private and shared randomness through the language AMOS, which is the language of labeled graphs having at most one selected node. More precisely, AMOS is the language of n-node graphs with labels in $\{0, 1\}$, and where at most one vertex is labeled 1. In [8] it is shown AMOS is *easy* for private-coin Arthur-Merlin protocols, as AMOS $\in \mathsf{dAM}^{\mathrm{p}}[1]$. We prove that AMOS $\in \mathsf{dAM}^{\mathrm{s}}[k, \Theta(\log\log n)]$ and hence there exists an unbounded gap between the two models.

Interestingly, regarding private and shared randomness, roles are reversed when we address dMA protocols instead of dAM protocols. In fact, in Section 5, we get an analogous result to that in [5] by proving that dMA protocols with shared randomness are more powerful than dMA protocols with private randomness. More precisely, if $\mathcal{L} \in \mathsf{dMA}^{\mathrm{p}}_{\varepsilon}[f(n)]$, then $\mathcal{L} \in \mathsf{dAM}^{\mathrm{s}}_{\varepsilon+\delta}[f(n) + \log n + \log(\delta^{-1})]$. We then separate the two classes. We introduce another language denoted 2-COL-EQ, which consists of graphs with n-bit labels corresponding to proper 2-colorings. In other words, the language consists of bipartite graphs where each part is colored with an n-bit label. We show that 2-COL-EQ separates shared and private randomness on distributed Merlin-Arthur protocols. More precisely, we show first that 2-COL-EQ $\in \mathsf{dMA}^{\mathrm{s}}[\log n]$. Then, we show that, for $\varepsilon < 1/4$, 2-COL-EQ $\in \mathsf{dAM}^{\mathrm{p}}_{\varepsilon}[\Theta(\sqrt{n})]$.

1.2 Related Work

The study of the role of shared and private randomness in distributed interactive proofs was initiated very recently [5]. With respect to the case $\mathsf{dAM}^{\mathrm{s}}[2] = \mathsf{dAM}^{\mathrm{s}}$, the authors show that any Arthur-Merlin protocol for both SYMMETRY and $\overline{\text{SYMMETRY}}$ must have certificates and messages of size $\Omega(\log\log n)$. Note that this is stronger than just saying SYMMETRY, $\overline{\text{SYMMETRY}} \notin \mathsf{dAM}^{\mathrm{s}}(o(\log\log n))$. On the positive side, in [5] the authors show that, in the $\mathsf{dMA}^{\mathrm{s}}$ model, shared randomness helps significantly if we want to decide whether a graph has no triangles. In fact, the language of triangle-free graphs belongs to $\mathsf{dMA}^{\mathrm{s}}[\sqrt{n}\log n]$ while any PLS requires certificates of size $n/e^{\mathcal{O}(\sqrt{\log n})}$.

By contrast, the issue of private versus shared randomness has been intensively addressed in the communication complexity framework. More precisely, in the Simultaneous Messages Model (SM). This two-player model was already present in Yao's seminal communication complexity paper of 1979 [21].

In the SM model, the two parties are unable to communicate with each other, but, instead, can send a single message to a referee. Yao proved that the message size complexity of EQ, which tests whether two n-bit inputs are equal, is $\Theta(n)$ in the deterministic case (in fact he proved that this is also true even if players can communicate back-and-forth). Later, clear separations have been proved between deterministic, private randomness, and shared randomness algorithms. In the shared randomness setting with constant one-sided error, the message size complexity of EQ is $\mathcal{O}(1)$ [1]. On the other hand, for private randomness algorithms of constant one-sided error, the message size complexity is much higher, $\Theta(\sqrt{n})$ [1, 20]. More generally, Babai and Kimmel [1] proved that, for any function f, the use of private randomness in simultaneous messages might lead to at most a square root improvement.

There are natural ways to extend the SM model to more than two players. This issue is addressed in [6] in the context of the *number-in-hand* model (where each player only knows its own input, there is no input graph G and players broadcast messages in each round). In problem AllEq there are k players, each one receives a boolean vector $\{0,1\}^n$, and they have to decide whether all the k vectors are equal. In problem ExistsEq, the k players have to decide whether there exist *at least* two players with the same input. It is not difficult to see that in both the deterministic case and the shared randomness case, the results for two players can be extended to k players (the number of players is irrelevant). The private coin case is more involved than the case of shared randomness. With respect to private coin algorithms of constant error, the authors prove, for problem AllEq, an upper bound of $\mathcal{O}(\sqrt{n/k} + \log(\min(n,k)))$ and a lower bound of $\Omega(\log n)$. In the case of ExistsEq the upper bound they show is $\mathcal{O}(\log k\sqrt{n})$ while the lower bound is $\Omega(\sqrt{n})$.

2 Model and Definitions

Let G be a simple connected n-node graph, let $I : V(G) \to \{0,1\}^*$ be an input function assigning labels to the nodes of G, where the size of all inputs is polynomially bounded on n. Let $\mathsf{id} : V(G) \to \{1,\dots,\text{poly}(n)\}$ be a one-to-one function assigning identifiers to the nodes. A *distributed language* $\mathcal{L}$ is a (Turing-decidable) collection of triples (G,id,I), called *network configurations*. In this paper, we are particularly interested in two languages. The first one, denoted amos, is the language of graphs where at most one node is selected. The second language, denoted 2-col-eq, consists in graphs with n-bit labels corresponding to proper 2-colorings. Formally,

- $\text{amos} = \Big\{(G,\mathsf{id},I) \mid I : V(G) \to \{0,1\} \text{ and } |\{v \in V(G) : I(v) = 1\}| \leq 1\Big\}$,
- $\text{2-col-eq} = \Big\{(G,\mathsf{id},I) \mid I : V(G) \to \{0,1\}^n \text{ is a proper two-coloring of } G\Big\}$.

Also, we introduce other problems that will be of interest in the following sections: symmetry, diameter, planar, outerplanar, 3-col, spanning tree and $\triangle$-free consisting in, respectively, deciding the existence of a non-trivial automorphism, determining whether the graph has diameter bounded by some threshold, whether the graph is (outer) planar, whether the graph is 3-colorable, whether a set of edges of the graph form a spanning tree, and whether the graph has no triangles (as subgraphs). For simplifying the notation, we denote by $[\![\mathrm{p}(x)]\!]$ the function that equals one iff the proposition $\mathrm{p}(x)$ is true.

A distributed interactive protocol consists of a constant series of interactions between a *prover* called Merlin, and a *verifier* called Arthur. The prover Merlin is centralized, has unlimited computing power and knows the complete configuration (G,id,I). However, he can not be trusted. On the other hand, the verifier Arthur is distributed, represented by the nodes in G, and has limited knowledge. In fact, at each node v, Arthur is initially aware only of his identity $\mathsf{id}(v)$, and his label $I(v)$. He does not know the exact value of n, but he knows that there exists a constant c such that $\mathsf{id}(v) \leq n^c$. Therefore, for instance, if one node v wants to communicate his $\mathsf{id}(v)$ to its neighbors, then the message is of size $\mathcal{O}(\log n)$.

Given any network configuration (G,id,I), the nodes of G must collectively decide whether (G,id,I) belongs to some distributed language $\mathcal{L}$. If this is indeed the case, then all nodes must accept; otherwise, at least one node must reject (with certain probabilities, depending on the precise specifications we are considering).

There are two types of interactive protocols: Arthur-Merlin and Merlin-Arthur. Both types of protocols have two phases: an interactive phase and a verification phase. Let us define first *Arthur-Merlin interactive protocols*. If Arthur is the party that starts the

interactive phase, he picks a random string $r_1(v)$ at each node v of G (this string could be either private or shared) and send them to Merlin. Merlin receives r_1, the collection of these n strings, and provides every node v with a certificate $c_1(v)$ that is a function of v, r_1 and (G, id, I). Then again Arthur picks a random string $r_2(v)$ at each node v of G and sends r_2 to Merlin, who, in his turn, provides every node v with a certificate $c_2(v)$ that is a function of v, r_1, r_2 and (G, id, I). This process continues for a fixed number of rounds. If Merlin is the party that starts the interactive phase, then he provides at the beginning every node v with a certificate $c_0(v)$ that is a function of v and (G, id, I), and the interactive process continues as explained before. In Arthur-Merlin protocols, the process ends with Merlin. More precisely, in the last, k-th round, Merlin provides every node v with a certificate $c_{\lceil k/2 \rceil}(v)$. Then, the verification phase begins. This phase is a one-round deterministic algorithm executed at each node. More precisely, every node v broadcasts a message M_v to its neighbors. This message may depend on $\mathsf{id}(v)$, $I(v)$, all random strings generated by Arthur at v, and all certificates received by v from Merlin. Finally, based on all the knowledge accumulated by v (i.e., its identity, its input label, the generated random strings, the certificates received from Merlin, and all the messages received from its neighbors), the protocol either accepts or rejects at node v. Note that Merlin knows the messages each node broadcasts to its neighbors because there is no randomness in this last verification round.

A *Merlin-Arthur interactive protocols* of k interactions is an Arthur-Merlin protocol with $k-1$ interactions, but where the verification round is randomized. More precisely, Arthur is in charge of the k-th interaction, which includes the verification algorithm. The protocol ends when Arthur picks a random string $r(v)$ at every node v and uses it to perform a (randomized) verification algorithm. In other words, each node v randomly chooses a message M_v from a distribution specified by the protocol, and broadcast M_v to its neighbors. Finally, as explained before, the protocol either accepts or rejects at node v. Note that, in this case, Merlin does not know the messages each node broadcasts to its neighbors (because they are randomly generated). If $k = 1$, a distributed Merlin-Arthur protocol is a (1-round) randomized decision algorithm; if $k = 2$, it can be viewed as the non-deterministic version of randomized decision, etc.

▶ **Definition 1.** *Let $\mathcal{V}$ be a verifier and $\mathcal{M}$ a prover of a distributed interactive proof protocol for languages over graphs of n nodes. If $(\mathcal{V}, \mathcal{M})$ corresponds to an Arthur-Merlin (resp. Merlin Arthur) k-round, $\mathcal{O}(f(n))$ bandwidth protocol, we note $(\mathcal{V}, \mathcal{M}) \in \mathsf{dAM}_{\mathsf{prot}}[k, f(n)]$ (resp. $(\mathcal{V}, \mathcal{M}) \in \mathsf{dMA}_{\mathsf{prot}}[k, f(n)]$).*

▶ **Definition 2.** *Let $\varepsilon \leq 1/3$. The class $\mathsf{dAM}_{\varepsilon}[k, f(n)]$ (resp. $\mathsf{dMA}_{\varepsilon}[k, f(n)]$) is the class of languages $\mathcal{L}$ over graphs of n nodes for which there exists a verifier $\mathcal{V}$ such that, for every configuration (G, id, I) of size n, the two following conditions are satisfied.*

- **Completeness.** *If $(G, \mathsf{id}, I) \in \mathcal{L}$ then, there exists a prover $\mathcal{M}$ such that $(\mathcal{V}, \mathcal{M}) \in \mathsf{dAM}_{\mathsf{prot}}[k, f(n)]$ (resp. $(\mathcal{V}, \mathcal{M}) \in \mathsf{dMA}_{\mathsf{prot}}[k, f(n)]$) and*

$$\mathbf{Pr}\Big[\mathcal{V} \text{ accepts } (G, \mathsf{id}, I) \text{ in every node given } \mathcal{M}\Big] \geq 1 - \varepsilon.$$

- **Soundness.** *If $(G, \mathsf{id}, I) \notin \mathcal{L}$ then, for every prover $\mathcal{M}$ such that $(\mathcal{V}, \mathcal{M}) \in \mathsf{dAM}_{\mathsf{prot}}[k, f(n)]$ (resp. $(\mathcal{V}, \mathcal{M}) \in \mathsf{dMA}_{\mathsf{prot}}[k, f(n)]$),*

$$\mathbf{Pr}\Big[\mathcal{V} \text{ rejects } (G, \mathsf{id}, I) \text{ in at least one nodes given } \mathcal{M}\Big] \geq 1 - \varepsilon.$$

We also denote $\mathsf{dAM}[k, f(n)] = \mathsf{dAM}_{1/3}[k, f(n)]$ and $\mathsf{dMA} = \mathsf{dMA}_{1/3}[k, f(n)]$.

We omit the subindex ε when its value is obvious from the context. For small values of k, instead of writing $\mathsf{dAM}[k, f(n)]$ and $\mathsf{dMA}[k, f(n)]$, we alternate Ms and As. For instance: $\mathsf{dMAM}[f(n)] = \mathsf{dAM}[3, f(n)]$, $\mathsf{dAMA}[f(n)] = \mathsf{dMA}[3, f(n)]$, etc. In particular $\mathsf{dAM}[f(n)] = \mathsf{dAM}[2, f(n)]$, $\mathsf{dMA}[f(n)] = \mathsf{dMA}[2, f(n)]$.

▶ **Definition 3.** *The shared randomness setting may be seen as if all the nodes, in any given round, sent the same random string to Merlin. In order to distinguish between the settings of private randomness and shared randomness, we denote them by* $\mathsf{dAM}^{\mathrm{p}}[k, f(n)]$ *and* $\mathsf{dAM}^{\mathrm{s}}[k, f(n)]$*, respectively.*

2.1 Simultaneous Messages Model

In the simultaneous messages model (SM) there are three players, *Alice*, *Bob* and a *referee*, who jointly want to compute a function $f(x, y)$. Alice and Bob are given inputs x and y, respectively. The referee has no input. Alice and Bob are unable to communicate with each other, but, instead, are able to send a single message to the referee. Their messages depend on their inputs and a number of random bits. Then, using only the messages of Alice and Bob and eventually another random string, the referee has to output $f(x, y)$ (up to some error probability ε, given by the coins of Alice, Bob, and the referee). A randomized protocol with error ε is *correct* in the SM model if the answer is correct with probability at least $1 - \varepsilon$.

We are only interested in the SM model with *private coins*, i.e., when the random strings generated by Alice, Bob, and the referee are independent. Interestingly, in this model, the power of randomness is very restricted. Indeed, in [1], Babai and Kimmel show that any randomized protocol computing a function f in the SM model using private coins requires messages of size at least the square root of its deterministic complexity. More precisely, if we define the deterministic complexity of f, $\mathsf{D}(f)$, as the size of the messages of an optimal SM deterministic protocol for f, the following proposition holds.

▶ **Proposition 4** ([1], Theorem 1.4)**.** *Let $f : X \times Y \to \{0, 1\}$ be any boolean function. Let $0 \leq \varepsilon < \frac{1}{2}$. Any ε-error SM protocol for solving f using private coins needs the messages to be of size at least $\Omega\left(\sqrt{\mathsf{D}(f)}\right)$.*

By incorporating a prover, we can define interactive proofs in the SM model. More precisely, we define $\mathsf{MA}^{\mathrm{sym}}$ as follows.

▶ **Definition 5.** *Let $f : X \times Y \to \{0, 1\}$ be a boolean function. We say that $f \in \mathsf{MA}^{\mathrm{sym}}_{\varepsilon}$ if there exists a protocol for Alice and Bob, where:*

- *A fourth player, the prover, provides Alice and Bob with a proof m (which he builds as a function of the input of Alice $x \in X$ and the input of Bob $y \in Y$).*
- *Alice and Bob simultaneously send a message to the referee, that depends on their inputs, their own randomness, and the certificate m provided by the prover. Let $\omega_{x,m}(r)$ be the message sent by Alice given the input x and the seed r and let $\varphi_{y,m}(s)$ be the message sent by Bob, given y and the seed s.*
- *Finally, let $\rho(\omega, \varphi)$ be the random variable indicating the referee's decision given the messages $\omega\varphi$ and its random bits.*

For all $x \in X, y \in Y$, the protocol must satisfy the following:

- **Completeness.** *If $f(x, y) = 1$, there exists a proof m s.t. $\mathbf{Pr}\left(\rho(\omega_{x,m}, \varphi_{y,m} = 1) \geq 1 - \varepsilon\right.$.*
- **Soundness.** *If $f(x, y) = 0$ then, for any proof m, $\mathbf{Pr}(\rho(\omega_{x,m}, \varphi_{y,m}) = 1) < \varepsilon$.*

Let $f : X \times Y \to \{0,1\}$ be a boolean function. The cost of an $\mathsf{MA}^{\mathrm{sym}}$ protocol that solves f is the sum of the proof size, along with the maximum size of a message considering all possible random bits. When there is no randomness we recover the classical definition of non-deterministic complexity in the SM model, which we denote by $\mathsf{M}^{\mathrm{sym}}(f)$.

▶ Remark 6. The assumption that both Alice and Bob receive the same proof does not affect the definition of the class: in case that Alice receives m_a and Bob receives m_b as proofs, then Merlin may concatenate $m_a m_b$ and then Alice and Bob just consider their part of the message (the referee verifies that Alice and Bob received, indeed, the same message).

3 The Limits of Shared Randomness

In this section we show that the largest possible gap between non-interactive proofs and interactive proofs with shared randomness is exponential. More precisely, we show that any interactive protocol using shared randomness can be derandomized into a non-interactive proof, with an exponential-factor overhead in the bandwidth. From this result we can obtain lower bounds, some of them even tight, for the bandwidth of interactive-proofs with shared randomness.

▶ **Theorem 7.** *Let $k \geq 1$ and let $\mathcal{L}$ be a language such that $\mathcal{L} \in \mathsf{dAM}^{\mathrm{s}}[k, f(n)]$. Then, $\mathcal{L} \in \mathsf{dM}(2^{O(k\, f(n))} + \log n)$.*

Proof. Let $\mathcal{P}$ be a protocol deciding $\mathcal{L}$ using shared randomness, k rounds of interaction, bandwidth $f(n)$, and with error probability $1/3$. We use $\mathcal{P}$ to define a protocol $\mathcal{P}'$ for $\mathcal{L}$ with only one round of interaction and bandwidth $2^{\mathcal{O}(k \cdot f(n))} + \log n$. Let us fix (G, id, I), an instance of $\mathcal{L}$.

For a prover $\mathcal{M}$ for protocol $\mathcal{P}$, we define a *transcript* of a node $v \in G$ as a k-tuple $\tau(\mathcal{M}, v) = (\tau_1, \tau_2, \ldots, \tau_k)$ such that $\tau_i \in \{0,1\}^{f(n)}$ is a sequence of bits communicated in the i-th round of interaction of $\mathcal{P}$, for each $i \in \{1, \ldots, k\}$. If both k and i are even, then τ_i is a message that $\mathcal{M}$ sends to node v in the i-th interaction. If k is even and i is odd, then τ_i is a random string drawn from the shared randomness. Finally, roles are reversed fin the case where k is odd.

Let us fix $\ell = \lfloor \frac{k}{2} \rfloor$ and let R be the set of all ℓ-tuples $r = (r_1, \ldots, r_\ell)$ such that $r_i \in \{0,1\}^{f(n)}$, for each $i \in \{1, \ldots, \ell\}$. For $v \in G$ and $r \in R$ and a fixed prover $\mathcal{M}$, we call $\tau(\mathcal{M}, v, r)$ the transcript $\tau(\mathcal{M}, v)$ such that $\tau_{2i-1} = r_i$ when k is even and $\tau_{2i} = r_i$ otherwise, for each $i \in \{1, \ldots, \ell\}$. In full words, $\tau(\mathcal{M}, v, r)$ is the transcript of the protocol, when the nodes draw the random strings from r.

We can construct a one-round protocol $\mathcal{P}'$, where the prover sends to each node v the following certificate:

1. A spanning tree T given by the id of a root ρ, the parent of v in the tree, denoted by t_v, and the distance in T from ρ to v, given by d_v.
2. The list $m_v = \{m_r^v\}_{r \in R}$, where $m_r^v \in \{0,1\}^{kf(n)}$ is interpreted as $\tau(\mathcal{M}, v, r)$.
3. A vector $\mathsf{acc}(v) \in \{0,1\}^{|R|}$ where $\mathsf{acc}(v)_r$ indicates that u accept in the transcript given by m_r^u, for all u in the subtree T_v associated to v.

Given the messages received from the prover, the nodes first verify the consistency of the tree given by **(1)**, following the spanning tree protocol given in [15]. Then, each node v checks that for each $r \in R$ the given transcript m_v^r is consistent with r. Then, for each $r \in R$, each node simulates the k rounds of protocol $\mathcal{P}$ using the certificates of its neighborhood, and decide whether to accept or reject. That information is stored in a vector $a^v \in \{0,1\}^{|R|}$. In order to check the consistency of the vector $\mathsf{acc}(v)$, for each $r \in R$ we say that $\mathsf{acc}(v)_r = 1$

if and only if $a_r^v = 1$ and $\mathsf{acc}(u)_r = 1$ for every children u in T_v. If all previous conditions are satisfied and v is not the root, then v accepts. Finally, the root ρ verifies previous conditions and counts the number of accepting entries in $\mathsf{acc}(\rho)$ and accepts if they are at least two-thirds of the total. In any other case, the nodes reject.

The number of bits sent by the prover is: $\mathcal{O}(\log n)$ in **(1)**, $(kf(n)) \cdot 2^{\mathcal{O}(k \cdot f(n))} = 2^{\mathcal{O}(k \cdot f(n))}$ in **(2)** and $2^{\mathcal{O}(k \cdot f(n))}$ in **(3)**. So, in total, the number of bits communicated in any round is $2^{\mathcal{O}(k \cdot f(n))} + \log n$. We now explain the completeness and soundness.

- **Completeness.** If an instance (G, id, I) is in $\mathcal{L}$, an honest prover will send the real answers that each node would have received in the k-round protocol, for which at least two-thirds of the coins all nodes accept, therefore the root accepts.
- **Soundness.** Suppose now that (G, id, I) is not in $\mathcal{L}$, and suppose by contradiction that there exist a prover $\tilde{\mathcal{M}}$ of protocol $\mathcal{P}'$ accepted by all vertices. Let m_v be the certificate that $\tilde{\mathcal{M}}$ gives to vertex v given by **(2)**. Now, let $\hat{\mathcal{M}}$ be a prover of $\mathcal{P}$ such that $\tau(\hat{\mathcal{M}}, v, r) = m_v^r$, for each $r \in R$. Since the root accepts, all nodes must accept two thirds of the transcripts, which contradicts the soundness of $\mathcal{P}$. ◀

A direct consequence of previous result is the transfer of lower bounds from non-determinism to distributed interactive protocols with shared randomness.

▶ **Corollary 8.** *Let $k \geq 1$ and let $\mathcal{L}$ be a language such that $\mathcal{L} \in \mathsf{dM}[\Omega(f(n))]$, where $f(n) = \omega(\log n)$. Then, $\mathcal{L} \in \mathsf{dAM}^{\mathrm{s}}[k, \Omega(\frac{\log f(n)}{k})] = \mathsf{dAM}^{\mathrm{s}}[k, \Omega(\log f(n))]$.*

▶ **Corollary 9.** *Let $k \geq 1$. Then, problems* SYMMETRY, DIAMETER, $\overline{\text{3-COL}}$, $\triangle$-*FREE* $\in \mathsf{dAM}^{\mathrm{s}}[k, \Omega(\log n)]$. *Also,* MST $\in \mathsf{dAM}^{\mathrm{s}}[k, \Omega(\log \log n)]$.

Proof. We just need to apply already known lower bounds: SYMMETRY $\in \mathsf{dM}[\Omega(n)]$ from [12], DIAMETER $\in \mathsf{dM}[\Omega(n)]$ from [4], $\overline{\text{3-COL}} \in \mathsf{dM}[\Omega(n)]$ from [12], $\triangle$-FREE $\in \mathsf{dM}[\Omega(n)]$ from [5], MST $\in \mathsf{dM}[\Omega(\log^2 n)]$ from [14]. ◀

▶ Remark 10. The lower bound saying that SYMMETRY $\in \mathsf{dAM}^{\mathrm{s}}[k, \Omega(\log n)]$ is tight. More precisely, the $\mathsf{dMAM}^{\mathrm{p}}[\log n]$ protocol given by Kol, Oshman and Saxena [13] for solving SYMMETRY can be easily adapted to work with shared randomness. In fact, the protocol is somehow designed in that way, where one particular node generates the random string and shares it with the other nodes (through Merlin). Therefore, SYMMETRY $\in \mathsf{dMAM}^{\mathrm{s}}[\log n]$. On the other hand, SYMMETRY $\in \mathsf{dM}[\Omega(n^2)]$ [12].

In the proof of Theorem 7, in order to design a dM protocol, we had to construct a spanning tree for verifying that two thirds of all coins are accepted by *all nodes*. In fact, it could be the case that, for negative instances, every node rejects a very small portion of the coins, getting the wrong idea that the instance is positive. For avoiding that, and coordinating the nodes, in the dM protocol we construct a spanning tree. This is where the additive $\log n$ term comes from. Next result states that previous situation does not occur if, instead of two thirds, we ask the interactive protocol to accept *with high probabilty*.

▶ **Theorem 11.** *Let $k \geq 1$ and let $\mathcal{L}$ be a language such that $\mathcal{L} \in \mathsf{dAM}^{\mathrm{s}}_{\varepsilon}[k, f(n)]$, with $\varepsilon < \frac{1}{n+1}$. Then, $\mathcal{L} \in \mathsf{dM}[2^{\mathcal{O}(k\, f(n))}]$.*

▶ **Corollary 12.** *Let $k \geq 1$ and let $\mathcal{L}$ be a language such that $\mathcal{L} \in \mathsf{dM}[\Omega(f(n))]$. Then, $\mathcal{L} \in \mathsf{dAM}^{\mathrm{s}}_{\varepsilon}[k, \Omega(\frac{\log f(n)}{k})] = \mathsf{dAM}^{\mathrm{s}}_{\varepsilon}[k, \Omega(\log f(n))]$, with $\varepsilon < \frac{1}{n+1}$.*

▶ **Corollary 13.** *Let $k \geq 1$. Then, problems* PLANAR, OUTERPLANAR, SPANNING-TREE $\in \mathsf{dAM}^{\mathrm{s}}_{\varepsilon}[k, \Omega(\log \log n)]$, *with $\varepsilon < \frac{1}{n+1}$.*

Proof. All these languages belong to $\mathsf{dM}[\log n]$ [12]. ◀

4 dAM^{s} vs dAM^{p}

A recent result shows that dAM protocols with private randomness are more powerful than dAM protocols with shared randomness [5]. The precise result corresponds to next proposition.

▶ **Proposition 14** ([5]). *Let $k \geq 1$, $0 < \varepsilon < \frac{1}{2}$, and $\mathcal{L}$ be a language such that $\mathcal{L} \in \mathsf{dAM}^{s}_{\varepsilon}[k, f(n)]$. Then, $\mathcal{L} \in \mathsf{dAM}^{p}_{\varepsilon}[k, f(n) + \log n]$.*

A natural question is whether the two models are equivalent. In this section we give a negative answer. We separate them through problem AMOS. Recall that AMOS is the language of labeled graphs where at most one node is selected. It is already known that AMOS $\in \mathsf{dM}[\Theta(\log n)]$ [12]. Moreover, in [8] the authors show that adding randomness *after* the nondeterministic round does not help. More precisely, AMOS $\in \mathsf{dMA}^{p}_{\varepsilon}[\Omega(\log n)]$, for $0 < \varepsilon < \frac{1}{5}$.

The situation changes dramatically when randomness goes *before* nondeterminism, as explained in the following proposition.

▶ **Proposition 15** ([8]). *Let $0 < \varepsilon < \frac{1}{2}$. Then,* AMOS $\in \mathsf{dAM}^{p}_{\varepsilon}[\log(\varepsilon^{-1})] = \mathsf{dAM}^{p}_{\varepsilon}[1]$.

In the shared randomness framework, we can construct a protocol that uses bandwidth $O(\log \log n)$. As we are going to see in Theorem 17, this upper bound is indeed tight.

▶ **Lemma 16.** AMOS $\in \mathsf{dAM}^{s}[\log \log n]$.

Proof. The protocol is the following. First, each node considers the smallest prime q such that $\log^{c+2} n \leq q \leq 2 \log^{c+2} n$ and constructs a polynomial over the field $\mathbb{F}_q$ associated to its id given by $p_v(x) = \sum_{i \leq \log(\mathsf{id}(v))} \mathsf{bin}_i(\mathsf{id}(v)) \cdot x^i$. Where $\mathsf{bin}_i(m)$ corresponds to i-th bit in the binary representation of m. All nodes generate a random string $s \in \mathbb{F}_q$ using the shared randomness. Then, the prover sends to each node the random evaluation of the selected node v_0. More precisely, $\bar{p} = p_{v_0}(s)$, which is of size $O(\log \log n)$. The nodes first check if they all received the same value $\bar{p}$. If a node v is not selected, then it always accepts; otherwise, it accepts if and only if $p_v(s) = \bar{p}$. If an instance belongs to AMOS, then all nodes accept. Otherwise, there exist at least two selected nodes u and v. But the probability that $p_v(s) = p_u(s)$ is at most $\frac{1}{\log^c n}$. ◀

From Corollary 12 we conclude that, for every $k \geq 1$, AMOS $\in \mathsf{dAM}^{s}_{\varepsilon}[k, \Omega(\log \log n)]$ with $\varepsilon < \frac{1}{n+1}$. In other words, the protocol given in Lemma 16 matches the lower bound for all correct protocols that run with high probability. Next theorem says that the upper bound is matched even when $\varepsilon = \frac{1}{3}$.

▶ **Theorem 17.** *Let $k \geq 1$. Then,* AMOS $\in \mathsf{dAM}^{s}[k, \Theta(\log \log n)]$.

5 dMA^{s} vs dMA^{p}

In this section we first see that, in what regards private and shared randomness, roles are reversed when we address dMA protocols instead of dAM protocols. In fact, we get a result analogous to that of Crescenzi, Fraigniaud, and Paz [5] (Proposition 14) which says that dMA protocols with shared randomness are more powerful than dMA protocols with private randomness.

▶ **Theorem 18.** *Let $\varepsilon, \delta > 0$ with $\varepsilon + \delta < \frac{1}{2}$ and let $\mathcal{L}$ be a language such that $\mathcal{L} \in \mathsf{dMA}^{p}_{\varepsilon}[f(n)]$. Then, $\mathcal{L} \in \mathsf{dAM}^{s}_{\varepsilon+\delta}[f(n) + \log n + \log(\delta^{-1})]$.*

As we did in previous section for dAM protocols, we are going to give here a negative answer to the question whether $\mathsf{dMA}^{\mathrm{s}}$ and $\mathsf{dMA}^{\mathrm{p}}$ are equivalent models. For obtaining such separation, we use the problem 2-COL-EQ. Recall that this language is the set of network configurations (G, id, I), where I is a function $I : V(G) \to \{0,1\}^n$, such that I is a proper two-coloring of G. In other words, (G, id, I) belongs to 2-COL-EQ if and only if there is a partition $\{V_0, V_1\}$ of $V(G)$, such that both V_0 and V_1 are inependent sets and, for all $v, w \in V_i$, we have that $I(v) = I(w)$, for $i \in \{0,1\}$.

Next lemma says that 2-COL-EQ is "easy" to solve using shared randomness.

▶ **Lemma 19.** 2-COL-EQ $\in \mathsf{dMA}^{\mathrm{s}}[\log n]$.

The goal now is to prove that 2-COL-EQ $\in \mathsf{dMA}^{\mathrm{p}}[\Theta(\sqrt{n})]$. Babai and Kimmel devise a private coin, randomized protocol in the simultaneous messages model (SM) that solves EQUALITY communicating $\mathcal{O}(\sqrt{n})$ bits [1]. Problem EQUALITY consists in deciding whether two n-bit boolean vectors, the inputs of Alice and Bob, are equal.

▶ **Proposition 20** ([1]). *There exists a private coin, randomized protocol in the* SM *model that solves* EQUALITY *using* $\mathcal{O}(\sqrt{n})$ *bits.*

By using the protocol of Babai and Kimel, one can directly construct a $\mathsf{dMA}^{\mathrm{p}}$ protocol for 2-COL-EQ.

▶ **Lemma 21.** 2-COL-EQ $\in \mathsf{dMA}^{\mathrm{p}}[\sqrt{n}]$.

The lower bound is considerably more involved, and we explain it in the next subsection.

5.1 The lower bound

In order to give a lower-bound on the bandwidth of any $\mathsf{dMA}^{\mathrm{p}}$ protocol solving 2-COL-EQ, we show that the result of Babai and Kimmel given by Proposition 4 can be extended to the scenario where Alice and Bob have access to random bits.

▶ **Theorem 22.** *Let* $f : X \times Y \to \{0,1\}$ *be any boolean function. Let* $0 < \varepsilon < \frac{1}{2}$. *Any* ε*-error* $\mathsf{MA}^{\mathrm{sym}}$ *protocol for solving* f *using private coins needs the messages to be of size at least* $\Omega\left(\sqrt{\mathsf{M}^{\mathrm{sym}}(f)}\right)$.

Using previous theorem, we can construct a lower-bound for 2-COL-EQ.

▶ **Lemma 23.** *If* 2-COL-EQ $\in \mathsf{dMA}^{\mathrm{p}}_{\varepsilon}[f(n)]$ *with* $\varepsilon < 1/4$, *then there exists a protocol* $\mathcal{P}$ *solving* EQUALITY *in the* MA^{sym} *model with bandwidth* $\mathcal{O}(f(n))$.

Sketch of the proof of Lemma 23. We show that, any dMA protocol $\mathcal{P}$ for 2-COL-EQ that uses random coins and error probability ε, can be transformed into an $\mathsf{MA}^{\mathrm{sym}}$ protocol $\mathcal{P}^*$ with error probability 3ε. Let $x, y \in \{0,1\}^n$, and assume without loss of generality that n is even. Given $n \in \mathbb{N}$, Alice, Bob and the referee, construct the following network configuration (G, id, I):

- G is a path of $2n+1$ nodes $v_1, \ldots, v_{2n+1}$.
- $\mathsf{id}(v_i) = i$ for each $i \in \{1, \ldots, 2n+1\}$.
- $I(v_i) = \begin{cases} 0^n & \text{if } i \text{ is odd,} \\ x & \text{if } i \text{ is even and } i \leq n, \\ y & \text{if } i \text{ is even and } i > n. \end{cases}$

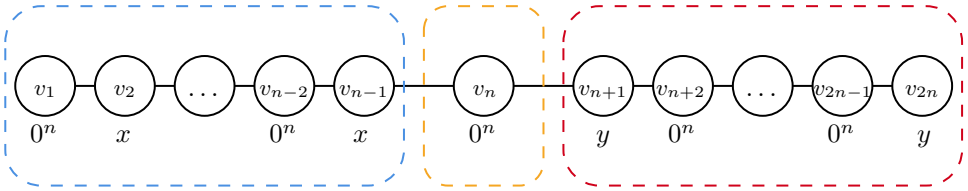

Figure 1 An instance (G, id, I) constructed by Alice and Bob. The blue box corresponds to the set of nodes assigned to Alice, along with input x. Those in the red box are the ones assigned to Bob, along with input y. The orange box contains a single node assigned to the referee, whose input is fixed.

Observe that (G, id, I) is a yes-instance of 2-COL-EQ if and only if (x, y) is a yes-instance of EQUALITY. Given the input x for Alice and y for Bob, the players proceed to construct the instance (G, id, I): Alice takes the first n nodes of G while Bob takes the last n. Finally, the central node is assigned to the referee. For each $v \in G$, let $m(v)$ be the certificate that Merlin sends to node v according to protocol $\mathcal{P}$. In protocol $\mathcal{P}^*$, Alice receives from the prover the certificate $(m(v_n), m(v_{n+1}))$, and Bob receives the certificate $(m(v_{n+1}), m(v_{n+2}))$. Then, the players construct all the possible certificates of the vertices in their side and communicate the most probable output to the referee, together with the messages that nodes v_n and v_{n+2} communicate to node v_{n+1} in protocol $\mathcal{P}$. Using the information received, the referee accepts if Alice, Bob and vertex v_{n+1} accept. ◀

We are now ready to explicitly give the lower bound for 2-COL-EQ.

▶ **Theorem 24.** 2-COL-EQ $\in \mathsf{dMA}^{\mathrm{p}}_{\varepsilon}[\Theta(\sqrt{n})]$ *for any* $\varepsilon < \frac{1}{4}$ *and* 2-COL-EQ $\in \mathsf{dMA}^{\mathrm{s}}_{1/3}[\Theta(\log n)]$.

Proof. In the classic 2-party communication model of Alice and Bob the problem EQUALITY has complexity $\Theta(n)$ even with the help of nondeterminism [17]. This bound translates naturally to the simultaneous messages model, and so $\mathsf{N}(\text{EQUALITY}) = \Theta(n)$. From Theorem 22 we deduce that any protocol in the model $\mathsf{MA}^{\mathsf{sym}}$ for EQUALITY using random bits requires $\Theta(\sqrt{n})$ bits. Now, let $\varepsilon < 1/4$. If there exists a protocol $\mathcal{P}$ for 2-COL-EQ using $o(\sqrt{n})$ bits with error smaller than ε, then, by Lemma 23, there would exist a protocol $\mathcal{P}^*$ for EQUALITY in the model $\mathsf{MA}^{\mathsf{sym}}$ using $o(\sqrt{n})$ bits with error smaller than $1/3$, a contradiction.

Moreover, for every $\varepsilon \leq 1/3$, if 2-COL-EQ belongs to $\mathsf{dMA}^{\mathrm{s}}_{\varepsilon}[f(n)]$ then $f(n) = \Omega(\log n)$, as we can derandomize the protocol and it would contradict the bound for EQUALITY. Thus, by Lemma 19, we conclude that the protocol is tight. ◀

References

1 László Babai and Peter G Kimmel. Randomized simultaneous messages: Solution of a problem of Yao in communication complexity. In *Proceedings of Computational Complexity. Twelfth Annual IEEE Conference*, pages 239–246. IEEE, 1997.

2 Mor Baruch, Pierre Fraigniaud, and Boaz Patt-Shamir. Randomized proof-labeling schemes. In *Proceedings of the 2015 ACM Symposium on Principles of Distributed Computing*, pages 315–324, 2015.

3 Florent Becker, Pedro Montealegre, Ivan Rapaport, and Ioan Todinca. The simultaneous number-in-hand communication model for networks: Private coins, public coins and determinism. In *International Colloquium on Structural Information and Communication Complexity*, pages 83–95. Springer, 2014.

4 Keren Censor-Hillel, Ami Paz, and Mor Perry. Approximate proof-labeling schemes. *Theoretical Computer Science*, 2018.

5 Pierluigi Crescenzi, Pierre Fraigniaud, and Ami Paz. Trade-offs in distributed interactive proofs. In *33rd International Symposium on Distributed Computing (DISC 2019)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2019.

6 Orr Fischer, Rotem Oshman, and Uri Zwick. Public vs. private randomness in simultaneous multi-party communication complexity. In *Proc. of the International Colloquium on Structural Information and Communication Complexity*, volume 9988 of *Lecture Notes in Computer Science*, pages 60–74, 2016.

7 Pierre Fraigniaud, Amos Korman, and David Peleg. Towards a complexity theory for local distributed computing. *Journal of the ACM (JACM)*, 60(5):1–26, 2013.

8 Pierre Fraigniaud, Pedro Montealegre, Rotem Oshman, Ivan Rapaport, and Ioan Todinca. On Distributed Merlin-Arthur Decision Protocols. In *International Colloquium on Structural Information and Communication Complexity*, pages 230–245. Springer, 2019.

9 Pierre Fraigniaud, Boaz Patt-Shamir, and Mor Perry. Randomized proof-labeling schemes. *Distributed Computing*, 32(3):217–234, 2019.

10 Oded Goldreich, Silvio Micali, and Avi Wigderson. Proofs that yield nothing but their validity or all languages in np have zero-knowledge proof systems. *Journal of the ACM (JACM)*, 38(3):690–728, 1991.

11 Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal on computing*, 18(1):186–208, 1989.

12 Mika Göös and Jukka Suomela. Locally checkable proofs in distributed computing. *Theory of Computing*, 12(1):1–33, 2016.

13 Gillat Kol, Rotem Oshman, and Raghuvansh R Saxena. Interactive distributed proofs. In *Proceedings of the 2018 ACM Symposium on Principles of Distributed Computing*, pages 255–264. ACM, 2018.

14 Amos Korman and Shay Kutten. Distributed verification of minimum spanning trees. *Distributed Computing*, 20(4):253–266, 2007.

15 Amos Korman, Shay Kutten, and David Peleg. Proof labeling schemes. *Distributed Computing*, 22(4):215–233, 2010.

16 Ilan Kremer, Noam Nisan, and Dana Ron. On randomized one-round communication complexity. *Computational Complexity*, 8(1):21–49, 1999. doi:10.1007/s000370050018.

17 Eyal Kushilevitz. Communication complexity. In *Advances in Computers*, volume 44, pages 331–360. Elsevier, 1997.

18 Moni Naor, Merav Parte, and Eylon Yogev. The power of distributed verifiers in interactive proofs. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1096–115. SIAM, 2020.

19 Moni Naor and Larry Stockmeyer. What can be computed locally? *SIAM Journal on Computing*, 24(6):1259–1277, 1995.

20 Ilan Newman and Mario Szegedy. Public vs. private coin flips in one round communication games. In *Proc. of the 28th ACM Symposium on Theory of Computing*, STOC '09, pages 561–570, 1996.

21 Andrew Chi-Chih Yao. Some complexity questions related to distributive computing (preliminary report). In *Proceedings of the eleventh annual ACM symposium on Theory of computing*, pages 209–213, 1979.