On Emulating Interactive Proofs with Public Coins*

Oded Goldreich and Maya Leshkowitz
Department of Computer Science,
Weizmann Institute of Science, Rehovot, Israel.

April 18, 2016

Abstract

The known emulation of interactive proof systems by public-coins interactive proof systems proceeds by selecting, at each round, a message such that each message is selected with probability that is at most polynomially larger than its probability in the original protocol. Specifically, the possible messages are essentially clustered according to the probability that they are selected in the original protocol, and the emulation selects a message at random among those that belong to the heaviest cluster.

We consider the natural alternative in which, at each round, if the parties play honestly, then each message is selected with probability that approximately equals the probability that it is selected in the original protocol. This is done by selecting a cluster with probability that is proportional to its weight, and picking a message at random in this cluster. The crux of this paper is showing that, essentially, no matter how the prover behaves, it cannot increase the probability that a message is selected by more than a constant factor (as compared to the original protocol). We also show that such a constant loss is inevitable.

*This research was partially supported by the Minerva Foundation with funds from the Federal German Ministry for Education and Research.

Contents

1 Introduction

The notion of interactive proof systems was introduced by Goldwasser, Micali, and Rackoff [GMR85] in order to capture the most general way in which one party can efficiently verify claims made by another, more powerful party. Interactive proofs generalize and contain as a special case the traditional NP-proof systems. However, we gain a lot from this generalization: the $\mathcal{IP}$ *Characterization Theorem* of Lund, Fortnow, Karloff, Nisan and Shamir [LFKN92, Sha92] states that every language in $\mathcal{PSPACE}$ has an interactive proof system.

An interactive proof system is a two-player protocol between a computationally bounded verifier, and a computationally unbounded prover whose goal is to convince the verifier of the validity of some claim. The verifier employs a probabilistic polynomial time strategy and sends the prover messages, to which the prover responds in order to convince the verifier. It is required that if the claim is true then there exists a prover strategy that causes the verifier to accept with high probability, whereas if the claim is false then the verifier rejects with high probability (no matter what strategy the prover employs). A formal definition of an interactive proof system is provided in Section 2. The class of sets having an interactive proof system is denoted by $\mathcal{IP}$.

Public coins versus private coins. An important aspect of interactive proofs is the verifier's randomness. Whereas we can assume, without loss of generality, that the prover is deterministic, the verifier must be randomized to benefit from the power of interactive proofs. Specifically, without randomness on the verifier's side, interactive proof systems exist only for sets in $\mathcal{NP}$. The verifier's messages in a general interactive proof system are determined based on the input, the interaction preformed so far, and the its internal coin tosses (i.e., the verifier's coin tosses). In that case, we may assume, without loss of generality, that the verifier tosses all coins at the very beginning of the interaction, and it is crucial that (with the exception for the last message) the verifier's messages only reveal partial information about its coins (and keep the rest secret). In contrast, in *public-coin* proof systems, introduced by Babai [Bab85] as *Arthur-Merlin games*, the message sent by the verifier in each round contains (or totally reveals) the outcome of all coin it has tossed at the current round. Thus, these messages reveal the randomness used toward generating them; that is, this randomness becomes public. The class of sets having an interactive *public coin* proof system is denoted $\mathcal{AM}$.

The relative power of *public coin* interactive proofs as compared to general interactive proofs was first studied by Goldwasser and Sipser [GS86], who showed that every interactive proof can be emulated using only public coins; hence, $\mathcal{IP} = \mathcal{AM}$. Intuitively, this means that, in order to test the prover, the verifier does not need to ask clever questions, which hide some secrets, but it rather suffices to ask random questions (which hide nothing). The fact that $\mathcal{IP} = \mathcal{AM}$ also follows from the *IP characterization theorem* of [LFKN92, Sha92], since the proof actually establishes $\mathcal{PSPACE} \subseteq \mathcal{AM}$, whereas $\mathcal{IP} \subseteq \mathcal{PSPACE}$.

A finer notion of interactive proofs refers to the number of prover–verifier communication rounds. For an integer function r, the complexity class $\mathcal{IP}(r)$ consists of sets having an interactive proof system in which, on common input x, at most $r(|x|)$ rounds of communication take place. The original proof of Goldwasser and Sipser that $\mathcal{IP} = \mathcal{AM}$ actually provides a *round efficient* emulation of $\mathcal{IP}$ by $\mathcal{AM}$. Specifically, they show that, for any polynomially bounded function $r : \mathbb{N} \to \mathbb{N}$, it holds that $\mathcal{IP}(r) \subseteq \mathcal{AM}(r+2)$.

In addition to being of intrinstic interest, the emulation of general interactive proofs by public-

coin interactive coins is instrumental for several fundamental results regarding general interactive proof systems, which are established by reducing them to the analogous results regarding *public coin* interactive coin systems. Examples include the round-reduction (a.k.a. speed-up) theorem of Babai and Moran asserting that $\mathcal{IP}(2r) \subseteq \mathcal{IP}(r)$, the zero-knowledge emulation asserting that $\mathcal{IP} = \mathcal{ZK}$ (provided that one-way functions exist), and the equivalence between one-sided and two-sided error versions of interactive proof systems. In all three cases, the result is easier to establish for *public coin* interactive proof systems (see [BM88], [BGGHKMR], and [FGMSZ], respectively); actually, no "direct proof" that works with arbitrary interactive proof systems is known (and it is even hard to imagine one). We stress that the use of a round-efficient emulation (of general interactive proofs by public coin ones) means that taking this ("via AM") route incurs no cost in terms of the round complexity of the resulting proof systems.

1.1 The known emulation of $\mathcal{IP}$ by $\mathcal{AM}$

The basic idea used in emulating a general interactive proof by a public coin one is changing the assertion, from proving that **one** (random) interaction using a specific sequence of private coins leads the verifier to accept, to proving that **most** of the sequences of coin tosses lead the verifier to accept. Calling such coin sequences **good**, the claim that there are many good coin sequences for a potential r-round interaction reduces to showing that the product of the number of verifier-messages (for the first round) times the number of good coin sequences that are consistent with each of these messages (and some prover response to it) is large. Hence, lower-bounding the number of good sequences for the r-round interaction is reduced to lower-bounding the number of good sequences for the remaining $r-1$ rounds.

The foregoing description makes sense when the next verifier message is uniformly distributed in some set, denoted S. In this case, the claim that there are M good coin sequences for the r-round interaction reduces to asserting that there are $|S|$ verifier messages such that each of them yields a $(r-1)$-round interaction with $M/|S|$ good coin sequences. The problem is that the foregoing uniformity condition may not hold in general.

Goldwasser and Sipser, who suggested this emulation strategy, resolved the foregoing problem by picking a set of messages that have roughly the same number of good coin sequences. Specifically, they *clustered* the potential messages that the original verifier could have sent on the next round into *clusters* according to the (approximate) number of good coin sequences that support each message. A constant-round, public-coin sampling protocol is utilized in order to sample from the cluster of messages that have the largest number of good coin sequences. Hence, the **chosen cluster** is determined as the "heaviest" one. (We go over the original emulation in more detail in Section 2.) The emulation succeeds assuming an initial *gap* between the number of good coin sequences for yes-instances and for no-instances. We provide a somewhat unorthodox phrasing of the $\mathcal{IP} = \mathcal{AM}$ theorem in terms of the initial gap; that is, the ratio between the completeness and soundness bounds (i.e., the ratio between the lower bound on the acceptance probabuility of yes-instances and the upper bound on the acceptance probabuility of no-instances).

Theorem 1 (Original emulation of $\mathcal{IP}$ by $\mathcal{AM}$ [GS86]) *Suppose that L has a $r = r(|x|)$ round interactive proof system that utilizes $n = n(|x|)$ random coins for an instance x, and a gap of $\Omega(n)^r$ between the number of accepting coins of yes-instances and no-instances. Then, the foregoing emulation yields a public-coin interactive system proof for L.*

1.2 Our contribution

We propose an alternative method for preforming a public-coin emulation of $\mathcal{IP}$. Our method is similar to the original method of [GS86], but differs in the way the **chosen cluster** of messages (from which the sampling is preformed) is determined. Whereas in the original emulation the **chosen cluster** is determined as the one with the largest number of coins, in our emulation the **chosen cluster** is selected probabilistically according to its weight (i.e., the number of good coins in the cluster). Therefore, this method gets closer to sampling from the real distribution of prover-verifier transcripts (see farther discussion in Section 1.3). Furthermore, as explained in Section 2, while the original method looses a factor of $\Theta(n)$ (in the said gap) in each round, the new method only looses a constant factor. Consequently, this method requires a smaller initial gap between the number of accepting coins of yes-instances and no-instances (in order to emulate interactive proofs using public coins).

Theorem 2 (New emulation of $\mathcal{IP}$ by $\mathcal{AM}$) *Suppose that L has a $r = r(|x|)$ round interactive proof system for an instance x, and a gap of B^r, for some universal constant $B > 1$, between the number of accepting coins of yes-instances and no-instances. Then, the new emulation yields a public coin interactive proof system for L.*

We present the emulation and the proof of Theorem 2 in Section 3.

We further show that, for the new emulation, the gap that we use is asymptotically tight. Namely, when the initial gap is $O(C^r)$ for some constant $C > 1$, we provide an interactive proof and a prover strategy that fails the new emulation.

Theorem 3 (Tightness of Theorem 2) *For some universal constant $C > 1$, there exists an interactive proof system for a set L that proceeds in $r = r(|x|)$ rounds and has a gap of $\Omega(C^r)$ between the number of accepting coins of yes-instances and no-instances such that emulating this proof system* (as described above) *fails to yield an interactive proof system for L.*

We provide the proof of Theorem 3 in Section 3.3.

1.3 An alternative perspective

As stated in Section 1.2, the new emulation can be viewed as an attempt to tightly emulate the original prover-verifier interaction. When choosing a cluster according to its weight, and sampling a message uniformly from this cluster, we are actually selecting a verifier-message with distribution that is quite close to the original, where the deviation is due to approximation that underlies the definition of a cluster (i.e., each cluster contains messages that have approximately, but not necessarily exactly, the same number of coins suppoprting them). Furthermore, essentially, malicious behavior of the prover can increase the probability that a specific message is chosen in a specific round by at most a constant factor as compared to the original interaction.

In contrast, the previous emulation strategy (of Goldwasser and Sipser [GS86]) selects messages with a distribution that is very far from the original interaction, even in the case that both parties are honest. Recall that this emulation always selects messages from the heaviest cluster, and so it may increase the probability that such a message is chosen in a certain round by a factor of $\Theta(n)$. Hence, our contribution is in showing that the new emulation strategy works too, and in fact that it works better. In particular, while the analysis of Goldwasser and Sipser [GS86] shows that their

emulation strategy loses a factor of $O(n)$ in each round, we show that the new emulation strategy loses a constant factor in each round (and that such a factor must be lost).

We comment that choosing clusters according to their weight was also employed by Goldreich, Vadhan, and Wigderson [GVW02], but in their work several such clusters are selected at each round, which makes the analysis of the protocol easier. We cannot afford doing so.

2 Preliminaries

Let us start by providing a formal definition of an interactive proof system, where the completeness and soundness bounds are parameters.

Definition 4 (Interactive Proof Systems) *Let $c, s : \mathbb{N} \to [0,1]$. An interactive proof system for a set S is a two party game, between a verifier executing a probabilistic polynomial time verifier strategy, denoted V, and a prover executing a* (computationally unbounded) *strategy satisfying the following two conditions:*

- *Completeness with bound c: For every $x \in S$, the verifier V accepts after interacting with the prover P on common input x with probability at least $c(|x|)$.*
- *Soundness with bound s: For every $x \notin S$ and every prover strategy P^*, the verifier V accepts after interacting with P^* on common input x with probability at most $s(|x|)$.*

When c and s are not specified, we mean $c \equiv 2/3$ and $s \equiv 1/3$. We denote by $\mathcal{IP}$ the class of sets having interactive proof systems.

A finer definition of interactive proofs refers to the number of prover-verifier communication rounds (i.e., number of pairs of verifier-message followed by a prover-message). For an integer function r, the complexity class $\mathcal{IP}(r)$ consists of sets having an interactive proof system in which on common input x, at most $r(|x|)$ rounds of communication are executed between the parties.

2.1 Accepting coins

In order to provide a precise description of the original and new emulations, we formally define the set of *accepting coins* for input x and partial transcript γ. The following definition refers to any fixed pair of deterministic strategies, (P, V), where V is provided with an auxiliary input ρ (which represents the outcomes of coin tosses). When using the following definition in the rest of this paper, we shall always fix V to be the verifier strategy given to us (where the verifier's internal copin tosses are viewed as input to V) and let P be a fixed optimal strategy that maximizes the acceptance probability of V.

Definition 5 (Accepting coins) *Let us denote by $\langle P, V(\rho)\rangle(x)$ the full transcript of the interaction of P and V on input x, when V uses coins ρ; that is,*

$$\langle P, V(\rho)\rangle(x) = (\alpha_1, \beta_1, \ldots, \alpha_r, \beta_r, (\sigma, \rho)) \tag{1}$$

where $\sigma = V(x, r, \beta_1, \ldots, \beta_r) \in \{0,1\}$ is V's final verdict and for every $i = 1, \ldots, r$ it holds that $\alpha_i = V(x, \rho, \beta_1, \ldots, \beta_{i-1})$ and $\beta_i = P(x, \alpha_1, \ldots, \alpha_i)$. For any partial transcript ending with a

P-message, $\gamma = (\alpha_1, \beta_1, \ldots, \alpha_{i-1}, \beta_{i-1})$, we denote by $ACC_x(\gamma)$ the set of coin sequences that are consistent with the partial transcript γ and lead V to accept x when interacting with P. Formally

$$ACC_x(\gamma) = \left\{\rho \in \{0,1\}^n : \exists \gamma' \in \{0,1\}^{\mathrm{poly}(|x|)} \text{ s.t } \langle P, V(\rho)\rangle(x) = \left(\gamma, \gamma', (1,\rho)\right)\right\} \tag{2}$$

When x and γ are clear from the context we refer to $ACC_x(\gamma)$ as the set of accepting coins.

Note that we assume, without loss of generality, that the verifier reveals its private coins ρ on the last round, which also includes its output (or verdict) bit. (In Eq. (2), we mandated an accepting verdict.)

2.2 The original emulation

In the original proof of $\mathcal{IP} = \mathcal{AM}$, the public coin emulation was preformed by clustering the possible messages the verifier can send on each round into n clusters according to the approximate number of accepting coins they have, that is, according to $|ACC_x(\gamma)|$. In [GS86], the i^{th} cluster contained messages with approximately 2^i accepting coins, but (mainly for clarity) we prefer to use a generic (constant) basis $b > 1$ (while noting that a choice of $b = 2$ is quite good). Thus, we shall use $n' \stackrel{\text{def}}{=} n/\log_2 b = \Theta(n)$ clusters (rather than n clusters). Thus, for the emulation of round r' with partial transcript γ we denote these clusters by $C_0, \ldots, C_{n'}$, where C_i is defined as

$$C_i = \left\{\alpha : b^i \leq |ACC_x(\gamma\alpha)| < b^{i+1}\right\} \tag{3}$$

Namely, C_i is the set of messages α that the verifier can send (on round r') that have approximately b^i coins that are consistent with the transcript $\gamma\alpha$, and lead the verifier to accept.

The original emulation proceeds as follows. Denote by c the completeness parameter of the interactive proof system. The prover's initial claim is that there are at least $c \cdot 2^n$ accepting coins for x i.e, that $|ACC_x(\emptyset)| \geq c \cdot 2^n$. The prover supplies the verifier with the sizes of the clusters $|C_0|, \ldots, |C_{n'}|$. The verifier checks that the number of accepting coins approximately sums up to the claim (namely, that $\sum_{i=0}^{n'} |C_i| \cdot b^{i+1} > c \cdot 2^n$), and chooses the cluster C_i with the largest number of accepting coins; that is, i is chosen so as to maximize $b^i \cdot |C_i|$. In order to validate that the claim is true, and to sample a message α from C_i, the prover and the verifier run a (constant-round) sampling protocol which utilizes only public coins. Next, the prover supplies its answer β to the sampled message α and the parties proceed to the next round, where the prover claims that there are at least 2^i accepting coins that are consistent with the interaction $\alpha\beta$ preformed so far. After the last round the complete prover-verifier transcript is determined, which also contains the verifier's internal coins tosses. The verifier then checks that the entire transcript is consistent and accepting.

We note that throughout the emulation the verifier does not "challenge" the prover on the number of accepting coins in the clusters other than the selected cluster C_i and the prover can use this to employ a strategy for fooling the verifier. For example, even if all of the accepting coins lie in cluster C_i, the prover can claim that there are $|C_i| - 1$ coins in each other cluster, and get away with this lie. In this way the gap between the number of accepting coins consistent with the interaction and the prover's claim regarding this number is cut by a factor of $\Theta(n)$ in each round. For this reason, the emulation requires an initial gap of $\Theta(n)^r$ between yes-instances and no-instances, where r is the number of rounds of the original interactive proof.

3 The new emulation

As mentioned in Section 2.2, an essential cause for the large initial gap required in the $\mathcal{AM}$ emulation of [GS86] is the deterministic way in which a cluster of messages is chosen by the verifier. Therefore, a promising approach is to have the verifier choose a cluster with probability proportional to the number of accepting coins the prover claims are in that cluster. This follows the intuition that we would like to challenge the prover by choosing "heavy" clusters, which contain many accepting coins, with higher probability than "lighter" clusters. The same intuition also underlies [GS86], but we apply it in a more smooth fashion.

We note that the prover still has a potential of fooling the verifier by supplying a message that does not belong to C_i but rather to some other cluster, when C_i is chosen. Nevertheless, we show that even an untrusted prover will not be able to fool the verifier too much.

3.1 The actual protocols

The original r-round interaction (P, V) is "emulated" in r iterations (each consisting of a constant number of message exchanges). The i^{th} iteration starts with a partial prover-verifier interaction $\gamma_{i-1} = (\alpha_1\beta_1 \dots \alpha_{i-1}\beta_{i-1})$ and a claimed bound M_{i-1} regarding the size of $ACC_x(\gamma_{i-1})$. In the first iteration γ_0 is the empty sequence and $M_0 = c \cdot 2^n$, where $c > 0$ is the completeness parameter of the interactive proof system. The i^{th} iteration proceeds as follows.

Construction 6 (The i^{th} iteration) *On input γ_{i-1} and M_{i-1}.*

1. *The prover computes the number of messages in each cluster, and sends the sizes of the clusters $N_0, \dots, N_{n'}$ to the verifier, where N_j is the number of messages in cluster C_j defined as in Eq. (3).*

   Recall that each message in cluster C_j has between b^j and b^{j+1} consistent and accepting coins.

2. *Verifier's initial checks: If $\sum_{j=0}^{n'} N_j \cdot b^{j+1} < M_{i-1}$, then the verifier aborts and rejects.*

3. *Verifier's selection of clusters: The verifier samples a cluster j according to the probability distribution J that assigns $j \in [n]$ probability proportional to $b^j \cdot N_j$. That is,*

$$\mathbf{Pr}[J = j] = \frac{N_j \cdot b^j}{\sum_{\ell=0}^{n'} N_\ell \cdot b^\ell} \tag{4}$$

4. *Sampling the selected cluster: The verifier and the prover run a sampling protocol* (as defined below) *to obtain a message α_i which the prover claims is in cluster C_j. The protocol is invokes with completeness parameter $\epsilon = \frac{1}{3r}$ and soundness parameter $\delta = b$.*

   (If not output is provided by the sampling protocol, then the verifier rejects.)

5. *Completing the current iteration: Next, the prover determines β_i such that $ACC_x(\gamma_{i-1}, \alpha_i, \beta_i) = ACC_x(\gamma_{i-1}, \alpha_i)$; that is, the prover selects a message that maximized the number of accepting coins, and sends it to the verifier.*
   Toward the next iteration, the parties set $M_i = 2^j$ and $\gamma_i = \gamma_{i-1}\alpha_i\beta_i$.

By our conventions, the last message the verifier sends contains the outcomes $\rho \in \{0,1\}^n$ of the n coins tossed. Thus, ρ can be easily extracted from $\gamma_r = (\alpha_1, \beta_1, \ldots, \alpha_r, \beta_r, (1, \rho))$. After the last iteration the verifier performs **final checks** and accepts if all of them hold:

i) Checking that ρ is accepting for γ_r: $V(x, \rho, \alpha_1, \ldots, \alpha_r) = 1$, and for every $i = 1, \ldots, r$ it holds that $\alpha_i = V(x, \rho, \beta_1, \ldots, \beta_{i-1})$. Note that the verifier needs ρ in order to verify these conditions, so it can only be done after the last iteration. Also note that if these checks pass then $|ACC_x(\gamma_r)| = 1$.

ii) Checking that $M_r = 1$; namely, checking that the prover's last claim was that there is a single sequence of coin tosses that is consistent with the complete interaction γ_r.

The sampling protocol used. Our protocol utilizes a constant-round, public-coin sampling protocol for sampling in arbitrary sets. The verifier is assisted by a computationally unbounded prover that the verifier does not trust. The prover provides the verifier with an integer N, which is supposed to be a lower bound on the size of the set (in our case the set of messages) denoted $S \subseteq \{0,1\}^\ell$. (We assume for simplicity that the length of the verifier's messages is exactly $\ell = \text{poly}(|x|)$ (which can be justified by padding the messages to be of size ℓ).) The sampling protocol with parameters $\epsilon > 0$ and $\delta > 1$, satisfies the following two properties:

Completeness (w.r.t ϵ)**:** If the lower bound on $|S|$ is valid (i.e. $|S| \geq N$), and the prover is honest, then with probability $1 - \epsilon$, the verifier will output an element of S.

Soundness (w.r.t δ)**:** For every T such that $|T| < N$, no matter how the prover plays, the probability that verifier will output an element of T is at most $\delta \cdot \frac{|T|}{N}$.

For the implementation we use families of pairwise independent hash functions $\left\{H_\ell^t\right\}_{\ell > t}$. The sampling protocol proceeds as follows.

Construction 7 (The sampling protocol) *Using parameters $\epsilon > 0$ and $\delta > 1$, on input ℓ and N, the parties proceed as follows.*

i) The verifier selects and sends the prover a random hash function $h : \{0,1\}^\ell \to \{0,1\}^t$, where $t = \lfloor \log_2(\epsilon N \rfloor - \lceil 2\log_2(\delta/(1-\delta)) \rceil$, and a random element from the image $y \in \{0,1\}^t$.

ii) The prover is supposed to answer with $K \stackrel{\text{def}}{=} \lfloor 2^{-t} N/\delta \rfloor$ elements of S that are preimages of y under h; that is, with $x_1, ..., x_K \in S$ such that $h(x_i) = y$ for every i.

iii) The verifier checks that the K elements are indeed preimages of y under h. Next, the verifier selects i uniformly in $[K]$ and outputs x_i; that is, it outputs one of these K elements selected uniformly using public randomness.

(If less than K elements are provided, or some of the elements are not preimages, then the verifier has no output).

The computational complexity of the protocol for the verifier is polynomial in ℓ/ϵ, since $K = 2^{-t} N/\delta = O_\delta(1/\epsilon)$, and the verifier's actions can be implemented in $\text{poly}(\ell) \cdot K$-time.

Lemma 8 (analysis of the sampling protocol) *For any constant $\delta > 1$ and all sufficiently small $\epsilon > 0$, the protocol of Construction 7 satisfies the foregoing completeness and soundness conditions.*

Proof: We start with the completeness condition. The family of pairwise independent hash functions satisfies an "almost uniform cover" condition (cf. [Gol08, Lem. D.4]); that is, for every $S \subseteq \{0,1\}^\ell$ and every $y \in \{0,1\}^t$, for all but at most a $\frac{2^t}{(1-(1/\delta))^2 \cdot |S|}$ fraction of $h \in H_\ell^t$ it holds that

$$|\{x \in S : h(x) = y\}| > \frac{|S|}{\delta \cdot 2^t}$$

(since the expected size of the set is $|S|/2^t$ and $\delta > 1$). On the other hand, using $|S| \geq N$, we have $K = \lfloor 2^{-t} N/\delta \rfloor \leq 2^{-t}|S|/\delta$. Hence the prover will fail in supplying K preimages with probability of at most

$$\begin{aligned} \frac{2^t}{(1-(1/\delta))^2 \cdot |S|} &\leq \frac{\delta^2 \cdot 2^t}{(\delta-1)^2 N} \\ &\leq \epsilon \end{aligned}$$

since $t \leq \log_2(\epsilon N) - 2\log_2(\delta/(\delta-1))$.

Turning to the soundness condition, we consider an arbitrary set $T \subseteq \{0,1\}^\ell$. Let Y be a random variable denoting the "cell" the verifier chooses (i.e., the set $h^{-1}(y)$). For every $y \in \{0,1\}^t$, denote by T_y the set of preimages of y under h that are in T; that is, $T_y \stackrel{\text{def}}{=} \{\alpha \in T : h(\alpha) = y\}$. Then, it holds that $\sum_{y \in \{0,1\}^t} |T_y| = |T|$. In Step (ii), the prover provides K preimages (of y under h), some of them may be in T, and the verifier selects one of them, which we denote by z. Hence, for y with $|T_y|$ preimages in T, the probability that the sampled element resides in T is at most $\frac{|T_y|}{K}$ (it may be less if the the prover does not provide all the elements in T_y, for example when $|T_y| > K$, or if the prover just acts "foolishly"). Hence the probability that the output z is in T is at most

$$\begin{aligned} \mathbf{Pr}[z \in T] &= \sum_{y \in \{0,1\}^t} \mathbf{Pr}[Y = y \wedge z \in T_y] \\ &= \sum_{y \in \{0,1\}^t} \mathbf{Pr}[Y = y] \cdot \mathbf{Pr}[z \in T_y] \\ &\leq \sum_{y \in \{0,1\}^t} \frac{1}{2^t} \cdot \frac{|T_y|}{K} \\ &= \frac{|T|}{K \cdot 2^t} \\ &\leq \frac{|T|}{((2^{-t} \cdot N/\delta) - 1) \cdot 2^t} \\ &= \delta \cdot \frac{|T|}{N - \delta \cdot 2^t} \end{aligned}$$

which is approximately $\delta \cdot |T|/N$. Actually, since $N > 2^t/\epsilon$, we get $\delta \cdot \frac{|T|}{N - \delta \cdot 2^t} = \frac{\delta}{1-\delta\epsilon} \cdot \frac{|T|}{N}$, which means that the claim holds for soundness parameter $\frac{\delta}{1-\delta\epsilon}$. (The original claim follows by substituting δ for $(1-2\epsilon) \cdot \min(\delta, 2)$.) ■

The round complexity of the emulation. In the Construction 6, the prover sends messages in Steps (1), (4) and (5), while the verifier sends messages in Steps (3) and (4), where Step (4) invokes the three-message protocol of Construction 7 (in which the verifier sends messages in Steps (i) and (iii), and the prover sends a message in Step (ii)). Denoting these messages by the sender's initial and the step number, we get the sequence `P1, V3, V4i, P4ii, V4iii, P5`, which means that we have two and a half rounds. It is possible to avoid this blowup in the number of rounds by combining the message sent by the prover in Step (ii) of the sampling protocol with its Step (5) message and the Step (1) message of the next iteration in one message. This is possible since the prover can provide the messages that it would have sent for each of the K possible messages of the verifier in Step (iii) of the sampling protocol. Details follow.

Recall that in Step (ii) of the sampling protocol the prover sends K messages allegedly belonging to C_j, and the verifier selects and sends one of these messages, denoted α_i, in Step (iii). The idea is to have the prover then provide its response β_i, to each of these possible α_i as well as and the sizes of the clusters for the next round. All these messages are sent in one new message that the prover sends in a Step (ii) of the modified protocol. So the sequence of messages has the form `V3+V4i, P4ii, V4iii`, where the possible `P5`-messages of the current iteration as well as the possible `P1`-messages of the next iteration are included in the `P4ii`-message. Lastly, the `V4iii`-message of the i-th iteration is combined with the `V3+V4i`-message of the $i+1^{\text{st}}$ iteration. Hence an r-round interactive proof system is emulated by an $(r+1)$-rounds public-coin interactive proof system.

3.2 Analysis of the emulation

We introduce some notation and terminology that will be useful for the analysis of the proposed emulation. Fixing a generic input x and letting $n = n(|x|)$, we consider an interactive proof system with completeness and soundness parameters $c = c(|x|)$ and $s = s(|x|)$, respectively. Hence if x is yes-instance (resp., a no-intance), then it has at least $c \cdot 2^n$ accepting coins (resp., at most $s \cdot 2^n$ accepting coins). Put differently, there is a **gap** of $g_0 \stackrel{\text{def}}{=} \frac{c}{s}$ between the number of accepting coins of yes-instances and no-instances. In the each iteration the prover's goal is to *lower the gap* regarding the number of accepting coins. We refer to the following definition.

Definition 9 (Gaps) *The **gap on the** i^{th} **iteration,** denoted g_i, is the ratio between the claimed bound regarding to the number of accepting coins on the i^{th} round, i.e. M_i, and the number of accepting coins consistent with the partial transcript γ_i, i.e., $|ACC_x(\gamma_i)|$. In case $|ACC_x(\gamma_i)| = 0$ we set $g_i = \infty$. That is,*

$$g_i = \begin{cases} \frac{M_i}{|ACC_x(\gamma_i)|} & \text{if } |ACC_x(\gamma_i)| > 0 \\ \infty & \text{otherwise} \end{cases} \tag{5}$$

Indeed, if the prover claims that some no-instance is a yes-instance, then at the beginning of the emulation $M_0 \geq c \cdot 2^n$ and $|ACC_x(\gamma_0)| \leq s \cdot 2^n$, thus $g_0 \geq \frac{c}{s}$. If the verifier accepts the complete emulation, then (in particular) the final checks pass and $M_r = |ACC_x(\gamma_r)| = 1$, thus $g_r = 1$.

3.2.1 The effect of a single iteration

Recall that we have fixed an arbitrary interactive proof system (P, V), and an input x to it. We consider the public coin emulation of (P, V) defined in Section 3.1, and fix an interaction index $i \in [r]$ as well as the transcript of the first $i-1$ iterations. Hence, the values γ_{i-1}, g_{i-1} and M_{i-1} are

fixed. Denote by G_i the random variable that represents g_i at the end of the i^{th} iteration, which is a function of the public randomness of the emulation protocol (of Construction 6 and the sampling protocol of Construction 7). Towards proving Theorem 2, we analyze the change in the gap on the i^{th} iteration, and show that for every $t \in \mathbb{N}$ the gap G_i is reduced by a factor of b^{-t} with probability at most $O(b^{-t})$. It is convenient to prove this claim by letting $j \in \mathbb{N}$ be such that $g_{i-1} \in (b^{j-1}, b^j]$. Hence if $G_i \in (b^{j-t-1}, b^{j-t}]$, this implies that the gap changed by a factor of approximately b^{-t}. The following lemma shows the probability that the gap changed by some factor F can be bounded in a way that is independent of the previous gap, and depends only on the factor F.

Lemma 10 (Main lemma) *Suppose that $g_{i-1} \in (b^{j-1}, b^j]$ and $j > t$. Then,*

$$\mathbf{Pr}\left[G_i \in (b^{j-t-1}, b^{j-t}]\right] \leq b^{-t+3}.$$

Proof: Recall that G_i is defined as the random variable representing the gap g_i, which is the ratio between the number of accepting coins that are consistent with the emulation according to the prover, and the actual number of accepting coins. The gap G_i is determined by the cluster the verifier chooses in Step (3), and by the cluster that the message sampled in Step (4) of the emulation resides in. We are interested in calculating the probability that $G_i \in (2^{j-t-1}, 2^{j-t}]$ for $j > t$. We can write this event as the union of disjoint events regarding to the cluster C_k that the verifier chooses in Step (3) of the emulation.

$$\mathbf{Pr}\left[G_i \in (b^{j-t-1}, b^{j-t}]\right] = \sum_{k=0}^{n'} \mathbf{Pr}\left[C_k \text{ is chosen} \wedge G_i \in (b^{j-t-1}, b^{j-t}]\right] \tag{6}$$

Assume that cluster C_k is chosen by the verifier, which implies that $M_i = b^k$. Recalling that $G_i = \frac{M_i}{|ACC_x(\gamma_{i-1}\alpha_i)|}$, it holds that if $G_i \in (b^{j-t-1}, b^{j-t}]$, then

$$b^{j-t-1} < \frac{b^k}{|ACC_x(\gamma_{i-1}\alpha_i)|} \leq b^{j-t}$$

or equivalently

$$b^{k-(j-t)} \leq |ACC_x(\gamma_{i-1}\alpha_i)| < b^{k-(j-t)+1}$$

In other words, $G_i \in (b^{j-t-1}, b^{j-t}]$ if and only if the sampled message α_i resides in $C_{k-(j-t)}$ and $k \geq j - t$. For each $k \in \{0, \ldots, n\}$, we introduce the following Boolean indicator variables:

Y_k**:** The event that cluster C_k is chosen by the verifier in Step (3).

Z_k**:** The event that the sampled message in Step (4) resides in cluster C_k

Using the aforementioned observation and the new notations introduced, we can write Eq. (6) as

$$\mathbf{Pr}[G_i \in (b^{j-t-1}, b^{j-t}]] = \sum_{k=j-t}^{n'} \mathbf{Pr}[Y_k \wedge Z_{k-(j-t)}] \tag{7}$$

Next, we calculate the probabilities that the events in Eq. (7) occur. We first note that the verifier chooses a cluster according to the distribution in Eq. (4), hence

$$\mathbf{Pr}\left[Y_k\right] = \frac{N_k \cdot b^k}{\sum_{\ell=0}^{n'} N_\ell \cdot 2^\ell} \tag{8}$$

Assume that cluster C_k was chosen by the verifier, which the prover claims is of size N_k. We can use the soundness property of the sampling protocol (with $T = C_\ell$ and $N = N_k$) to upper bound the probability that the sampled message resides in C_ℓ.

$$\mathbf{Pr}[Z_\ell \,|\, Y_k] \leq \frac{b \cdot |C_\ell|}{N_k} \tag{9}$$

(since the soundness parameter δ was set to b). Combining Equations (8) and (9), we get

$$\begin{aligned}
\mathbf{Pr}[Y_k \wedge Z_{k-(j-t)}] &= \mathbf{Pr}[Y_k] \cdot \mathbf{Pr}[Z_{k-(j-t)} \,|\, Y_k] \\
&\leq \frac{N_k \cdot b^k}{\sum_{\ell=0}^{n'} N_\ell \cdot 2^\ell} \cdot \frac{b \cdot |C_{k-(j-t)}|}{N_k} \\
&= \frac{b^{k+1} \cdot |C_{k-(j-t)}|}{\sum_{\ell=0}^{n'} N_\ell \cdot 2^\ell} \\
&= \frac{b^{j-t+1} \cdot b^{k-(j-t)} \cdot |C_{k-(j-t)}|}{\sum_{\ell=0}^{n'} N_\ell \cdot 2^\ell}
\end{aligned} \tag{10}$$

Note that this quantity does not depend on N_k, which is the purported size of the cluster C_k as claimed by the prover. Moreover, Eq. (10) is proportional to the number of coins in the cluster $C_{k-(j-t)}$, which is approximately $b^{k-(j-t)} \cdot \left|C_{k-(j-t)}\right|$. Hence, plugging in the quantity from Eq. (10) in Eq. (7), we get

$$\begin{aligned}
\mathbf{Pr}[G_i \in (b^{j-t-1}, b^{j-t}]] &\leq \sum_{k=j-t}^{n'} \frac{b^{j-t+1} \cdot b^{k-(j-t)} \cdot |C_{k-(j-t)}|}{\sum_{\ell=0}^{n'} N_\ell \cdot 2^\ell} \\
&= \frac{b^{j-t+1}}{\sum_{\ell=0}^{n'} N_\ell \cdot 2^\ell} \cdot \sum_{k=j-t}^{n'} |C_{k-(j-t)}| \cdot b^{k-(j-t)} \\
&= \frac{b^{j-t+1}}{\sum_{\ell=0}^{n'} N_\ell \cdot 2^\ell} \cdot \sum_{\ell=0}^{n-(j-t)} |C_\ell| \cdot b^\ell
\end{aligned}$$

Thus,

$$\mathbf{Pr}[G_i \in (b^{j-t-1}, b^{j-t}]] \leq \frac{b^{j-t+1}}{\sum_{\ell=0}^{n'} N_\ell \cdot 2^\ell} \cdot \sum_{\ell=0}^{n'} |C_\ell| \cdot b^\ell \tag{11}$$

The accepting coins, $ACC_x\left(\gamma_{i-1}\right)$, are partitioned between the clusters $C_0, \ldots, C_{n'}$. Moreover, the number of accepting coins in cluster C_ℓ is at least $b^\ell \cdot |C_\ell|$. Thus,

$$\sum_{\ell=0}^{n'} |C_\ell| \cdot b^\ell \leq |ACC_x(\gamma_{i-1})| \tag{12}$$

Passing Step (2) of the emulation protocol mandates that $\sum_{\ell=0}^{n'} N_\ell \cdot b^{\ell+1} \geq M_i$. Hence

$$\sum_{\ell=0}^{n'} N_\ell \cdot b^\ell = \frac{1}{b} \cdot \sum_{\ell=0}^{n'} N_\ell \cdot b^{\ell+1} \geq \frac{1}{b} \cdot M_i \tag{13}$$

Using Eq. (12) and (13) and recalling that $\frac{M_{i-1}}{ACC_x(\gamma_{i-1})} = g_{i-1} > b^{j-1}$, we can upper bound Eq. (11) and get

$$\begin{aligned}
\mathbf{Pr}[G_i \in (b^{j-t-1}, b^{j-t}]] &\leq \frac{b^{j-t+1} \cdot |ACC_x(\gamma_{i-1})|}{\frac{1}{b} \cdot M_i} \\
&= \frac{b^{j-t+2} \cdot |ACC_x(\gamma_{i-1})|}{M_i} \\
&= \frac{b^{j-t+2}}{g_{i-1}} \\
&\leq \frac{b^{j-t+2}}{b^{j-1}} \\
&= b^{-t+3}
\end{aligned}$$

which completes the proof. ■

3.2.2 Proof of Theorem 2

We shall show that the emulation protocol of Construction 6 (combined with the sampling protocol of Construction 7) yields a public-coin interactive proof system for any set having r rounds and a gap of at least B^r. Recall that when these two constructions are combined as detailed at the end of Section 3.1, the resulting public-coin protocol has $r + 1$ rounds. The completeness feature of this protocol is quite straightforward (but will be spelled out next). The soundness feature will be proven later, while relying on the main lemma.

Completeness. We claim that if x is a yes-instance, and the prover is honest, then the verifier accepts with probability greater than $\frac{2}{3}$. We first show that if the sampling goes well, namely the message sampled reside in the chosen cluster in all of the iterations, then the verifier accepts. We then show that the sampling goes well with probability greater than $\frac{2}{3}$.

We prove that if the sampling goes well then on every iteration i the verifier does not abort and $|ACC_x(\gamma_i)| \geq M_i$. We prove this by induction on the iteration index. By the induction hypotheses, we assume that the verifier does not abort up to iteration i of the emulation. For iteration $i + 1$, when the prover sets $N_\ell = |C_\ell|$ as directed by the emulation protocol, the verifier doesn't abort in the Step (2) since the prover is honest and

$$\sum_{\ell=0}^{n'} N_\ell \cdot b^{\ell+1} = \sum_{\ell=0}^{n'} |C_\ell| \cdot b^{\ell+1} > |ACC_x(\gamma_i)| \geq M_i$$

Now, assume the verifier chooses cluster C_k. When a message α_{i+1} from the chosen cluster C_k is sampled, the prover supplies its response β_{i+1} to the message α_{i+1} so that $|ACC_x(\gamma_i, \alpha_{i+1}, \beta_{i+1})| \geq$

$b^k = M_{i+1}$. In particular, after the last iteration, γ_r consists of a full transcript that is consistent with verifier's coins ρ and $|ACC_x(\gamma_r)| = M_r = 1$, so the verifier accepts.

It is left to show that, with probability greater than $\frac{2}{3}$, the sampled messages reside in the chosen cluster in all of the iterations. Recall that we run the sampling protocol with completeness parameter $\frac{1}{3r}$. Since the prover and the verifier follow the sampling protocol, by the properties of the sampling protocol, on each iteration the sampled message resides in the chosen cluster with probability at least $1 - \frac{1}{3r}$. Therefore, with probability greater than $\frac{2}{3}$, elements from the chosen clusters are sampled in all the iterations.

Soundness. We show that if x is a no-instance, then for any prover strategy the verifier accepts with probability at most $\frac{1}{3}$. If the verifier accepts after a complete transcript γ_r is sampled, then $M_r = |ACC_x(\gamma_r)| = 1$ must hold; namely, there is one sequence of coin tosses consist with the interaction, and this is what the prover claims on the last round. In this case, the "gap" after the last round is 1 (i.e. $g_r = 1$). Therefore, in order to upper bound the probability the verifier accepts, it suffices to upper bound the probability that the gap after the last round, g_r, is smaller than or equal to 1. As before, we denote by G_i for $i \in \{0, \ldots, r\}$ the random variable that represent the gap after the i^{th} iteration. We set $G_0 \stackrel{\text{def}}{=} g_0$, where g_0 is the initial gap between the number of accepting coins for yes-instances and no-instances. Hence, it is enough to show that if $g_0 = B^r$, then $\mathbf{Pr}[G_r \leq 1] < \frac{1}{3}$, where B is a constant that will be determined later.

We define random variables $D_1, \ldots, D_r$ representing the decrease in the gap between two consecutive rounds

$$D_i = \begin{cases} \frac{G_{i-1}}{G_i} & \text{if } G_i < \infty \\ 0 & \text{otherwise} \end{cases}$$

Conditioning on $G_{i-1} \in (b^{j-1}, b^j]$, we know that if $D_i \in (b^{t-1}, b^t]$ then

$$\frac{b^{j-1}}{b^t} < G_i = \frac{G_{i-1}}{D_i} \leq \frac{b^j}{b^{t-1}}$$

or equivalently $G_i \in (b^{j-t-1}, b^{j-t+1}]$. Hence, the main lemma asserts that for $i \in \{1, \ldots, r\}$ and $t < j$, we have

$$\begin{aligned} \mathbf{Pr}[D_i \in (b^{t-1}, b^t] \,|\, G_{i-1} \in (b^{j-1}, b^j]] &\leq \mathbf{Pr}[G_i \in (b^{j-t-1}, b^{j-t+1}] \,|\, G_{i-1} \in (b^{j-1}, b^j]] \\ &= \mathbf{Pr}[G_i \in (b^{j-t-1}, b^{j-t}] \,|\, G_{i-1} \in (b^{j-1}, b^j]] \\ &\quad + \mathbf{Pr}[G_i \in (b^{j-(t-1)-1}, b^{j-(t-1)}] \,|\, G_{i-1} \in (b^{j-1}, b^j]] \\ &\leq b^{-t+3} + b^{-(t-1)+3} \\ &= (1+b)b^{-t+3} \end{aligned}$$

By the definition of D_i if $G_i = \infty$ then $D_i = 0$, and in particular $D_i < b^{t-1}$. Thus

$$\mathbf{Pr}[D_i \in (b^{t-1}, b^t] \,|\, G_i = \infty] = 0$$

Hence, we can omit the conditioning on G_{i-1}, since we bounded the probability conditioning on every value of G_{i-1} by a term which is independent of the condition. We get

$$\mathbf{Pr}\left[D_i \in (b^{t-1}, b^t]\right] \leq (1+b) \cdot b^{-t+3}$$

or equivalently

$$\mathbf{Pr}\left[\log_b(D_i) \in (t-1,t]\right] \leq (1+b)\cdot b^{-t+3} \tag{14}$$

If for every iteration $i \in \{1,\ldots,r\}$ it holds that $G_i < \infty$, then by the definition of D_i we have

$$G_r = \frac{G_{r-1}}{D_r} = \ldots = \frac{G_0}{D_1 \cdot \ldots \cdot D_r}$$

Otherwise, there exists an iteration i for which $G_i = \infty$. In such a case, by the definition of g_i it follows that $|ACC_x(\gamma_i)| = 0$ and hence the number of accepting coins for every transcript that γ_i is a prefix of is also zero. In particular $|ACC_x(\gamma_r)| = 0$ and hence then $G_r = \infty$. On the other hand when $G_i = \infty$ we have $D_i = 0$. Hence if we interpret $\frac{1}{0}$ as ∞ we have that

$$G_r = \frac{G_0}{D_1 \cdot \ldots \cdot D_r} = \infty$$

Hence,

$$\begin{aligned}
\mathbf{Pr}[G_r \leq 1] &= \mathbf{Pr}\left[\frac{G_0}{D_1 \cdot \ldots \cdot D_r} \leq 1\right] \\
&= \mathbf{Pr}[D_1 \cdot \ldots \cdot D_r \geq B^r] \\
&= \mathbf{Pr}[\log_b[D_1 \cdot \ldots \cdot D_r] \geq r \cdot \log_b B]
\end{aligned}$$

and

$$\mathbf{Pr}[G_r \leq 1] = \mathbf{Pr}\left[\sum_{i=1}^{r} \log_b(D_i) \geq r \log_b B\right] \tag{15}$$

We define random variables L_i for $i \in \{1,\ldots,r\}$

$$L_i = \begin{cases} \lceil \log_b(D_i) \rceil & \text{if } \log_b(D_i) \geq 0 \\ 0 & \text{if } \log_b(D_i) < 0 \end{cases}$$

where $\log_b 0$ is interpreted as $-\infty$. We can upper bound the expectation of L_i using Eq. (14)

$$\begin{aligned}
\mathbb{E}[L_i] &= \sum_{t=1}^{n'} \mathbf{Pr}[\lceil \log_b(D_i) \rceil = t] \cdot t \\
&= \sum_{t=1}^{n'} \mathbf{Pr}[\log_b(D_i) \in (t-1,t]] \cdot t \\
&\leq \sum_{t=1}^{n'} (1+b)\cdot b^{-t+3} \cdot t \\
&< (1+b)\cdot \frac{b^4}{(b-1)^2}
\end{aligned}$$

Setting B such that $\log_b B = 3(1+b)\cdot \frac{b^4}{(b-1)^2}$,

$$\sum_{i=1}^{r} \mathbb{E}[L_i] < \frac{r \cdot \log_b B}{3} \tag{16}$$

and using Markov inequality we get

$$\begin{aligned}\mathbf{Pr}\left[\sum_{i=1}^{r} L_i \geq r\log_b B\right] &\leq \frac{\mathbb{E}\left[\sum_{i=1}^{r} L_i\right]}{r\log_b B}\\ &\leq \frac{1}{3}\end{aligned}$$

Lastly, recall that the variable L_i upper bounds $\log_b(D_i)$, thus we can upper bound the value of Eq. (15) by using the L_i's

$$\mathbf{Pr}\left[G_r \leq 1\right] \leq \mathbf{Pr}\left[\sum_{i=1}^{r}\log_b\left(D_i\right) \geq r\log_b B\right] \leq \mathbf{Pr}\left[\sum_{i=1}^{r} L_i \geq r\log_b B\right] \leq \frac{1}{3} \tag{17}$$

which completes the proof of the theorem.

On the choice of the base parameter b. Recall that $B = b^{3\cdot(1+b)b^4/(b-1)^2}$, where B^r is the initial gap required by our emulation. Wishing to minimize B calls for minimizing $f(b) = \frac{(1+b)b^4\ln b}{(b-1)^2}$, and one can readily verify that the optimum value is in the interval $[1.01, 10]$, since $f(2) < 48$ whereas $f(b) > 100$ for both $b \in (1, 1.01]$ and $b > 10$. The optimun value is $b \approx 1.32821$, yet $f(2) < 2 \cdot f(1.32821)$.

3.3 Lower bounds

We first observe that for any base parameter $b > 1$, the gap may be reduced by a factor of b in each iteration (of the emulation protocol) due to the here fact that each element in each C_j is counted as if it has a weight of b^{j+1} whereas its actual weight may be merely b^j. Thus, if b is a constant, then Theorem 3 follows (with $C = b$). So we should deal with the case of $b = 1 + o(1)$, or, equivalently, establish a bound that is independent of b. Hence, we may assume that $b \in (1, 2]$.

The key observation is that the prover can easily reduce the gap when neighboring clusters have similar weight. That is, suppose that $|C_j| \cdot b^j = |C_{j+1}| \cdot b^{j+1}$ (and that all messages in C_k have weight exactly b^k). Further suppose that the prover claims that $N_{j+t} = |C_j|$ and $N_{j+t+1} = |C_{j+1}|$, which supports a gap of b^t. Now, the verifier will select the index $j + t$ with probability half, but the prover can try to let it sample from a set that contains as many elements of C_{j+1} as possible (and use elements of C_j only to fill-up the rest). Indeed, the prover should provided $N_{t+j} = |C_j|$ elements, whereas $|C_{j+1}| = |C_j|/b$. Still, when the prover does so, the verifier selects an element of $|C_{j+1}|$ with probability (approximately) $1/b$, and when this happens the parties continue to the next iteration with a gap of $\frac{b^{t+j}}{b^{j+1}} = b^{t-1}$ rather than b^t. These considerations establish the fact that *with probability at least $1/2b$, the prover can decrease the gap by a factor of b.* In light of the first paragraph, this seems quite useless, but the point is that the argument can be extended to clusters that are a distance k apart. Specifically:

Claim 11 (unavoidable gap decrease) *For any $k \geq 1$, with probability $1/2b^k$, the prover can decrease the gap by a factor of b^k.*

Proof: We iterate the foregoing argument, but use $|C_j| \cdot b^j = |C_{j+k}| \cdot b^{j+k}$. Suppose that the prover claims that $N_{j+t} = |C_j|$ and $N_{j+t+k} = |C_{j+k}|$, which supports a gap of b^t. Now, the verifier

will select the index $j+t$ with probability half, and the prover can try to let it sample from a set that contains as many elements of C_{j+k} as possible. When the prover does so, the verifier selects an element of $|C_{j+k}|$ with probability (approximately) $1/b^k$, and when this happens the parties continue to the next iteration with a gap of $\frac{b^{t+j}}{b^{j+k}} = b^{t-k}$. ■

Recalling that $b \in (1,2]$, we just choose k such that $b^k \in [2,4]$, and apply Claim 11. It follows that, in each iteration, with probability $1/8$, the prover can decrease the gap by a factor of 2. Theorem 3 follows with $C = 2^{1/9}$, since (for sufficiently large r) with high probability the prover will be successful in at least $r/9$ of the iterations.

Bibliography

[**Bab85**] L. Babai. Trading Group Theory for Randomness. In *17th STOC*, pages 421–429, 1985.

[**BM88**] L. Babai and Shlomo Moran. Arthur-Merlin games: a randomized proof system, and a hierarchy of complexity clusters. *Journal of Computer and System Sciences*, 36.2 (1988): 254-276.

[**BGGHKMR**] Ben-Or, M., Goldreich, O., Goldwasser, S., Hastad, J., Kilian, J., Micali, S., & Rogaway, P. (1990, January). Everything provable is provable in zero-knowledge. In *Advances in Cryptology: Crypto'88*, (pp. 37-56). Springer New York.

[**FGMSZ**] M. Fürer, O. Goldreich, Y. Mansour, M. Sipser, and S. Zachos. On Completeness and Soundness in Interactive Proof Systems. In *Advances in Computing Research: a research annual*, Vol. 5 (Randomness and Computation, S. Micali, ed.), pages 429–442, 1989.

[**Gol08**] O. Goldreich. *Computational Complexity: A Conceptual Perspective*. Cambridge University Press, 2008.

[**GVW02**] Goldreich, Oded, Salil Vadhan, and Avi Wigderson. On interactive proofs with a laconic prover. *Computational Complexity*, 11, no. 1-2 (2002): 1-53.

[**GMR85**] Goldwasser, Shafi, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof-systems. In the proceedings of the 17th annual ACM symposium on Theory of computing. ACM, 1985.

[**GS86**] Goldwasser, Shafi, and Michael Sipser. Private coins versus public coins in interactive proof systems. In the proceedings of the 18th annual ACM symposium on Theory of computing. ACM, 1986.

[**LFKN92**] Lund, Carsten, Lance Fortnow, Howard Karloff, and Noam Nisan. Algebraic methods for interactive proof systems. *Journal of the ACM*, 39, no. 4 (1992): 859-868.

[**Sha92**] Shamir, Adi. IP= PSPACE. *Journal of the ACM*, 39, no. 4 (1992): 869-877.