Randomized Proof-Labeling Schemes

Mor Baruch, Pierre Fraigniaud, Boaz Patt-Shamir

Randomized Proof-Labeling Schemes[*]

Mor Baruch
School of Electrical Engineering
Tel Aviv University
Israel

Pierre Fraigniaud[†]
Dept. of Computer Science
CNRS and Univ. Paris Diderot
France

Boaz Patt-Shamir[‡]
School of Electrical Engineering
Tel Aviv University
Israel

Abstract

A *proof-labeling scheme*, introduced by Korman, Kutten and Peleg [PODC 2005], is a mechanism enabling to certify the legality of a network configuration with respect to a boolean predicate. Such a mechanism finds applications in many frameworks, including the design of fault-tolerant distributed algorithms. In a proof-labeling scheme, the *verification* phase consists of exchanging *labels* between neighbors. The size of these labels depends on the network predicate to be checked. There are predicates requiring large labels, of poly-logarithmic size (e.g., MST), or even polynomial size (e.g., Symmetry). In this paper, we introduce the notion of *randomized* proof-labeling schemes. By reduction from deterministic schemes, we show that randomization enables the amount of communication to be exponentially reduced. As a consequence, we show that checking any network predicate can be done with probability of correctness as close to one as desired by exchanging just a logarithmic number of bits between neighbors. Moreover, we design a novel space lower bound technique that applies to both deterministic and randomized proof-labeling schemes. Using this technique, we establish several tight bounds on the verification complexity of classical distributed computing problems, such as MST construction, and of classical predicates such as acyclicity, connectivity, and cycle length.

[*]Research supported in part by the French-Israeli project MAIMONIDE 31768XL, and by the French-Israeli Laboratory FILOFOCS.
[†]Additional support from the ANR project DISPLEXITY, and from the INRIA project GANG.
[‡]Supported in part by the Israel Science Foundation (grant No. 1444/14).

1 Introduction

Context and Objective. Deciding the validity of a predicate over a distributed system (e.g., whether the nodes are properly colored, or whether the nodes have reached consensus), in a decentralized fashion, is a topic that has recently gained interest for its applications to various domains, including checking the results obtained from the execution of a distributed program [8, 16, 31], designing time lower bounds on the hardness of distributed approximation [11], estimating the complexity of logics required for distributed run-time verification [17], and setting up a complexity theory for distributed computing [15]. In the context of local computing in networks, a distributed decision is typically performed by having each node inspecting its close neighborhood, i.e., inspecting its data, as well as the data of its neighbors in the network, and returning either TRUE or FALSE depending on whether this collection of data is consistent with a legal (global) state of the network. The decision is correct if all nodes return TRUE on legal states, and at least one node returns FALSE on every illegal state. (A node returning FALSE could, e.g., launch a recovery procedure). For instance, deciding the correctness of the predicate stating that the nodes are properly colored is straightforward: every node collects the colors of its neighbors, and returns TRUE if and only if each of these colors is different from its own color.

Not all distributed network predicates can be decided locally. One typical example of such a predicate is "being a spanning tree", since nodes cannot even locally distinguish between a path and a cycle [15]. Nevertheless, in several contexts, it is legitimate to enhance the network with additional information in order to allow nodes to decide locally the correctness of a distributed predicate. This is typically the case of checking the correctness of the output of a distributed program, where, in addition to its own individual output, every node can compute a *certificate* allowing them to verify the correctness of the global output formed by the collection of individual outputs. For instance, in an algorithm computing a spanning tree (i.e., every node is computing the identity $p(v)$ of its parent in the tree), it is sufficient that every node v additionally computes its distance $d(v)$ to the root r in the tree, as well as the identity $\text{id}(r)$ of this root. Indeed, as observed long ago (see, e.g., [7, 23]), verifying in a decentralized fashion whether the set of pointers $\{p(v),\, v \in V\}$ forms a spanning tree of a network $G = (V, E)$ is easy once every node v is provided with the certificate $(\text{id}(r), d(v))$, as follows. Every node v just needs to check that it agrees with its neighbors on $\text{id}(r)$, and that $d(p(v)) = d(v) - 1$. (The root r has $p(r) = \perp$, and just checks $d(r) = 0$). If these tests are passed, node v returns TRUE, otherwise it returns FALSE. This verification procedure satisfies that all nodes return TRUE on a spanning tree with nodes provided with the correct certificates, and at least one node returns FALSE on every 1-factor $\{p(v),\, v \in V\}$ distinct from a spanning tree, whatever the certificates. That is, the procedure cannot be fooled by "fake" certificates on an illegal instance.

The notion of distributed verification such as presented above is well abstracted by the concept of *proof-labeling scheme*, introduced in [31]. A proof-labeling scheme for a predicate $\mathcal{P}$ consists of a pair *prover-verifier*. The prover is an oracle which, for every legal state of the network, assigns a *label* $\ell(v)$ to every node v. The verifier is a distributed algorithm which takes as input for node v the local state of v, its label $\ell(v)$, as well as the label $\ell(w)$ of each of its neighbors w, and returns a boolean value. The proof-labeling scheme is correct for predicate $\mathcal{P}$ if the following two conditions are satisfied: (1) For every legal state, the prover assigns labels to the nodes such that the verifier returns TRUE at every node; (2) For every illegal state, and for every label assignment to the nodes, the verifier returns FALSE in at least one node.

The complexity measure considered for evaluating the quality of a proof-labeling scheme is the *label size*. Indeed, this measure captures the amount of information every node has to transmit to its neighbors for performing the verification. Some predicates can be verified using labels whose sizes are of the same order of magnitude as the size of a node ID, like, e.g., $O(\log n)$-bit labels for spanning tree in n-node networks. However, some other predicates require labels whose sizes are significantly larger than the size of an ID, like, e.g., $\Omega(\log^2 n)$ bits for minimum-weight spanning tree (MST) [29], and even $\Omega(n^2)$ bits for Symmetry (i.e., the existence of a non-trivial automorphism) [21]. For MST, the communication complexity for performing the verification is high, which may be problematic in networks with limited bandwidth. For Symmetry, it is so high that it prevents us from verifying that predicate efficiently under any reasonable model of communication.

The main objective of the paper is to measure by how much communication complexity of verification can be decreased by using randomization. For this purpose, we present and investigate *randomized* proof-labeling schemes. In such a scheme, every node v has access to private random coins. Given its label $\ell(v)$ and its random coins, node v computes a randomized certificate for each of its neighbors. The verifier is a Monte-Carlo algorithm which may err with small probability. More precisely, the verifier at node v takes as input the local state of v, its label $\ell(v)$ and all certificates received from its neighbors. It returns a boolean at each node v according to the following specification, for some arbitrarily small constant $\epsilon \in (0, \frac{1}{2})$ fixed a priori: (1) For every legal state, the prover assigns labels to the nodes such that the probability that the verifier returns TRUE at all nodes is at least $1-\epsilon$; (2) For every illegal state, and for every label assignment to the nodes, the probability that the verifier returns FALSE in at least one node is at least $1 - \epsilon$. We also consider the stronger 1-sided error scenario, in which the scheme is not allowed to err on legal instances, that is: for every legal state, the prover assigns labels to the nodes such that the probability that the verifier returns TRUE at all nodes is 1. On illegal instances, the specifications are identical in the two scenarios with 1-sided or 2-sided error.

Intuitively, in the framework of checking the validity of outputs produced by an algorithm that is not completely trustworthy, or whose outputs may be corrupted somehow, a randomized proof-labeling scheme can be used to make sure that if the output is incorrect, then nodes will collectively be able to detect it. For that purpose, in addition to its outputs which are required to be so that some predicate is satisfied (e.g., the constructed distributed data structure must form a MST), the algorithms must also produce a label at each node, from which certificates are generated randomly. By exchanging the certificates between each pair of neighbors, the randomized proof-labeling scheme guarantees probabilistically to return TRUE everywhere if and only if the outputs are correct w.r.t. the predicate. This guarantee holds even in the face of adversarial labels in the following sense: if the outputs are incorrect, then there is no label assignment that guarantees (probabilistically) that the certificates will be accepted everywhere, while if the outputs are correct, and the labels are according to the specification, then with good probability, all individual verification procedures will accept. In case some node v returns FALSE, which occurs with small probability on a legal instance (or even with probability zero in the 1-sided error case), but with high probability on an illegal instance, then v may launch a recovery procedure or another execution of the algorithm.

Let us make a few remarks before summarizing the main outcome of the paper. First, observe that a randomized proof-labeling scheme does not exchange the labels between the nodes, but only the randomized certificates. It is thus expected that the communication complexity of randomized proof-labeling schemes be significantly reduced compared to the communication complexity of deterministic ones. Second, the choice for ϵ governing the success probability $1 - \epsilon$ has typically very small impact on the ability to design a randomized proof-labeling scheme for a given boolean predicate specifying the legal states of the system. In particular, all the schemes that we design in this paper are oblivious to ϵ, in the sense that ϵ can be chosen as close to 0 as desired by straightforward tunings of the parameters governing our schemes. In term of complexity, the dependency in ϵ only appears in the constants hidden in the big-O and big-Ω notations, for all the schemes presented in the paper. Therefore, for the sake of concreteness, we present our schemes with success guarantee at least $\frac{2}{3}$ (i.e., for $\epsilon = \frac{1}{3}$).

Our Results. We introduce and formalize the concept of *randomized* proof-labeling schemes in the next section. We establish the existence of a 2-sided error generic randomized proof-labeling scheme with certificates on $O(\log n + \log k)$ bits in n-node networks, where $k = k(n)$ is the number of bits used to encode the state of a node (including its identity, its potential input, etc.). Hence, even if the states of the nodes require poly(n) bits, our scheme insures that, by exchanging only $O(\log n)$ bits, every (sequentially decidable) property can be probabilistically verified with success probability at least $\frac{2}{3}$. (In contrast, there are natural properties that require to exchange $\Omega(\text{polylog}(n))$ bits, or even $\Omega(\text{poly}(n))$ bits to be verified deterministically). We show that our logarithmic bound is tight, by exhibiting a property for which any randomized proof-labeling scheme requires certificates on $\Omega(\log n + \log k)$ bits to be verified. In fact, we also prove that, for any property, randomization provides an exponential improvement over deterministic schemes. Indeed, we prove that for any property that can be certified with a deterministic proof-labeling scheme exchanging κ-bit messages, there is a randomized proof-labeling scheme exchanging messages on just $O(\log \kappa)$ bits. An important corollary of this latter result is that there exists a randomized proof-labeling scheme for MST using certificates on just $O(\log \log n)$ bits.

In addition to the above, we provide a general lower bound technique for the certificate size of a proof-labeling scheme. This techniques is based on the novel notion called *graph crosses*. It applies to both deterministic and probabilistic schemes, and generalizes previous specific lower bounds of the literature for deterministic proof-labeling schemes. The probabilistic version applies to any 1-sided error randomized prof-labeling schemes. It also applies to 2-sided error schemes, under some additional constraints regarding the way the certificates are randomly generated by the nodes. Under these assumptions, the upper bound $O(\log \log n)$ bits on the certificate size for MST is tight.

Finally, we consider a set of natural problems and properties (cycle freeness, MST, biconnectivity, etc), and provide randomized proof-labeling schemes for them, with optimal or close-to-optimal certificate sizes. In particular we consider two versions of the longest-cycle-in-graphs problem. We show an upper bound of $O(\log \log n)$ bits, and a lower bound $\Omega(\log \log c)$ bits for the certificate size of a randomized proof-labeling scheme for the predicate cycle-at-least-c. Unless NP $=$ co-NP, there are no proof-labeling schemes for cycle-at-most-c involving $O(\text{poly}(n))$ computation time at each node. In term of communication complexity, we prove lower bounds of $\Omega(\log \frac{n}{c})$ and of $\Omega(\log \log \frac{n}{c})$ bits for the messages involved in proof-labeling schemes and randomized proof-labeling schemes for cycle-at-most-c, respectively.

Related Work. There have been a lot of work on labeling schemes in the past, which can be decomposed into two main streams: *informative labeling schemes*, and *proof-labeling schemes*. In the framework of informative labeling schemes, one is given a function f on pairs (or sets) of nodes, and $(\mathcal{M}, \mathcal{D})$ is an f-labeling scheme for a graph family $\mathcal{F}$ if $\mathcal{M}$, the *marker*, is an algorithm that, given a graph $G = (V, E)$ in $\mathcal{F}$, assigns a label $\ell(v)$ to every $v \in V$, and $\mathcal{D}$, the *decoder*, is an algorithm that satisfies $\mathcal{D}(\ell(u), \ell(v)) = f(u, v)$ for every pair of nodes in G. The main performance criterium is the size of the labels, which should be as small as possible. Since the seminal work of Kannan, Naor, and Rudich [24] on adjacency-labeling, there have been quite a lot of investigations, for a large set of functions f, including the following: adjacency [4, 24], distance [2, 10, 19, 18, 20, 25, 27, 35, 38], connectivity and flow [26, 28], nearest common ancestor [3, 36], etc. In particular, the notion of universal f-matrices for several functions f was introduced in [32], and used to construct upper and lower bounds on the sizes of the corresponding f-labeling schemes. Most investigations related to the design of compact routing tables can also be placed in the framework of informative labeling. This includes, e.g., the papers [12, 13, 39] on routing in trees.

Proof-labeling schemes are not dealing with computing a function, but with verifying a proof that the given instance satisfies some given boolean predicate. This proof is distributed among the nodes under the form of labels assigned to the node by a *prover* which assigns a label to every node. The *verifier* is a distributed algorithm in charge of verifying the distributed proof. As for informative labeling scheme, the main performance criterium is the size of the labels. This concept was introduced by Korman, Kutten, and Peleg in [31]. Among the results that were presented in this paper, it is worth mentioning the $\Theta(\log n)$ bit bound on the verification complexity of acyclicity and the upper bound $O(\log^2 n + \log n \log W)$ bits for MST, where W is the maximal possible weight of an edge. This bound was improved to $O(\log n \log W)$ bits in [29], where a matching lower bound of $\Omega(\log n \log W)$ bits is established for $W > \log n$. It is worth noticing that proof-labeling schemes are closely related to self stabilizing algorithms, that is, algorithms which have to periodically verify the correctness of the system state. See, e.g., [1] where the notion of local detection was introduced and used for designing a self stabilizing protocol constructing a spanning tree, and [30] for another example of using distributed local verification of proofs for the design of self stabilizing algorithms. The reader interested in the tight connections between proof-labeling schemes and self-stabilization is referred to the recent paper [9]. Proof-labeling schemes, where nodes may communicate at distance greater than 1, i.e., may take their individual decision based on the labels of the nodes in their vicinity at distance

$t > 1$, was recently studied in [21]. Finally, distributed decision and verification processes in which the global interpretation of the collection of individual outputs is not restricted to be the logical conjunction of these outputs has been studied in [5, 6].

To our knowledge, there are very few papers dealing with randomization in the framework of informative labeling schemes, or proof-labeling schemes. Randomized informative schemes for trees, including randomized schemes for adjacency and ancestry, were presented in [14]. More recently, [15] provided a framework that could be used for setting up a complexity theory for local distributed computing. This framework includes several complexity classes, including NLD (for non-deterministic local decision) and BPNLD (for bounded probability NLD). The former is a generalization of proof-labeling schemes (with slight differences, including the fact that certificates should be independent of the IDs), and the latter is a randomized version of the former. Nevertheless, in both cases, the emphasis was put on the existence of a proof, and not on its size. In fact, it is proved that all (decidable) languages are in BPNLD, but the proof of this result involves labels as large as $\text{poly}(n)$ bits. In contrast, the randomized proof-labeling schemes described in this paper involve labels of poly-logarithmic size.

2 Model and Definitions

2.1 Computational Framework

A network is modeled as a connected graph $G = (V, E)$, without self-loops or multiple edges. Recall that two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ are *isomorphic* if there exists a bijection $\sigma : V_1 \to V_2$ such that: $\{u, v\} \in E_1 \iff \{\sigma(u), \sigma(v)\} \in E_2$. We assume that the edges incident to a node v are numbered in sequence $1, \ldots, \deg(v)$, where $\deg(v)$ is the *degree* of v. The number of e at v is the *port number* of e at v. An edge may have different port numbers on its two endpoints.

In a *configuration* G_s, we are given a graph $G = (V, E)$, a *state* space S, and a state assignment function $s : V \to S$. The state of a node v, denoted $s(v)$, includes all local input to v. In particular, the state may include the node identity $\text{Id}(v)$ (if the network is not anonymous) and weights of its incidents edges (for edge-weighted networks). The state of v may also include other data like, e.g., the result of an algorithm.

Mechanisms such as proof-labeling scheme involve very simple distributed algorithms, acting in one synchronous round of communication and computation, during which every node sends a value to each of its neighbors, and, upon reception of the values from all its neighbors, every node computes an output. In the context of proof-labeling scheme, this output is a boolean, i.e., either TRUE or FALSE.

Unless specified otherwise, we always assume non-anonymous networks, i.e., every node v is provided with an identity $\text{Id}(v)$, that is part of the state of v. All identities in the same network are pairwise distinct. Nevertheless, the definition of proof-labeling scheme does not need the presence of identities.

2.2 Deterministic and Randomized Proof-Labeling Schemes

We first recall the definition of deterministic proof-labeling schemes (abbreviated PLS henceforth), as introduced in [31]. Given a family $\mathcal{F}$ of configurations, and a boolean predicate $\mathcal{P}$ over $\mathcal{F}$, a PLS for $(\mathcal{F}, \mathcal{P})$ is a mechanism for deciding $\mathcal{P}(G_s)$ for every $G_s \in \mathcal{F}$. A PLS consists of two components: a *prover* **p**, and a *verifier* **v**. The prover is an oracle which, given any configuration $G_s \in \mathcal{F}$, assigns a bit string $\ell(v)$ to every node v, called the *label* of v. The verifier is a decentralized algorithm running concurrently at every node. At each node v, it takes as input the state $s(v)$ of v, its label $\ell(v)$ and the labels of all its neighbors, i.e., the ordered set $\{\ell(w_i) \mid i = 1, \ldots, \deg(v)\}$ where w_i is the neighbor reachable from v through the edge with port number i. The verifier **v** at each node outputs a boolean. If the outputs are TRUE at all nodes, **v** is said to *accept* the configuration, and otherwise (i.e., **v** outputs FALSE in at least one node) **v** is said to *reject* the configuration. For correctness, a proof-labeling scheme (**p**, **v**) for $(\mathcal{F}, \mathcal{P})$ must satisfy the following requirements, for every $G_s \in \mathcal{F}$:

- If $\mathcal{P}(G_s) = \text{TRUE}$ then, using the labels assigned by **p**, the verifier **v** accepts G_s.

- If $\mathcal{P}(G_s) = \text{FALSE}$ then, for every label assignment, the verifier **v** rejects G_s.

The *verification complexity* of a deterministic proof-labeling scheme $(\mathbf{p}, \mathbf{v})$, denoted by κ, is the maximal length of the labels assigned by the prover **p** on a legal configuration $G_s \in \mathcal{F}$ (i.e., a configuration satisfying $\mathcal{P}$).

In this paper we extend the above definition to *randomized proof labeling schemes* (RPLS). The idea is to allow randomization in verification part of the scheme. Specifically, an RPLS is defined as follows. The goal and the prover in an RPLS remain exactly as defined for PLS. However, in an RPLS the verifier **v** has access to a source of independent random bits at each node. At node v, using the label $\ell(v)$ and the private random bits available at v, the verifier produces a random bit string, called *certificate*, for each of its neighbors. The random certificate of v for w_i, the neighbor reachable through port i, is denoted by $c_i(v)$. In an RPLS, *only the certificates are communicated for verification*. More precisely, the input of the verifier at node v consists of its state $s(v)$, its label $\ell(v)$, and all the certificates received from its neighbors, i.e., the collection $\{c_{p_i}(w_i),\ i = 1, \ldots, \deg(v)\}$ where w_i is the neighbor reachable from v through the edge e_i with port number i at v, and p_i is the port number of e_i at w_i. Following the sequential complexity classes, we define two flavors of RPLS. A randomized scheme $(\mathbf{p}, \mathbf{v})$ for $(\mathcal{F}, \mathcal{P})$ must satisfy the following requirements, for every $G_s \in \mathcal{F}$:

- If $\mathcal{P}(G_s) = \text{TRUE}$ then, using the labels assigned by **p**, $\Pr[\mathbf{v} \text{ accepts } G_s] \geq p_{\text{accept}}$.
- If $\mathcal{P}(G_s) = \text{FALSE}$ then, for every label assignment, $\Pr[\mathbf{v} \text{ rejects } G_s] \geq p_{\text{reject}}$.

Following the sequential complexity classes RP and BPP (see, e.g., [34]), we define two flavors of RPLSs: in *one-sided error* RPLS, we have $p_{\text{accept}} = 1$ and $p_{\text{reject}} = \frac{1}{2}$; and in *two-sided error* RPLS, we have $p_{\text{accept}} = p_{\text{reject}} = \frac{2}{3}$.[1] Unless explicitly stated otherwise, in this paper we refer by RPLS to two-sided RPLS.

Clearly, RPLSs give weaker guarantees than deterministic PLSs. The main reason to prefer an RPLS over a PLS is the possible saving in verification complexity, defined next.

Definition 2.1 *The* verification complexity *of a randomized proof-labeling scheme* $(\mathbf{p}, \mathbf{v})$*, denoted by* κ*, is the maximal length of the (random) certificates generated by the (randomized) verifier* **v** *based on the labels assigned to the nodes by the prover* **p** *on a legal configuration* $G_s \in \mathcal{F}$ *(i.e., a configuration satisfying* $\mathcal{P}$*).*

3 Relation to Deterministic Schemes

In this section we show that in RPLS, one can save exponentially in the verification complexity w.r.t. PLS. We start with a reduction of RPLS to PLS.

Theorem 3.1 *Let* $\mathcal{F}$ *be a family of configurations, and let* $\mathcal{P}$ *be a boolean predicate over* $\mathcal{F}$*. If there is a PLS for* $(\mathcal{F}, \mathcal{P})$ *with verification complexity* κ*, then there is an RPLS for* $(\mathcal{F}, \mathcal{P})$ *with verification complexity* $O(\log \kappa)$*.*

The proof of this theorem uses a result about (2-party) communication complexity. In a 2-party communication complexity problem there are two players, Alice and Bob. Alice receives as input a λ-bit string x and Bob receives another λ-bit string y. The goal is for Alice and Bob to compute a certain function $f(x, y)$ by exchanging the smallest possible number of bits. For any two λ-bit strings x and y, let $\text{EQ}(x, y)$ denote the equality predicate. The following fact is well known (see [33]).

Lemma 3.2 *The randomized communication complexity of deciding EQ over* n*-bit strings is* $\Theta(\log n)$*.*

The idea in the proof of Theorem 3.1 is to replicate the PLS label of a node over all its neighbors, use the protocol from Lemma 3.2 to verify the integrity of the replicas, and then apply the verifier of the assumed PLS to the local replicas as if they arrived from the neighbors. See Appendix A for details.

Next, we note the existence of a universal PLS construction summarized in the following lemma (see Appendix B for details).

[1]The choice of $1/2$ and $2/3$ is somewhat arbitrary. The idea is that we can boost the probability of correctness to $1 - \delta$ by repeating the verification procedure $O(\log(1/\delta))$ times independently and outputting the majority of outcomes.

Lemma 3.3 ([21, 31]) *Let $\mathcal{F}$ be a family of configurations with states in S, and let $\mathcal{P}$ be a boolean predicate over $\mathcal{F}$. Assume that every state in S can be represented using $k = k(n)$ bits in n-node networks. There exists a PLS for $(\mathcal{F}, \mathcal{P})$ with verification complexity $O(\min\{n^2, m \log n\} + nk)$.*

Combining Theorem 3.1 and Lemma 3.3 we obtain the following universal result for RPLSs.

Corollary 3.4 *Let $\mathcal{F}$ be a family of configurations with states in S, and let $\mathcal{P}$ be a boolean predicate over $\mathcal{F}$. Assume that every state in S can be represented using $k = k(n)$ bits in n-node networks. There exists a randomized proof-labeling scheme for $(\mathcal{F}, \mathcal{P})$ with verification complexity $O(\log n + \log k)$.*

The upper bound in Corollary 3.4 is tight, as stated below.

Theorem 3.5 *For any function $k(n)$, there exist a family $\mathcal{F}$ of configurations with states on $k = k(n)$ bits in n-node graphs, and a predicate $\mathcal{P}$ over $\mathcal{F}$ such that any randomized proof-labeling scheme for $(\mathcal{F}, \mathcal{P})$ has verification complexity $\Omega(\log n + \log k)$.*

The proof follows from the lower bound on the 2-party communication complexity of EQ stated in Lemma 3.2. Details are provided in Appendix C.

4 Generic Lower Bounds

In this section we formalize a general tool that can be used to prove lower bounds for both deterministic and randomized proof-labeling schemes. The general idea was used many times in the past, most relevantly in [31].

4.1 A General Tool: Edge Crossing

We start with a technical definition, and then define our main concept.

Definition 4.1 *Let $G = (V, E)$ be a graph and let $H_1 = (V_1, E_1)$ and $H_2 = (V_2, E_2)$ be two subgraphs of G. H_1 and H_2 are* independent *if and only if $V_1 \cap V_2 = \emptyset$ and $E \cap (V_1 \times V_2) = \emptyset$.*

The following definition is illustrated in Figure 1.

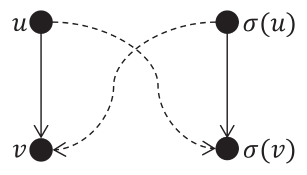

Figure 1: *Crossing two edges under an isomorphism σ. Solid edges are the edges of G, and dashed edged are the edges of $\sigma_{\bowtie}(G)$.*

Definition 4.2 (Crossing) *Let $G = (V, E)$ be a graph, and let $H_1 = (V_1, E_1)$ and $H_2 = (V_2, E_2)$ be two independent isomorphic subgraphs of G with isomorphism $\sigma : V_1 \to V_2$. The* crossing *of G induced by σ, denoted by $\sigma_{\bowtie}(G)$, is the graph obtained from G by replacing every pair of edges $\{u, v\} \in E_1$ and $\{\sigma(u), \sigma(v)\} \in E_2$, by the pair $\{u, \sigma(v)\}$ and $\{\sigma(u), v\}$.*

Crossing can be very useful in proving lower bounds on the verification complexity of both deterministic and randomized proof labeling schemes. We start by showing the simpler case of deterministic PLSs.

Proposition 4.3 *Let $(\mathbf{p}, \mathbf{v})$ be a deterministic PLS for $(\mathcal{F}, \mathcal{P})$ with verification complexity κ. Suppose that there is a configuration $G_s \in \mathcal{F}$ where G contains r pairwise independent isomorphic subgraphs $H_1, \ldots, H_r$ with s edges each, and let $\sigma_i : H_1 \to H_i$ be a port-preserving isomorphism for $i \in \{1, \ldots, r\}$. If $\kappa < \frac{\log r}{2s}$, then there are $1 \le i < j \le r$ such that G_s is accepted by $(\mathbf{p}, \mathbf{v})$ if and only if $(\sigma_j \circ \sigma_i^{-1})_{\bowtie}(G)_s$ is accepted by $(\mathbf{p}, \mathbf{v})$.*

Proof: Let G_s be a configuration as described in the statement. Assume that $\kappa < \frac{\log r}{2s}$, and consider a collection $\{\sigma_i : V(H_1) \to V(H_i), i = 1, \ldots, r\}$ of r port-preserving isomorphisms. Order the nodes of H_1 arbitrarily. This order induces an order on the nodes of H_i thanks to σ_i. For every i, consider the bit-string L_i constructed by concatenating the labels given by $\mathbf{p}$ to the nodes of H_i, in the order induced by σ_i. We have $|L_i| < \log r$ for every i because $|V(H_i)| \leq 2s$, and thus there are less than r possible distinct L_i's in total. Therefore, by the pigeonhole principle, there are $i \neq j$ such that $L_i = L_j$. define $\sigma = \sigma_j \circ \sigma_i^{-1}$, and consider the output of the verifier $\mathbf{v}$ in G_s and in $\sigma_{\bowtie}(G)_s$.

Suppose G_s is accepted by $(\mathbf{p}, \mathbf{v})$, i.e., with the labels provided by $\mathbf{p}$, the verifier $\mathbf{v}$ outputs TRUE at all nodes of G. Therefore, all nodes outside H_i and H_j also output TRUE in $\sigma_{\bowtie}(G)_s$. Now, let v be a node in H_i. Each neighbor w of v from H_i is replaced in $\sigma_{\bowtie}(G)_s$ by the node $\sigma(w)$ from H_j. Since w and $\sigma(w)$ have the same label, the verifier acts the same at v in both G_s and $\sigma_{\bowtie}(G)_s$. Therefore $\mathbf{v}$ outputs TRUE at every node of H_i in $\sigma_{\bowtie}(G)_s$. For the same reason, the verifier acts the same at every node v of H_j, in both G_s and $\sigma_{\bowtie}(G)_s$. Therefore, the verifier also outputs TRUE at all nodes in $\sigma_{\bowtie}(G)_s$, which implies that $\sigma_{\bowtie}(G)_s$ is accepted by $(\mathbf{p}, \mathbf{v})$.

Similarly, if G_s is rejected by $(\mathbf{p}, \mathbf{v})$ then, for any label-assignment to the nodes, the verifier $\mathbf{v}$ outputs FALSE in at least one node v of G. If this node v is not in H_i or H_j, then $\mathbf{v}$ also outputs FALSE at v in $\sigma_{\bowtie}(G)_s$. If this node v belongs to one of the two subgraph H_i or H_j, then, since the verifier acts the same at v in both G_s and $\sigma_{\bowtie}(G)_s$, $\mathbf{v}$ also outputs FALSE at this node in $\sigma_{\bowtie}(G)_s$, which implies that $\sigma_{\bowtie}(G)_s$ is rejected by $(\mathbf{p}, \mathbf{v})$, and the proposition follows. ▌

Proposition 4.3 has the following useful consequence.

Theorem 4.4 *Let $\mathcal{F}$ be a family of configurations, and let $\mathcal{P}$ be a boolean predicate over $\mathcal{F}$. Suppose that there is a configuration $G_s \in \mathcal{F}$ satisfying that (1) G contains as subgraphs r pairwise independent isomorphic copies $H_1, \ldots, H_r$ with s edges each, and (2) there exist r port-preserving isomorphisms $\sigma_i : V(H_1) \to V(H_i)$ such that for every $i \neq j$, the isomorphism $\sigma^{ij} = \sigma_j \circ \sigma_i^{-1}$ satisfies $\mathcal{P}(G_s) \neq \mathcal{P}({\sigma^{ij}}_{\bowtie}(G)_s)$. Then the verification complexity of any proof-labeling scheme for $(\mathcal{F}, \mathcal{P})$ is $\Omega(\frac{\log r}{s})$.*

Remark: Note that Theorem 4.4 cannot yield lower bounds greater than $\Omega(\log n)$, because $r = O(n)$.

4.2 Generic Lower Bounds for Randomized Proof-Labeling Schemes

We now proceed with a generalization of Theorem 4.4 to randomized proof-labeling schemes. First we define *edge-independent* RPLSs.

Definition 4.5 *An RPLS $(\mathbf{p}, \mathbf{v})$ is called* edge-independent *if the verifier $\mathbf{v}$ uses independent random bits for each certificate $c_i(v)$, for all nodes v, and all edges e_i incident to v, $i = 1, \ldots, \deg(v)$.*

We can prove the following result for edge-independent two-sided error RPLSs. The proof is somewhat involved, and due to space shortage, it is given in Appendix D.

Proposition 4.6 *Let $(\mathbf{p}, \mathbf{v})$ be an edge-independent RPLS for $(\mathcal{F}, \mathcal{P})$ with verification complexity κ. Assume that there is a configuration $G_s \in \mathcal{F}$ with $\mathcal{P}(G_s) =$ TRUE such that G contains r pairwise independent isomorphic subgraphs $H_1, \ldots, H_r$ with s edges each, and let $\sigma_i : H_1 \to H_i$ be a port-preserving isomorphism for each $i \in \{1, \ldots, r\}$. If $\kappa < (\frac{1}{2s} - o(1)) \log\log r$, then there are $1 \leq i < j \leq r$ such that G_s is accepted by $(\mathbf{p}, \mathbf{v})$ if and only if $(\sigma_j \circ \sigma_i^{-1})_{\bowtie}(G)_s$ is accepted by $(\mathbf{p}, \mathbf{v})$.*

The following corollary of Proposition 4.6 is the way we use to bound the verification complexity of two-sided error, edge independent RPLSs.

Theorem 4.7 *Let $\mathcal{F}$ be a family of configurations, and let $\mathcal{P}$ be a boolean predicate over $\mathcal{F}$. If there is a configuration $G_s \in \mathcal{F}$ satisfying that (1) G contains r pairwise independent isomorphic subgraphs $H_1, \ldots, H_r$ with s edges each, and (2) there exist r port-preserving isomorphisms $\sigma_i : V(H_1) \to V(H_i)$ such that $\mathcal{P}(G_s) \neq \mathcal{P}(\sigma_{\bowtie}(G)_s)$ for every isomorphism $\sigma = \sigma_j \circ \sigma_i^{-1}$ between H_i and H_j, for $1 \le i \neq j \le r$, then the verification complexity of any edge-independent RPLS for $(\mathcal{F}, \mathcal{P})$ is $\Omega(\frac{\log\log r}{s})$.*

Again, we note that since r cannot be greater than $O(n)$, Theorem 4.7 cannot be used to prove lower bounds greater than $\Omega(\log\log n)$.

Lower bounds for one-sided RPLSs. We can replace the assumption of edge-independent RPLS in Theorem 4.7 by the requirement that the RPLS is *one sided* and obtain essentially the same lower bound. Recall that in one-sided RPLS we insist that if $\mathcal{P}(G_s) = \text{TRUE}$, then the verifier $\mathbf{v}$ must accept *always*, i.e., with probability 1. For this case we have the following proposition, whose proof is much simpler.

Proposition 4.8 *Let $(\mathbf{p}, \mathbf{v})$ be a one-sided RPLS for $(\mathcal{F}, \mathcal{P})$ with verification complexity κ. Assume that there is a configuration $G_s \in \mathcal{F}$ with $\mathcal{P}(G_s) = \text{TRUE}$ such that G contains r pairwise independent isomorphic subgraphs $H_1, \ldots, H_r$ with s edges each, and let $\sigma_i : H_1 \to H_i$ be a port-preserving isomorphism for each $1 \le i \le r$. If for any isomorphism $\sigma^{ij} = \sigma_j \circ \sigma_i^{-1}$ we have that $\mathcal{P}(\sigma_{\bowtie}(G)_s) = \text{FALSE}$, then $\kappa \ge \frac{1}{2s}\log\log r$.*

Proof: Fix G_s as in the statement, and let $(\mathbf{p}, \mathbf{v})$ be a one-sided RPLS for $(\mathcal{F}, \mathcal{P})$. Assume w.l.o.g. that all certificates under $(\mathbf{p}, \mathbf{v})$ have length exactly κ. For each edge $e = \{u, v\}$, let $x(u, v)$ denote the support of the certificates sent over e by u, i.e., all bit strings of length κ with positive probability to be sent from u to v under G_s and $(\mathbf{p}, \mathbf{v})$. Since there are κ bits in each certificate, the number of distinct certificates is at most 2^κ, and hence the number of distinct supports is $2^{(2^\kappa)}$. Ordering the nodes of H_1 somehow and using the order induced by the port-preserving isomorphisms, we can represent each specific setting of the $2s$ certificates sent over the edges of H_i as a $2s\kappa$-long bit string. Now, if $\kappa < \frac{1}{2s}\log\log r$, then $2^{(2^{2s\kappa})} < r$, and hence, by the pigeonhole principle, there are $1 \le i < j \le r$ such that the supports of all the $2s$ respective (directed) edges in H_i and H_j are identical. Define $\sigma = \sigma_j \circ \sigma_i^{-1}$, and let $u' = \sigma(u)$ for some node u in H_i. Let v be a neighbor of u in H_i and let $v' = \sigma(v)$.

Fix any global certificate $\underline{c}$ assignment (assigning a certificate to each direction of each edge of G) that can be generated in G_s by $(\mathbf{p}, \mathbf{v})$ with positive probability. Note that since $\mathcal{P}(G_s) = \text{TRUE}$ and the RPLS is one sided, and since the probability of generating $\underline{c}$ is strictly positive, it must be the case that under $\underline{c}$, the verifier accepts at all nodes (deterministically). Now, let c_1 denote the coordinate of (u, v) in $\underline{c}$, i.e., the certificate sent by u to v in $\underline{c}$. Similarly let c_2 denote the certificate sent by u' to v' in $\underline{c}$. Let $\underline{c}'$ denote the global certificate obtained from $\underline{c}$ by switching c_1 and c_2, i.e., under $\underline{c}'$, u sends c_1 to v' and u' sends c_2 to v' (cf. Figure 1).

We claim that the verifier $\mathbf{v}$ accepts at all nodes under $\underline{c}'$. To see that, note that since $\mathcal{P}(G_s) = \text{TRUE}$, it must be the case (by the independence of certificates received at a node, given the labels) that *any* certificate in $x(u', v')$ sent to v' results in the verifier $\mathbf{v}$ at v' outputting TRUE: otherwise, there will be a non-zero probability that $\mathbf{v}$ rejects at v', resulting in rejecting a "yes" instance. Therefore, the fact that $x(u, v) = x(u', v')$ (by our choice) necessarily implies that if v' receives $c_1 \in x(u, v) = x(u', v')$, the verifier $\mathbf{v}$ at v' outputs TRUE. Similarly for v accepting c_2.

Continuing inductively in this fashion we switch the certificates edge by edge, and arrive at the conclusion that the verifier $\mathbf{v}$ accepts $\sigma_{\bowtie}(G)_s$. Moreover, since we can apply the switching procedure described above to *any* legal certificate assignment $\underline{c}$ for G_s, we have that if $\kappa < \frac{1}{2s}\log\log r$, then $\sigma_{\bowtie}(G)_s$ is accepted by the RPLS with probability 1, contradicting our assumption that $\mathcal{P}(\sigma_{\bowtie}(G)_s) = \text{FALSE}$ and the requirement that a one-sided RPLS rejects a "no" instance with probability at least $\frac{1}{2}$. ▮

We note that all the upper bounds we derive in Section 5 are both edge-independent and one-sided.

5 The Verification Complexity of Some Graph-Theoretic Predicates

We now bound, from above and from below, the deterministic and randomized verification complexity of a few specific problems using the tools developed in Sections 3 and 4. We study three important problems of independent interest. Each of these problems has received attention in the framework of (deterministic) proof-labeling schemes, as well as in other frameworks like distributed algorithm design, property testing, etc. We note that all RPLSs constructed in this section are edge-independent and one-sided RPLSs, and that all lower bounds here are for RPLSs which are either edge independent or one-sided.

In the following, let $\mathcal{F}_{con}$ be the family of all connected graphs. Most proofs are deferred to Appendix E.

5.1 Minimum-Weight Spanning Tree

Recall that a Minimum-weight Spanning Tree (MST) of a weighted n-node graph G is a spanning tree of G whose the sum of all its edge-weights is minimum among all spanning trees of G. In this setting, we assume that every node is aware of the weights of its incident edges (i.e., these weights, indexed by port numbers, are part of its state).

Theorem 5.1 *The randomized verification complexity of* $(\mathcal{F}_{con}, MST)$ *is* $\Theta(\log\log n)$.

The idea in the lower bound proof is to show that even a simpler predicate, namely deciding acyclicity, is hard. The upper bound follows from the $O(\log^2 n)$ upper bound on the deterministic verification complexity proved in [31]. Details can be found in Appendix E.

5.2 Vertex Bi-Connectivity

A connected graph G is called vertex-biconnected if the result of removing any node from G is a connected graph.

In [31], the authors proved a $\Theta(\log n)$ bound on the deterministic verification complexity of the s-t connectivity problem. In this problem, given a connected graph $G = (V, E)$ and two specified nodes $s, t \in V$, the goal is for all nodes to agree on a natural number k, where k is the vertex connectivity between s and t in G. Note that this is not a decision problem as it was presented. Slightly modified, where k is a parameter of the problem, we obtain the problem s-t k-connectivity, where the goal is to decide whether the vertex connectivity between s and t is exactly k, and the $\Theta(\log n)$ bound still holds. This problem is closely related to vertex-biconnectivity, with the main differences that in the latter we consider the connectivity between *all* pairs of nodes and we only check whether it is at least 2. Let v2con denote the predicate of vertex-biconnectivity. We have the following result.

Theorem 5.2 *The deterministic verification complexity of* $(\mathcal{F}_{con}, \text{v2con})$ *is* $\Theta(\log n)$*, and its randomized verification complexity is* $\Theta(\log\log n)$.

The upper bounds follow from the observation that we can encode all relevant information used in the biconnectivity algorithm of Tarjan [22] using $O(\log n)$ bits. The lower bounds use our crossing argument on a cycle with chords (Figure 2). See Appendix E for details.

Another result in [31] regarding connectivity is an upper bound of $O(k \log n)$ on the deterministic verification complexity of the k-flow problem. In this problem, the goal is to decide whether the value of the maximum flow between s and t is exactly k. Using Theorem 3.1 on that result, we get an upper bound of $O(\log k + \log\log n)$ on the randomized verification complexity of the k-flow problem.

5.3 Maximum-Length Cycle

A graph G is called Hamiltonian if there is a cycle in G visiting every node exactly once. For any positive integer c, we define the predicate "cycle-at-most-c" over graphs, which is true for G if and only if no simple cycle in G

contains more than c nodes. We also define the predicate "cycle-at-least-c" over graphs, which is true for G if and only if there is a simple cycle in G with at least c nodes. We have the following upper bounds for cycle-at-least-c.

Theorem 5.3 *The deterministic verification complexity of* $(\mathcal{F}_{con}, \text{cycle-at-least-}c)$ *is* $O(\log n)$*, and its randomized verification complexity is* $O(\log\log n)$.

The upper bounds follow by marking the cycle nodes using $O(\log n)$ bits. The following theorem states lower bounds for cycle-at-least-c.

Theorem 5.4 *The verification complexity of any deterministic PLS for* $(\mathcal{F}_{con}, \text{cycle-at-least-}c)$ *is* $\Omega(\log c)$*, and of any RPLS is* $\Omega(\log\log c)$.

Proof: Let G be a graph that consists of a c-node cycle, with port numbers consistently ordered, and additional edges from one to all other nodes of the graph (see Figure 2). Formally, $G = (\{v_0, \ldots, v_{n-1}\}, E_c \cup E_0)$ where $E_c = \big\{\{v_i, v_{(i+1) \bmod c}\} \mid i = 0, \ldots, c-1\big\}$ and $E_0 = \big\{\{v_0, v_j\} \mid j = 2, \ldots, n-1\,,\ j \neq c-1\big\}$. This graph satisfies cycle-at-least-$c(G) = \text{TRUE}$. Let $H_1 = \{v_0, v_1\}$. For $i = 2, \ldots, \left\lfloor \frac{c}{3} \right\rfloor - 1$, let $H_i = \{v_{3i}, v_{3i+1}\}$, and $\sigma_i : V(H_1) \to V(H_i)$ satisfying $\sigma_i(v_0) = v_{3i}$ and $\sigma_i(v_1) = v_{3i+1}$. For any $1 \leq i < j \leq \left\lfloor \frac{c}{3} \right\rfloor - 1$, let $\sigma^{ij} = \sigma_j \circ \sigma_i^{-1}$. We get that ${\sigma^{ij}}_{\bowtie}(G)$ consists of two disjoint cycles of size strictly less than $c-1$ each, with some edges in E_0 between them. The size of every simple cycle in ${\sigma^{ij}}_{\bowtie}(G)$ is strictly less than c, and therefore, cycle-at-least-$c({\sigma^{ij}}_{\bowtie}(G)) = \text{FALSE}$. Hence, the conditions of Theorems 4.4 and 4.7 are satisfied, and the lower bounds follow. ▮

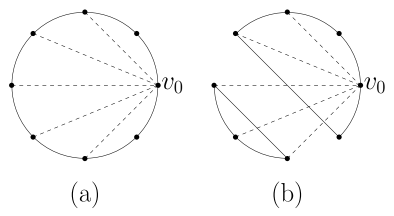
Figure 2: *(a): The graph G in the proof of Theorem 5.2, and restricted to nodes $\{v_0, \ldots, v_{c-1}\}$ in the proof of Theorem 5.4. Dashed edges are E_0. (b): The graph ${\sigma^{ij}}_{\bowtie}(G)$.*

The lower bound in Theorem 5.4 shows the hardness of distinguishing between graphs which contain a cycle of length c and ones which contain only cycles of length up to $c-1$. We now present an alternative technique, which shows that this lower bound holds also in the case where the question is to distinguish between graphs with a cycle of size n and graphs where all cycles are strictly smaller than c. Formally, let $\mathcal{F} = \mathcal{F}_1 \cup \mathcal{F}_2$, where $\mathcal{F}_1$ is the family of all connected graphs over $n \geq c$ nodes that contains an n-node cycle, and $\mathcal{F}_2$ is the family of all connected graphs over $n \geq c$ nodes where all cycles have size at most $c-1$.

Theorem 5.5 *The verification complexity of any deterministic PLS for* $(\mathcal{F}, \text{cycle-at-least-}c)$ *is* $\Omega(\log c)$.

We note that the proof of Theorem 5.5, which can be found in Appendix E, applies crossing iteratively.

Finally, consider now the problem of deciding the predicate cycle-at-most-c. We note that it is co-NP hard, because for $c = n-1$, cycle-at-most-c is the complement of Hamiltonian Cycle. Observe that a proof-labeling scheme for $(\mathcal{F}_{con}, \text{cycle-at-most-}c)$, with polynomial verification complexity and polynomial computation at each node can be translated into a sequential verifiable proof of polynomial size, and hence the existence of such a PLS would imply that NP = co-NP. Therefore, we do not expect to find an efficient PLS (let alone RPLS) for this problem. The universal scheme, in which the computation complexity at each node is unbounded, is the best scheme we know for this problem from the viewpoint of verification complexity. A lower bound on the verification complexity is presented in the following theorem.

Theorem 5.6 *The verification complexity of any deterministic proof-labeling scheme for* $(\mathcal{F}_{con}, \text{cycle-at-most-}c)$ *is* $\Omega(\log \frac{n}{c})$*, and of any randomized proof-labeling scheme is* $\Omega(\log\log \frac{n}{c})$.

The proof of Theorem 5.6 uses our crossing argument on a chain of cycles. See Appendix E for details.

6 Conclusion

In this paper we introduced the concept of randomized proof-labeling schemes (RPLS) and derived a few fundamental results as well as a few concrete results. The main message of the paper is that randomized proof-labeling schemes have very low communication complexity, and can be efficiently used for verifying *every* predicate, whereas there are some natural predicates that cannot be verified within reasonable communication complexity by deterministic schemes. Many questions remain open. We list a few below.

- What is the relation between one-sided and two-sided RPLSs?
- Can the lower bound of Theorem 4.7 be extended to two-sided RPLSs which are not edge independent? And, what about the model that allows shared randomness between nodes?
- What are the exact connections between RPLS and the different complexity classes in [15], including LD, BPLD, NLD, and BPNLD?
- Can the complexity of the prover $\mathbf{p}$ be also accounted for in proof-labeling scheme? In particular, does randomization matter in this respect?
- In this paper we mostly used the crossing technique with single-edge gadgets, with the exception of Theorem 5.5 where we applied it iteratively. Are there examples of problems for which we need more sophisticated uses of crossing.

References

[1] Y. Afek, S. Kutten, and M. Yung. The local detection paradigm and its application to self-stabilization. *Theor. Comput. Sci.*, 186(1-2):199–229, 1997.

[2] S. Alstrup, P. Bille, and T. Rauhe. Labeling schemes for small distances in trees. In *14th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 689–698, 2003.

[3] S. Alstrup, C. Gavoille, H. Kaplan, and T. Rauhe. Nearest common ancestors: a survey and a new distributed algorithm. In *14th ACM Symp. on Parallelism in Algorithms and Architectures (SPAA)*, pages 258–264, 2002.

[4] S. Alstrup and T. Rauhe. Small induced-universal graphs and compact implicit graph representations. In *43rd Symposium on Foundations of Computer Science (FOCS)*, pages 53–62, 2002.

[5] H. Arfaoui, P. Fraigniaud, D. Ilcinkas, and F. Mathieu. Distributedly testing cycle-freeness. In *40th Int. Workshop on Graph-Theoretic Concepts in Computer Science (WG)*, LNCS, pages 15–28. Springer, 2014.

[6] H. Arfaoui, P. Fraigniaud, and A. Pelc. Local decision and verification with bounded-size outputs. In *15th Symp. on Stabilization, Safety, and Security of Distributed Systems (SSS)*, LNCS, pages 133–147. Springer, 2013.

[7] B. Awerbuch, S. Kutten, Y. Mansour, B. Patt-Shamir, and G. Varghese. Time optimal self-stabilizing synchronization. In *25th ACM Symposium on Theory of Computing (STOC)*, pages 652–661, 1993.

[8] B. Awerbuch, B. Patt-Shamir, and G. Varghese. Self-stabilization by local checking and correction. In *32nd Symposium on Foundations of Computer Science (FOCS)*, pages 268–277. IEEE, 1991.

[9] L. Blin, P. Fraigniaud, and B. Patt-Shamir. On proof-labeling schemes versus silent self-stabilizing algorithms. In *16th Int. Symp. on Stabilization, Safety, and Security of Distributed Systems (SSS)*, LNCS, pages 18–32. Springer, 2014.

[10] E. Cohen, E. Halperin, H. Kaplan, and U. Zwick. Reachability and distance queries via 2-hop labels. In *13th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 937–946, 2002.

[11] A. Das Sarma, S. Holzer, L. Kor, A. Korman, D. Nanongkai, G. Pandurangan, D. Peleg, and R. Wattenhofer. Distributed verification and hardness of distributed approximation. *SIAM J. Comput.*, 41(5):1235–1265, 2012.

[12] P. Fraigniaud and C. Gavoille. Routing in trees. In *28th International Colloquium on Automata, Languages and Programming (ICALP)*, LNCS, pages 757–772. Springer, 2001.

[13] P. Fraigniaud and C. Gavoille. A space lower bound for routing in trees. In *19th Symp. on Theoretical Aspects of Computer Science (STACS)*, LNCS, pages 65–75. Springer, 2002.

[14] P. Fraigniaud and A. Korman. On randomized representations of graphs using short labels. In *21st ACM Symp. on Parallelism in Algorithms and Architectures (SPAA)*, pages 131–137, 2009.

[15] P. Fraigniaud, A. Korman, and D. Peleg. Towards a complexity theory for local distributed computing. *J. ACM*, 60(5):35, 2013.

[16] P. Fraigniaud, S. Rajsbaum, and C. Travers. Locality and checkability in wait-free computing. *Distributed Computing*, 26(4):223–242, 2013.

[17] P. Fraigniaud, S. Rajsbaum, and C. Travers. On the number of opinions needed for fault-tolerant run-time monitoring in distributed systems. In *5th International Conference on Runtime Verification (RV)*, LNCS, pages 92–107. Springer, 2014.

[18] C. Gavoille, M. Katz, N. A. Katz, C. Paul, and D. Peleg. Approximate distance labeling schemes. In *9th European Symposium on Algorithms (ESA)*, pages 476–487, 2001.

[19] C. Gavoille and C. Paul. Split decomposition and distance labelling: An optimal scheme for distance hereditary graphs. *Electronic Notes in Discrete Mathematics*, 10:117–120, 2001.

[20] C. Gavoille, D. Peleg, S. Perennes, and R. Raz. Distance labeling in graphs. In *12th ACM Symposium on Discrete Algorithms (SODA)*, pages 210–219, 2001.

[21] M. Göös and J. Suomela. Locally checkable proofs. In *30th ACM Symp. on Principles of Distributed Computing (PODC)*, pages 159–168, 2011.

[22] J. E. Hopcroft and R. E. Tarjan. Efficient algorithms for graph manipulation. *Commun. ACM*, 16(6):372–378, 1973.

[23] G. Itkis and L. A. Levin. Fast and lean self-stabilizing asynchronous protocols. In *35th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 226–239. IEEE, 1994.

[24] S. Kannan, M. Naor, and S. Rudich. Implicit representation of graphs. *SIAM J. Discrete Math.*, 5(4):596–603, 1992.

[25] H. Kaplan and T. Milo. Short and simple labels for small distances and other functions. In *7th Int. Workshop on Algorithms and Data Structures (WADS)*, pages 246–257, 2001.

[26] M. Katz, N. A. Katz, A. Korman, and D. Peleg. Labeling schemes for flow and connectivity. In *13th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 927–936, 2002.

[27] M. Katz, N. A. Katz, and D. Peleg. Distance labeling schemes for well-separated graph classes. *Discrete Applied Mathematics*, 145(3):384–402, 2005.

[28] A. Korman. Labeling schemes for vertex connectivity. *ACM Transactions on Algorithms*, 6(2), 2010.

[29] A. Korman and S. Kutten. Distributed verification of minimum spanning trees. *Distributed Computing*, 20:253–266, 2007.

[30] A. Korman, S. Kutten, and T. Masuzawa. Fast and compact self stabilizing verification, computation, and fault detection of an MST. In *30th Annual ACM Symposium on Principles of Distributed Computing (PODC)*, pages 311–320, 2011.

[31] A. Korman, S. Kutten, and D. Peleg. Proof labeling schemes. *Distributed Computing*, 22(4):215–233, 2010.

[32] A. Korman, D. Peleg, and Y. Rodeh. Constructing labeling schemes through universal matrices. *Algorithmica*, 57(4):641–652, 2010.

[33] E. Kushilevitz and N. Nisan. *Communication complexity*. Cambridge University Press, 1997.

[34] C. H. Papadimitriou. *Computational Complexity*. Addison Wesley Longman, 1994.

[35] D. Peleg. Proximity-preserving labeling schemes and their applications. In *25th International Workshop Graph-Theoretic Concepts in Computer Science (WG)*, pages 30–41, 1999.

[36] D. Peleg. Informative labeling schemes for graphs. *Theor. Comput. Sci.*, 340(3):577–593, 2005.

[37] R. E. Tarjan. Depth-first search and linear graph algorithms. *SIAM J. Comput.*, 1(2):146–160, 1972.

[38] M. Thorup. Compact oracles for reachability and approximate distances in planar digraphs. In *42nd Symp. on Foundations of Computer Science (FOCS)*, pages 242–251, 2001.

[39] M. Thorup and U. Zwick. Compact routing schemes. In *13th ACM Symp. on Parallelism in Algorithms and Architectures (SPAA)*, pages 1–10, 2001.

APPENDIX

A Proof of Theorem 3.1

We first prove the following upper bound implied by Lemma 3.2.

Lemma A.1 (based on [33]) *There exists a 2-party randomized protocol π for EQ with communication complexity $O(\log \lambda)$ where λ is the length of the input strings. Moreover, for any input strings a, b we have that* $\Pr[\pi(a,b) = \text{TRUE} \mid a = b] = 1$ *and* $\Pr[\pi(a,b) = \text{FALSE} \mid a \neq b] > 2/3$.

Proof: Let $a = a_0 a_1 \ldots a_{\lambda-1}$ be the λ-bit input string of Alice, and $b = b_0 b_1 \ldots b_{\lambda-1}$ be the λ-bit input string of Bob. We view a and b as polynomials A and B over the field $GF[p]$ where $3\lambda < p < 6\lambda$ is a prime number. That is, we define

$$\begin{aligned} A(x) &= a_0 + a_1 x + \cdots + a_{\lambda-1} x^{\lambda-1} \bmod p, \text{ and} \\ B(x) &= b_0 + b_1 x + \cdots + b_{\lambda-1} x^{\lambda-1} \bmod p. \end{aligned}$$

To solve the problem, Alice picks a number x in $GF[p]$ uniformly at random, and sends the pair $(x, A(x))$ to Bob. The size of $(x, A(x))$ is $O(\log p) = O(\log \lambda)$. Bob outputs TRUE if $A(x) = B(x)$ and FALSE otherwise. Obviously, if $a = b$ then $A(x) = B(x)$ and the output is always TRUE. If $a \neq b$, then A and B are two distinct polynomials of degree $\lambda - 1$, so they can be equal on at most $\lambda - 1$ elements of the field $GF[p]$. Hence, the probability that $A(x) = B(x)$ when $a \neq b$ is at most $\frac{\lambda-1}{p} < \frac{\lambda}{3\lambda} = \frac{1}{3}$. Therefore, the probability to reject an input composed of two different strings is at least $2/3$, as desired. This completes the proof of the lemma. ▮

Proof of Theorem 3.1: Let $(\mathbf{p}, \mathbf{v})$ be a proof-labeling scheme for $(\mathcal{F}, \mathcal{P})$ with verification complexity κ. We construct a randomized proof-labeling scheme $(\mathbf{p}', \mathbf{v}')$ for $(\mathcal{F}, \mathcal{P})$ as follows. Let $G_s \in \mathcal{F}$ be a configuration satisfying predicate $\mathcal{P}$. For every node v, the prover $\mathbf{p}'$ sets the label of v as the vector of labels

$$\ell'(v) = \big(\ell(v), \ell(w_1), \ldots, \ell(w_d)\big)$$

where $\ell(\cdot)$ is the label assignment for G_s provided by $\mathbf{p}$, and $w_1, \ldots, w_d$ denotes the $d = \deg(v)$ neighbors of v in G, ordered by port number. To compute the certificate $c(v)$ of node v, given a label

$$\ell'(v) = \big(\ell_0, \ell_1, \ldots, \ell_d\big),$$

the (random) verifier $\mathbf{v}'$ aims at playing the protocol in the proof of Lemma A.1, with $\lambda = \kappa$. That is, it picks a random number $x \in GF[p]$ where $3\kappa < p < 6\kappa$ is a prime number, and sets $c(v) = (x, P_v^{(0)}(x))$ where $P_v^{(0)}$ is the polynomial of v corresponding to the κ-bit label ℓ_0. Given the certificates $c(w_i) = (x_i, P_{w_i}^{(0)}(x_i))$ received from each of v's neighbors w_i, $i = 1, \ldots, d$, the verifier $\mathbf{v}'$ returns FALSE if $P_v^{(i)}(x_i) \neq P_{w_i}^{(0)}(x_i)$ for some i, where $P_v^{(i)}$ is the polynomial of v corresponding to the κ-bit label ℓ_i. If $P_v^{(i)}(x_i) = P_{w_i}^{(0)}(x_i)$ for all $i = 1, \ldots, d$, then $\mathbf{v}'$ applies verifier $\mathbf{v}$ at node v to the set of labels $(\ell_0, \ell_1, \ldots, \ell_d)$, and returns TRUE or FALSE in agreement to what $\mathbf{v}$ outputs on this set. This concludes the construction of the random scheme $(\mathbf{p}', \mathbf{v}')$.

The logarithmic verification complexity of $(\mathbf{p}', \mathbf{v}')$ follows directly from the construction. We now show correctness of the scheme.

If G_s satisfies the predicate, then, with the labels assigned by $\mathbf{p}'$, we have that, for every node v, $\ell'(v) = \big(\ell(v), \ell(w_1), \ldots, \ell(w_d)\big)$, with the labels $\ell(\cdot)$ assigned by $\mathbf{p}$. Therefore $P_v^{(i)} = P_{w_i}^{(0)}$ for any $i = 1, \ldots, d$. Thus, in particular, $P_v^{(i)}(x_i) = P_{w_i}^{(0)}(x_i)$ for any random choice of x_i by the ith neighbor w_i of v. It follows that $\mathbf{v}'$ applies $\mathbf{v}$ at node v on the set $(\ell(v), \ell(w_1), \ldots, \ell(w_d))$, for which $\mathbf{v}$ returns TRUE at v. Hence, $\mathbf{v}'$ returns TRUE at v as well.

If G_s does not satisfy the predicate, we first note that if the labels are *consistent* in the sense that, at every node v, the sub-label ℓ_i at v is equal to the sub-label ℓ_0 at w_i, then $\mathbf{v}'$ returns FALSE at some node because $\mathbf{v}$ cannot be fooled by “fake” labels. In the case the labels are not consistent, there is at least one pair of adjacent nodes $\{v, w\}$ such that w is the ith neighbor of v, and the sub-label ℓ_i of v’s label is different from the sub-label ℓ_0 of w. In this case we rely on the correctness of the equality protocol in Lemma A.1, insuring that v outputs FALSE with probability at least $2/3$. ▮

B Universal Construction of PLS

We recall a universal construction that works for any family of configurations, and any boolean predicate over that family (see [15, 21, 31]). Let $\mathcal{F}$ be a family of configurations with states in S, and let $\mathcal{P}$ be a boolean predicate over $\mathcal{F}$. Assume that every state in S can be represented using $k = k(n)$ bits in n-node networks. There is a proof-labeling scheme $(\mathbf{v}, \mathbf{p})$ with verification complexity $O(\min\{n^2, m \log n\} + nk)$. To see why, notice that any n-node m-edge graph can be represented by an $n \times n$ adjacency matrix on $O(n^2)$ bits, and also by an adjacency list on $O(m \log n)$ bits. The global state of a configuration in an n-node graph can be represented by an array consuming $O(nk)$ bits. So, overall, a configuration G_s can be represented on $O(\min\{n^2, m \log n\} + nk)$ bits. If G_s satisfies $\mathcal{P}$, then the oracle provides each node v with a representation R of G_s. The verifier at each node v checks the local consistency of the representation R of G_s with the neighborhood of v. If the local neighborhood of v is consistent with the representation R of G_s given to the nodes in this neighborhood, then v checks whether R satisfies $\mathcal{P}$. If all tests are passed, then v accepts, otherwise v rejects. This scheme is correct because if the description of the neighborhood of v in the actual configuration is consistent with the neighborhood of v in the given representation R for every node v, then necessarily the actual configuration is isomorphic to R (recall that the state of a node includes its identity, and this identity is unique in the network).

C Proof of a Lower Bound for Universal RPLS

In this section we prove Theorem 3.5.

We start by proving a lemma for each of the two parts of the lower bound separately. Let $\mathcal{F}_1$ be the family of all connected graphs over n nodes, where the state of each node is only its identifier. A connected graph $G = (V, E)$ is *symmetric* if there exists an edge $e \in E$ such that $G' = (V, E \setminus e)$ consists of exactly two isomorphic connected components. The predicate Sym over a graph G is true if and only if G is symmetric.

Lemma C.1 *Any randomized proof-labeling scheme for* $(\mathcal{F}_1, \text{Sym})$ *has verification complexity* $\Omega(\log n)$.

In the proof of this lemma we show that any randomized proof-labeling scheme for $(\mathcal{F}_1, \text{Sym})$, with verification complexity κ, can be used to construct a 2-party communication protocol for EQ of n-bit strings, with error probability of at most $1/3$, using κ bits of communication. Then, from Lemma 3.2, we get the desired lower bound. We give the full proof.

Proof of Lemma C.1: Let $(\mathbf{p}, \mathbf{v})$ be a randomized proof-labeling scheme for $(\mathcal{F}_1, \text{Sym})$. We show how to derive a 2-party protocol for EQ using that scheme.

Let x and y be the λ-bit input strings of Alice and Bob, respectively. We use each string as a description of a graph in the following way. Given a λ-bit string z, let the bits of z be numbered $z_0, \ldots, z_{s-1}$, and let $\nu = 2\lambda + 3$. We define the corresponding ν-node graph $G(z) = (V, E)$ as follows (See Figure 3 for illustration). The set of nodes V is composed of three node sets: U and W of size λ each, and T of size 3. Denote the nodes of U by $\{u_0, \ldots, u_{\lambda-1}\}$, the nodes of W by $\{w_0, \ldots, w_{\lambda-1}\}$ and the nodes of T by $\{t_0, t_1, t_2\}$. The set of edges E is composed of four edge sets: E_u, E_w, E_t and $\{e_0 = \{t_0, u_0\}\}$. Three of the sets are independent on z, $E_u = \{\{u_i, u_{i+1}\} \mid 0 \le i \le \lambda - 2\}$ is a path on the nodes of U, $E_t = \{\{t_i, t_j\} \mid 0 \le i < j \le 2\}$ is a triangle over the nodes of T, and e_0 is an edge connecting the triangle to u_0. The edge set E_w is dependent on z in the following way. $E_w = \{\{w_i, u_i\} \mid z_i = 1\} \cup \{\{w_i, t_1\} \mid z_i = 0\}$. Intuitively, a node w_i represents ‘1’ in the string

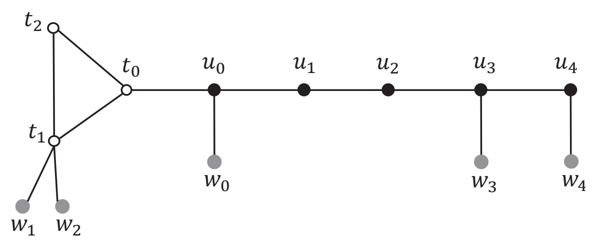

Figure 3: *Illustration for the proof of Lemma C.1. The graph $G(z)$, where $z = 10011$.*

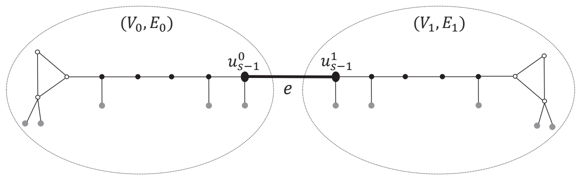

Figure 4: *The graph $G(z, z)$, where $z = 10011$.*

z if it is connected to the path of nodes U, and it represents '0' if it is connected to t_1. The edge e_0 indicates u_0, by connecting the triangle to it, to avoid isomorphism between $G(z)$ and $G(z^R)$ where $z \neq z^R$ (z reversed).

Given two λ-bit strings z and z', define the graph $G(z, z') = (V_0 \cup V_1, E_0 \cup E_1 \cup e)$ as follows (see Figure 4). $V_0 = U_0 \cup W_0 \cup T_0$ where $U_0 = \{u^0_0, \ldots, u^0_{\lambda-1}\}$, $W_0 = \{w^0_0, \ldots, w^0_{\lambda-1}\}$ and $T_0 = \{t^0_0, t^0_1, t^0_2\}$. $V_1 = U_1 \cup W_1 \cup T_1$ where $U_1 = \{u^1_0, \ldots, u^1_{\lambda-1}\}$, $W_1 = \{w^1_0, \ldots, w^1_{\lambda-1}\}$ and $T_1 = \{t^1_0, t^1_1, t^1_2\}$. E_0 is the set of edges of $G(z)$ over nodes V_0, E_1 is the set of edges of $G(z')$ over nodes V_1, and $e = \{u^0_{\lambda-1}, u^1_{\lambda-1}\}$

Claim C.2 *For any two λ-bit strings z and z', $G(z, z')$ is symmetric if and only if $z = z'$.*

To establish the claim, notice first that, since both z and z' are of length λ, $|V_0| = |V_1| = 2\lambda + 3$. Hence, since e is the only edge between V_0 and V_1, removing any other edge from $G(z, z')$ either not disconnects the graph or disconnects it to two connected components with different number of nodes, and in particular not isomorphic. By construction of $G(z, z')$, removing the edge e from $G(z, z')$ disconnects the graph to two connected components which are exactly $G(z) = (V_0, E_0)$ and $G(z') = (V_1, E_1)$. Therefore, $G(z, z')$ is symmetric if and only if $G(z)$ and $G(z')$ are isomorphic. If $z = z'$ then $G(z) = G(z')$, so $G(z, z')$ is symmetric. Suppose now that $z \neq z'$. For $\lambda = 1$, one can easily verify that $G('0')$ and $G('1')$ are not isomorphic. Note that for every string b, $G(b)$ contains exactly one triangle. For $\lambda \geq 2$, by construction, in both $G(z)$ and $G(z')$ there is only one node of the triangle that is connected to a path of length at least λ, which does not include a triangle edge. In $G(z)$ this node is t^0_0, and in $G(z')$ this node is t^1_0. Since there is no isomorphism between $G(z)$ and $G(z')$ that does not map t^0_0 to t^1_0, from now on, we consider only the nodes that are reachable from t^0_0 and t^1_0 without using a triangle edge. Let i be the smallest index such that $z_i \neq z'_i$. W.l.o.g, $z_i = 1$ and $z'_i = 0$. If $i < \lambda - 1$, by construction, there is exactly one node of degree greater than one at distance $i + 1$ from t^0_0. This node is u^0_i, and its degree is 3. Similarly, in that case, there is exactly one node of degree greater than one at distance $i + 1$ from t^1_0. This node is u^1_i, and its degree is 2. Therefor, $G(z)$ and $G(z')$ are not isomorphic in that case. If $i = \lambda - 1$, there is a node, w^0_i, at distance $i + 2$ from t^0_0. On the other hand, there is no node at distance $i + 2$ from t^1_0. Therefore, $G(z)$ and $G(z')$ are not isomorphic. Hence, $G(z, z')$ is not symmetric. This conclude the proof of Claim C.2.

We now construct a protocol for EQ using the scheme $(\mathbf{p}, \mathbf{v})$. Given λ, define ν as above, and set $n = 2\nu$.

1. Alice constructs the graph $G(x, x)$.
2. Bob constructs the graph $G(y, y)$.
3. Alice and Bob apply, separately, the prover $\mathbf{p}$ to $G(x, x)$ and $G(y, y)$, respectively, and in this way obtain labels for all nodes in their graph.

4. Alice and Bob simulate the verifier **v** in the following way: Alice simulates the nodes in V_0 of $G(x,x)$, and Bob simulates the nodes in V_1 of $G(y,y)$. The only communication between Alice and Bob induced by this simulation consists of exchanging the bits that are crossing link $\{u^0_{\lambda-1}, u^1_{\lambda-1}\}$.
5. Alice (resp., Bob) outputs TRUE if and only if all nodes she (resp., he) simulated accept.

Let $G = G(x,y)$. G has n nodes, and, by Claim C.2, we have

$$x = y \iff \text{Sym}(G) = \text{TRUE}.$$

Hence, if $x = y$, then, with probability at least $2/3$, all nodes of G output TRUE. Thus, with probability at least $2/3$, both Alice and Bob returns TRUE. Similarly, if $x \neq y$, then, with probability at least $2/3$, at least one node of G outputs FALSE. Hence, with probability at least $2/3$, either Alice, or Bob, or both, returns FALSE. Therefore, the 2-party protocol is correct with probability at least $2/3$. In this protocol, Alice and Bob exchange κ bits. If $\kappa = o(\log n)$ then $\kappa = o(\log \lambda)$ as well since $n \sim \lambda$. This would be in contradiction with the lower bound in Lemma 3.2. Thus the verification complexity of $(\mathbf{p}, \mathbf{v})$ is at least $\Omega(\log n)$. ∎

The second lemma is for the $\Omega(\log k)$ part of the lower bound. For any function $k(n) \in \Omega(\log n)$, let $\mathcal{F}_k$ be the family of all connected symmetric graphs over the set of n nodes V, and states from the set $S = [1,n] \times \{0,1\}^k$, to distinguish identities in $[1,n]$ from the rest. Given a state $s(u) = (x,y)$, we note $y = s'(u)$. Consider the predicate Unif that is true on $G_s \in \mathcal{F}_k$ if and only if $s'(u) = s'(v)$ for every two nodes $u, v \in V$.

Lemma C.3 *For any function $k(n)$, any randomized proof-labeling scheme for $(\mathcal{F}_k, \text{Unif})$ has verification complexity $\Omega(\log k)$.*

Proof: Let $k(n)$ be any function in $\Omega(\log n)$, and let $(\mathbf{p}, \mathbf{v})$ be a randomized proof-labeling scheme for $(\mathcal{F}_k, \text{Unif})$. We show how to derive a 2-party protocol for EQ using that scheme. Given a bit-string z, let $G(z)$ be the graph which consists of a single edge $\{v_1, v_2\}$, whose extremities have respective states $(1,z)$ and $(2,z)$. Let x and y be the k-bit input strings of Alice and Bob, respectively. The protocol to solve EQ(x,y) is as follows.

1. Alice constructs $G(x)$, and Bob constructs $G(y)$.
2. Alice uses **p** to construct the labels for both nodes of $G(x)$, and so does Bob to construct the labels for $G(y)$.
3. Alice simulates the verifier **v** on $v_1 \in G(x)$, and Bob does so on $v_2 \in G(y)$, in order to generate the certificates $c(v_1)$ and $c(v_2)$, respectively.
4. Alice and Bob exchange their certificates.
5. Alice outputs the result of **v** at v_1, and Bob outputs the result of **v** at v_2.

Let G be the graph which consists of a single edge $\{v_1, v_2\}$, whose extremities have respective states $(1,x)$ and $(2,y)$. By construction, we have

$$x = y \iff \text{Unif}(G).$$

Therefore, by Lemma 3.2, the verification complexity of $(\mathbf{p}, \mathbf{v})$ is $\Omega(\log k)$. ∎

Proof of Theorem 3.5: Let $\mathcal{F} = \mathcal{F}_1 \cup \mathcal{F}_k$. Consider the predicate Unif-Sym that is true on $G_s \in \mathcal{F}$ if and only if $\text{Unif}(G_s) \wedge \text{Sym}(G_s) = \text{TRUE}$. Let $(\mathbf{p}, \mathbf{v})$ be a randomized proof-labeling scheme for $(\mathcal{F}, \text{Unif-Sym})$. By Lemma C.1, and the fact that for every graph $G_1 \in \mathcal{F}_1$ it holds that $\text{Unif}(G_1) = \text{TRUE}$, the verification complexity of $(\mathbf{p}, \mathbf{v})$ is $\Omega(\log n)$. By Lemma C.3, and the fact that for every graph $G_2 \in \mathcal{F}_k$ it holds that $\text{Sym}(G_2) = \text{TRUE}$, the verification complexity of $(\mathbf{p}, \mathbf{v})$ is $\Omega(\log k)$. Therefore, the verification complexity of $(\mathbf{p}, \mathbf{v})$ is $\Omega(\log n + \log k)$, and we get the desired family $\mathcal{F}$ and predicate $\mathcal{P}$. ∎

D Proof of Proposition 4.6

We start the proof by a collection of technical preliminary results. Given a real number x and $0 < \epsilon \leq 1$, we denote by $\lfloor x \rfloor_\epsilon$ the value of x rounded down to the closest integer multiple of ϵ, i.e., $\lfloor x \rfloor_\epsilon \overset{\text{def}}{=} \lfloor \frac{x}{\epsilon} \rfloor \epsilon$. Given a real function $f : X \to \mathbb{R}$ over a set X, and $\epsilon > 0$, we define the *ϵ-rounded* $f_\epsilon : X \to \mathbb{R}$ of f by $f_\epsilon(x) \overset{\text{def}}{=} \lfloor f(x) \rfloor_\epsilon$ for every $x \in X$.

We shall consider ϵ-rounded probability distributions. Note that an ϵ-rounded probability distribution is not necessarily a probability distribution, because it may not sum up to 1. However, it has the following probabilistic interpretation. Let X be a set, and let $\Pr$ and $\Pr'$ be two probability distributions over X. Then

$$\Pr_\epsilon = \Pr'_\epsilon \;\Rightarrow\; \forall Y \subseteq X, \left|\Pr[Y] - \Pr'[Y]\right| \le \epsilon |X| \,. \tag{1}$$

Indeed, if $\Pr_\epsilon[x] = \Pr'_\epsilon[x]$ for every $x \in X$, then we have that $\left|\Pr[x] - \Pr'[x]\right| < \epsilon$ for every x. Using the triangle inequality we thus get $\left|\Pr[Y] - \Pr'[Y]\right| \le \sum_{x\in Y} \left|\Pr[x] - \Pr'[x]\right| < \epsilon|Y| \le \epsilon|X|$.

The advantage of ϵ-rounded distributions is that there are not too many. Indeed, let X be a finite set, and let $\mathcal{D}$ be the set of all probability distributions over X. The number of distinct ϵ-rounded distributions over X is at most $2^{|X|}\epsilon^{-|X|}$, that is,

$$|\{\Pr_\epsilon \mid \Pr \in \mathcal{D}\}| \le (2/\epsilon)^{|X|}. \tag{2}$$

This is because the set of ϵ-rounded distributions is a subset of the functions from X to $\{\epsilon i, i = 0, 1 \ldots, \lfloor 1/\epsilon \rfloor\}$, which implies that their number is at most $(1 + \frac{1}{\epsilon})^{|X|} \le 2^{|X|}\epsilon^{-|X|}$.

We now have all the ingredients we need to prove the proposition. Let G_s be a configuration and $\{\sigma_i\}_{i=1}^r$ be the isomorphisms as described in the proposition. Note that for any $i \neq j$, if G_s is labeled by $\mathbf{p}$ and the crossing with $\sigma = \sigma_j \circ \sigma_i^{-1}$ satisfies

$$\Big|\Pr[\mathbf{v} \text{ accepts } G_s] - \Pr[\mathbf{v} \text{ accepts } \sigma_{\bowtie}(G)_s]\Big| < \frac{1}{3}, \tag{3}$$

then G_s is accepted by $(\mathbf{p}, \mathbf{v})$ if and only if $\sigma_{\bowtie}(G)_s$ is accepted by $(\mathbf{p}, \mathbf{v})$. We prove that if $\kappa < (\frac{1}{2s} - o(1)) \log\log r$, then there exist $i \neq j$ that satisfy Eq. (3). We use a counting argument. Order the edges of H_1 arbitrarily, and obtain, using σ_i, an ordering for each H_i. Assume w.l.o.g. that all certificates are exactly κ bits long. Then, using the order we defined, there is a 1-1 correspondence between each $2\kappa s$ bit string and each particular choice of certificates communicated in any H_i. Let $\mathcal{D}$ denote the set of distributions over $2\kappa s$-long bit strings, and define

$$\epsilon = 1/(12s \cdot 2^{2s\kappa}) \,.$$

Consider the set of distributions in $\mathcal{D}$, ϵ-rounded. Since there are at most $2^{2s\kappa}$ bit strings of length $2\kappa s$, from Eq. (2) we conclude that there are no more than $(2/\epsilon)^{(2^{2s\kappa})}$ such ϵ-rounded distributions. Let us now make the following technical observation. Let $\alpha \ge 1$, $\beta \ge 1$ and $\gamma \ge 2$ be such that $\log(\beta + \log\alpha) = o(\log\log\gamma)$. Then

$$\beta < (1 - o(1)) \log\log\gamma \;\Rightarrow\; \gamma > (\alpha 2^\beta)^{(2^\beta)}.$$

This follows from the fact that

$$\beta < \log\log\gamma - \log(\beta + \log\alpha) \iff 2^\beta(\beta + \log\alpha) < \log\gamma \iff (\alpha 2^\beta)^{(2^\beta)} < \gamma.$$

By setting $\alpha = 24s$ and $\beta = 2s\kappa$ we obtain that the number of ϵ-rounded distributions satisfies

$$(2/\epsilon)^{(2^{2s\kappa})} = \left(24s \cdot 2^{2s\kappa}\right)^{(2^{2s\kappa})} < r.$$

Therefore, by the pigeonhole principle, it must be the case that among $H_1, \ldots, H_r$ there are H_i and H_j, where $i \neq j$, with identical ϵ-rounded distributions over the certificates.

Now, let $\sigma = \sigma_j \circ \sigma_i^{-1}$. For any $u \in H_i$, we say that u and $\sigma(u) \in H_j$ are *siblings*. Consider running $\mathbf{v}$ on G_s and on $\sigma_{\bowtie}(G)_s$, where, in both cases, we assume that the correct labels are given to the nodes by prover $\mathbf{p}$ applied to G_s. This means that in both G_s and $\sigma_{\bowtie}(G)_s$, the distributions of certificates sent by sibling nodes are the same.

However, the distributions of certificates *received* by sibling nodes may have changed (albeit only slightly, as we show next), due to crossing H_i with H_j.

To analyze **v** on G_s and $\sigma_{\bowtie}(G)_s$, we change the certificates sent in G_s to those sent in $\sigma_{\bowtie}(G)_s$ inductively, and show that each such modification results only in a small change in the probability of acceptance. We view $G=(V,E)$ as a symmetric directed graph, i.e., each edge $e=\{u,v\}\in E$ is viewed as two symmetric arcs (u,v) and (v,u). Let us order these arcs in G arbitrarily, and let $\mathcal{C}$ denote the set of certificate vectors $\vec{c}$ for which the verifier **v** accepts G_s, with coordinates ordered according to the fixed order of the arcs. Consider a node v in H_i, and one of its incoming arcs (u,v) in H_i. Let u' and v' in H_j be the respective siblings of u and v. Assume, w.l.o.g., that $(u,v)=e_1$. Let $\vec{c}\in\mathcal{C}$. The certificate sent to v by u along e_1 is therefore c_1. Let $\vec{c}_{-1}$ be the vector $\vec{c}$ with the first coordinate c_1 omitted, i.e., $\vec{c}_{-1}$ is one dimension less than $\vec{c}$. Using the above notations, we have:

$$\begin{aligned}\Pr[\mathbf{v}\text{ accepts }G_s] &= \sum_{\vec{c}\in\mathcal{C}}\Pr[\text{nodes in }V\text{ send }\vec{c}\text{ on }E] &(4)\\ &= \sum_{\vec{c}\in\mathcal{C}}\Big(\Pr[u\text{ sends }c_1\text{ on }e_1]\cdot\Pr[V\text{ sends }\vec{c}_{-1}\text{ on }E\setminus\{e_1\}]\Big) &(5)\\ &> \sum_{\vec{c}\in\mathcal{C}}\Big(\big(\Pr[u'\text{ sends }c_1\text{ on }(u',v')]-\epsilon\big)\cdot\Pr[V\text{ sends }\vec{c}_{-1}\text{ on }E\setminus\{e_1\}]\Big) &(6)\end{aligned}$$

Eq. (4) is by definitions. Eq. (5) follows from the independence of c_1 from $\vec{c}_{-1}$ by our assumption of edge independence of **v**. Eq. (6) follows from Eq. (1) since u and u' have the same ϵ-rounded distribution over their certificates. Let G'_s be the virtual labeled configuration consisting of G_s labeled by **p** but where the certificate distribution sent by u along e_1 is changed to the distribution of certificates sent by u' along (u',v') in G_s. We get that

$$\begin{aligned}\Pr[\mathbf{v}\text{ accepts }G_s] &= \Pr[\mathbf{v}\text{ accepts }G'_s]-\epsilon\sum_{\vec{c}\in\mathcal{C}}\Pr[V\text{ sends }\vec{c}_{-1}\text{ on }E\setminus\{e_1\}] &(7)\\ &\geq \Pr[\mathbf{v}\text{ accepts }G'_s]-\epsilon\,2^{\kappa} &(8)\end{aligned}$$

Eq. (7) is by definition of G'_s, and Eq. (8) follows from the observation that the second sum in Eq. (7) is at most 2^κ since the number of distinct c_1 values is at most 2^κ, and, for any fixed certificate value γ,

$$\sum_{\vec{c}\in\mathcal{C},c_1=\gamma}\Pr[V\text{ sends }\vec{c}_{-1}\text{ on }E\setminus\{e_1\}]\leq 1.$$

We repeat the same process for the certificate sent along another arc (a,b) of H_i, resulting in a virtual configuration G''_s in which we replace the distribution of the certificates sent by a to b by the distribution of the certificates sent by a' to b', where a' and b' are the respective siblings of a and b in H_j. Again, we get

$$\Pr[\mathbf{v}\text{ accepts }G'_s]\geq\Pr[\mathbf{v}\text{ accepts }G''_s]-\epsilon 2^\kappa.$$

By repeating the process $4s$ times, once for every arc in H_i and in H_j, we eventually get

$$\Pr[\mathbf{v}\text{ accepts }G_s]\geq\Pr[\mathbf{v}\text{ accepts }\sigma_{\bowtie}(G)_s]-4\epsilon s2^\kappa.$$

Moreover, by switching the roles of G_s and $\sigma_{\bowtie}(G)_s$ in the analysis, we also get that,

$$\Pr[\mathbf{v}\text{ accepts }\sigma_{\bowtie}(G)_s]\geq\Pr[\mathbf{v}\text{ accepts }G_s]-4\epsilon s2^\kappa.$$

I.e., $\Pr[\mathbf{v}\text{ accepts }\sigma_{\bowtie}(G)_s]$ and $\Pr[\mathbf{v}\text{ accepts }G_s]$ differ by at most $\pm 4\epsilon s 2^\kappa$. By the choice of ϵ, we conclude that

$$\Pr[\mathbf{v}\text{ accepts }\sigma_{\bowtie}(G)_s]-\frac{1}{3}\;\leq\;\Pr[\mathbf{v}\text{ accepts }G_s]\;\leq\;\Pr[\mathbf{v}\text{ accepts }\sigma_{\bowtie}(G)_s]+\frac{1}{3}\,,$$

which proves Eq. (3), and completes the proof. ∎

E Proofs for Section 5

Proof of Theorem 5.1: The upper bound follows from combining Theorem 3.1 with the proof-labeling scheme for MST in [31], whose verification complexity is $O(\log^2 n)$ (assuming polynomial edge weights). For the lower bound, we will show that any randomized proof-labeling scheme that solves a much simpler problem, namely verifying *acyclicity*, has verification complexities as stated. Let $\mathcal{F}$ be the family of graphs that consist of lines and cycles, i.e., if $G \in \mathcal{F}$ then G is a line or G is a cycle, where all edges have weight 1. Let $P = (u_1, u_2, \ldots, u_n)$ be the n-node path, with port numbers consistently ordered. We define $H_1, \ldots, H_r$, for $r = \left\lfloor \frac{n}{3} \right\rfloor - 1$, as follows. Let $H_1 = \{u_1, u_2\}$. For $i = 2, \ldots, \left\lfloor \frac{n}{3} \right\rfloor - 1$, let $H_i = \{u_{3i}, u_{3i+1}\}$, and $\sigma_i : V(H_1) \to V(H_i)$ satisfying $\sigma_i(u_1) = u_{3i}$ and $\sigma_i(u_2) = u_{3i+1}$. For any $1 \leq i < j \leq \left\lfloor \frac{n}{3} \right\rfloor - 1$, let $\sigma^{ij} = \sigma_j \circ \sigma_i^{-1}$. We get that ${\sigma^{ij}}_{\bowtie}(P)$ consists of removing edges $\{u_{3i}, u_{3i+1}\}$ and $\{u_{3j}, u_{3j+1}\}$ from P, and replacing them by $\{u_{3i}, u_{3j+1}\}$ and $\{u_{3j}, u_{3i+1}\}$, creating the cycle $C = (u_{3i+1}, u_{3i+2}, \ldots, u_{3j-1}, u_{3j})$. Note that ${\sigma^{ij}}_{\bowtie}(P) \notin \mathcal{F}$, but C is a connected component of ${\sigma^{ij}}_{\bowtie}(P)$ and $C \in \mathcal{F}$. Moreover, C is a cycle, i.e., not satisfying the predicate. Therefore, by Theorem 4.7, the verification complexity of $(\mathcal{F}, \mathit{acyclicity})$ is $\Omega(\log \log n)$. Hence, since $\mathcal{F} \subset \mathcal{F}_{con}$, the verification complexity of $(\mathcal{F}_{con}, \mathit{MST})$ is $\Omega(\log \log n)$. ∎

Proof of Theorem 5.2: We first describe a deterministic proof-labeling scheme $(\mathbf{p}, \mathbf{v})$ for $(\mathcal{F}_{con}, \text{v2con})$, based on the algorithm for finding the biconnected components of a graph, presented in [22] (a detailed analysis of this algorithm can be found in [37]). Let $G = (V, E)$ be a 2-vertex-connected graph. The prover $\mathbf{p}$ assigns the following labels to G. Let T be a DFS tree of G. For every node $v \in V$, the label $\ell(v)$ of v consists of the following components:

- id-root(v) - the identity of the root of T.
- dist(v) - the distance of v from the root.
- preo(v) - the pre-order number of v defined by the DFS traversal generating T.
- span(v) - the interval of the pre-order numbers of the nodes in the subtree of T rooted at v.
- lowpt(v) - the smallest pre-order number among the pre-order numbers of the nodes that can be reached from the subtree rooted at v using a back edge.

All the labels are therefore on $O(\log n)$ bits. We call a node u satisfying $\text{dist}(u) = \text{dist}(v) + 1$ a *child* of v. The verifier $\mathbf{v}$ is the logical conjunction of the following predicates, at every node v:

- DFS verification. (P_1) For every neighbor u of v, $\text{id-root}(v) = \text{id-root}(u)$. (P_2) $\text{dist}(v) \geq 0$. (P_3) If $\text{dist}(v) = 0$ then $\text{id-root}(v) = \text{Id}(v)$, and if $\text{dist}(v) > 0$ then v has exactly one neighbor w that satisfies $\text{dist}(w) = \text{dist}(v) - 1$. (P_4) The set of intervals $\{\text{span}(u) \mid u \text{ is a child of } v\}$ is a partition of span(v), not including the pre-order number of v itself. (P_5) For every neighbor u of v, $\text{dist}(u) \neq \text{dist}(v)$. (P_6) For every neighbor u of v, if $\text{dist}(u) < \text{dist}(v)$ then $\text{span}(v) \subset \text{span}(u)$, and if $\text{dist}(u) > \text{dist}(v)$ then $\text{span}(u) \subset \text{span}(v)$.

- lowpt-values verification. Let $\text{child}_{min}(v) = \min\{\text{lowpt}(u) \mid u \text{ is a child of } v\}$ and let $\text{neighbor}_{min}(v) = \min\{\text{preo}(u) \mid u \text{ is a neighbor of } v\}$. (P_7) $\text{lowpt}(v) = \min\{\text{child}_{min}, \text{neighbor}_{min}\}$.

- Bi-connectivity verification. (P_8) If v is the root (i.e., $\text{dist}(v) = 0$), then v has no more than one child, and, if v is not the root, then, for every child u of v, $\text{lowpt}(u) < \text{preo}(v)$.

Let us now prove the correctness of that scheme $(\mathbf{p}, \mathbf{v})$. If $\text{v2con}(G) = \text{TRUE}$, then, by construction, all the above predicates $P_1, \ldots, P_8$ are true at every node, and therefore all nodes returns TRUE, as desired. Conversely, assume that all nodes returns TRUE. From the DFS verification part (predicates $P_1, \ldots, P_6$), and Theorem 1 in [37], it follows that the id-root, dist and span values of all nodes define a proper DFS tree T of G. From the lowpt-values verification part (predicate P_7), and by definition of LOWPT in [37], we derive that the lowpt values are correct according to this DFS tree T. From the bi-connectivity verification part (predicate P_8), and Lemma 5 in [37], there are no articulation points in G. Therefore, $\text{v2con}(G) = \text{TRUE}$. Therefore, $(\mathbf{p}, \mathbf{v})$ is a proof-labeling scheme for $(\mathcal{F}_{con}, \text{v2con})$. The desired randomized proof-labeling scheme follows from Theorem 3.1.

For the lower bounds, let G be a graph that consists of an n-node cycle, with port numbers consistently ordered, and additional edges from one node to all other nodes. (See Figure 2a for illustration). That is, $G =$

$(\{v_0,\ldots,v_{n-1}\}, E_c \cup E_0)$ where

$$E_c = \big\{\{v_i, v_{(i+1) \bmod n}\} \mid i = 0,\ldots,n-1\big\} \text{ and } E_0 = \big\{\{v_0, v_j\} \mid j = 2,\ldots,n-2\big\}.$$

We have v2con$(G) = \textsc{true}$. Now, let $H_1 = \{v_1, v_2\}$. For $i = 2,\ldots,\lfloor\frac{n}{3}\rfloor - 1$, let $H_i = \{v_{3i}, v_{3i+1}\}$, and $\sigma_i : V(H_1) \to V(H_i)$ satisfying $\sigma_i(v_1) = v_{3i}$ and $\sigma_i(v_2) = v_{3i+1}$. For any $1 \le i < j \le \lfloor\frac{n}{3}\rfloor - 1$, let $\sigma^{ij} = \sigma_j \circ \sigma_i^{-1}$. We get that ${\sigma^{ij}}_{\bowtie}(G)$ consists of two disjoint cycles with some edges in E_0 between them, and v_0 is an articulation point. Therefore, v2con$({\sigma^{ij}}_{\bowtie}(G)) = \textsc{false}$. Hence, the conditions of Theorems 4.4 and 4.7 are satisfied, and the lower bounds follow. ▮

Proof of Theorem 5.3: Let $G = (V, E)$ be a graph. Consider the following labeling scheme $(\mathbf{p}, \mathbf{v})$. Let $C = (V', E')$, where $V' \subseteq V$ and $E' \subseteq E$, be a cycle with maximum length in G. Let us index the nodes in V' from 0 to $|V'| - 1$, clockwise, starting from some node $v' \in V'$. For every node $v \in V$, the prover $\mathbf{p}$ assigns label $\ell(v)$ to v, consisting of the following components:

- dist(v): the distance of v from the cycle C.
- index(v): if $v \in V'$, then index(v) is the index of v in C, otherwise index$(v) = 0$.

The verifier $\mathbf{v}$ is defined as the logical disjunction of the following two predicates.

(P_1) dist$(v) = 0$ and v has exactly two neighbors, u_1 and u_2, that satisfy dist$(u_1) = 0$ and dist$(u_2) = 0$. In addition, if index$(v) = i$ then index$(u_1) = i + 1$ (or index$(u_1) = 0$ if $i \ge c - 1$) and index$(u_2) = i - 1$ (or index$(u_2) \ge c - 1$ if $i = 0$).

(P_2) dist$(v) > 0$ and there exists some neighbor u of v that satisfies dist$(u) =$ dist$(v) - 1$.

If cycle-at-least-$c(G) = \textsc{true}$ then by the specified construction, the verifier accepts at all $v \in V$. Conversely, if the verifier accepts at all $v \in V$, then every node v with dist$(v) = d > 0$, has a neighbor u with dist$(u) = d - 1$. By induction, there is at least one node v' with dist$(v') = 0$. Consider such node v', with index$(v') = i$. Node v' has exactly two neighbors, u_1 and u_2, that satisfy dist$(u_1) = 0$, dist$(u_2) = 0$, index$(u_1) = i + 1$ (or index$(u_1) = 0$ if $i \ge c - 1$); and index$(u_2) = i - 1$ (or index$(u_2) \ge c - 1$ if $i = 0$). Therefore, we get a sequence of nodes with dist-values equal to 0, and the following index-values: $(\ldots, 0, 1, 2, \ldots, c_1, 0, 1, \ldots, c_2, 0, \ldots)$, for $c_i \ge c - 1$. Since the graph is finite, and this sequence is infinite, this is a cycle and its size is at least c. Hence, cycle-at-least-$c(G) = \textsc{true}$. This concludes the proof of the proof-labeling scheme. The randomized proof-labeling scheme follows from Theorem 3.1. ▮

Proof of Theorem 5.5: Let G be as described in the lower bound part of the proof of Theorem 5.2 (see Figure 2a), where $n \ge c$. This graph satisfies $G \in \mathcal{F}$ and cycle-at-least-$c(G) = \textsc{true}$. Let $(\mathbf{p}, \mathbf{v})$ be a proof-labeling scheme for cycle-at-least-c over $\mathcal{F}$. Assume, for the purpose of contradiction, that the verification complexity of that scheme is less than $\frac{1}{2}\log(\lfloor\frac{c-1}{3}\rfloor)$. Let $\{H_1, \ldots, H_r\}$, for $r = \lfloor\frac{c-1}{3}\rfloor$, be a set of pairwise independent cycle edges. This set exists since $n \ge c - 1$. For each edge, there are less than $\log(\lfloor\frac{c-1}{3}\rfloor)$ bits in the sequence of both labels of the extremities of the edge. Hence, there are less than $\lfloor\frac{c-1}{3}\rfloor$ possible sequences. Therefore, by the pigeonhole principle, there are two independent cycle edges, say e_1 and e_2, that have exactly the same labels. Recall that the port numbers of the cycle edges are consistently ordered. Hence, for the corresponding isomorphism, $\sigma_{\bowtie}(G)$ consists of two disjoint cycles of size strictly less than n each, with only edges in E_0 between them. Note that possibly $\sigma_{\bowtie}(G) \notin \mathcal{F}$. We apply this crossing inductively as long as there is a cycle of size at least $c - 1$. This process terminates when we eventually get at a graph G' which consists of a set $\mathcal{C}$ of disjoint cycles, all of size less than $c - 1$, with only edges in E_0 connecting them (see Figure 2b). Note that at most two E_0 edges can participate in any simple cycle in G'. Since there are no other connections between the cycles in $\mathcal{C}$, both edges in E_0 have to be connected to the same cycle in $\mathcal{C}$. Therefore, a simple cycle of maximum length in G' is a simple cycle of maximum length in $\mathcal{C}$, with two edges from E_0 instead of just one, and its size is strictly less than c. Hence, $G' \in \mathcal{F}$ and cycle-at-least-$c(G') = \textsc{false}$. On the other hand, the verifier accepts G', a contradiction. This concludes the proof of the theorem. ▮

Proof of Theorem 5.6: Let G be a chain of $\lceil\frac{n}{c}\rceil$ disjoint cycles of c nodes each (except one of at most c nodes), where every two neighboring cycles are connected by an edge (see Figure 5a). We have cycle-at-most-$c(G) =$

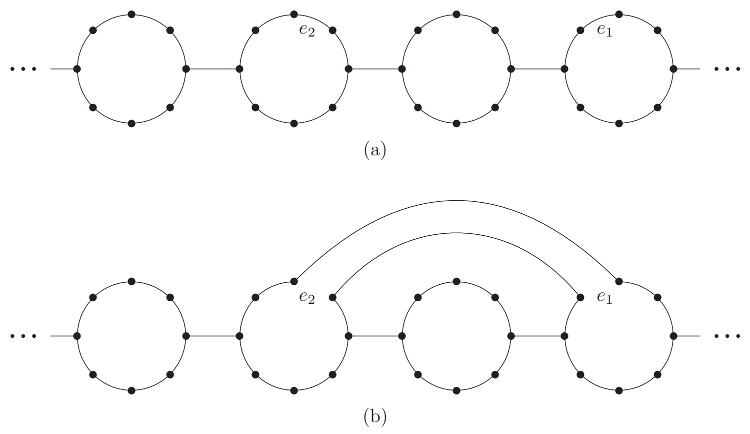

Figure 5: *(a) the graph G in the proof of* Theorem 5.6 *for* $c = 8$. *(b)* $\sigma_{\bowtie}(G)$.

TRUE. Let $\{H_1, \ldots, H_r\}$, for $r = \lceil \frac{n}{c} \rceil$, be a set of pairwise independent edges from distinct cycles. This set exists since there are $\lceil \frac{n}{c} \rceil$ cycles. For every two independent edges from different cycles, say $e_1 = \{u, v\}$ from cycle i and $e_2 = (u', v')$ from cycle j, where $i \neq j$, we have that cycle-at-most-$c(\sigma_{\bowtie}(G)) =$ FALSE for any isomorphism σ (see Figure 5). Therefore, the conditions of Theorems 4.4 and 4.7 are satisfied, and the lower bounds follow. ▮