On Prover-Efficient Public-Coin Emulation of Interactive Proofs

Gal Arnon ✉
Weizmann Institute of Science, Rehovot, Israel

Guy N. Rothblum ✉
Weizmann Institute of Science, Rehovot, Israel

Abstract

A central question in the study of interactive proofs is the relationship between private-coin proofs, where the verifier is allowed to hide its randomness from the prover, and public-coin proofs, where the verifier's random coins are sent to the prover. The seminal work of Goldwasser and Sipser [STOC 1986] showed how to transform private-coin proofs into public-coin ones. However, their transformation incurs a super-polynomial blowup in the running time of the honest prover.

In this work, we study transformations from private-coin proofs to public-coin proofs that preserve (up to polynomial factors) the running time of the prover. We re-consider this question in light of the emergence of doubly-efficient interactive proofs, where the honest prover is required to run in polynomial time and the verifier should run in near-linear time. Can every private-coin doubly-efficient interactive proof be transformed into a public-coin doubly-efficient proof? Adapting a result of Vadhan [STOC 2000], we show that, assuming one-way functions exist, there is no general-purpose black-box private-coin to public-coin transformation for doubly-efficient interactive proofs.

Our main result is a loose converse: if (auxiliary-input infinitely-often) one-way functions do *not* exist, then there exists a general-purpose efficiency-preserving transformation. To prove this result, we show a general condition that suffices for transforming a doubly-efficient private coin protocol: every such protocol induces an efficiently computable function, such that if this function is efficiently invertible (in the sense of one-way functions), then the proof can be efficiently transformed into a public-coin proof system with a polynomial-time honest prover.

This result motivates a study of other general conditions that allow for efficiency-preserving private to public coin transformations. We identify an additional (incomparable) condition to that used in our main result. This condition allows for transforming any private coin interactive proof where (roughly) it is possible to efficiently approximate the number of verifier coins consistent with a partial transcript. This allows for transforming any *constant-round* interactive proof that has this property (even if it is not doubly-efficient). We demonstrate the applicability of this final result by using it to transform a private-coin protocol of Rothblum, Vadhan and Wigderson [STOC 2013], obtaining a doubly-efficient public-coin protocol for verifying that a given graph is close to bipartite in a setting for which such a protocol was not previously known.

2012 ACM Subject Classification Theory of computation → Computational complexity and cryptography; Theory of computation → Interactive proof systems

Keywords and phrases Interactive Proofs, Computational complexity, Cryptography

Digital Object Identifier 10.4230/LIPIcs.ITC.2021.3

Related Version *Full Version*: `https://eccc.weizmann.ac.il/report/2019/176/`

Funding This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No. 819702).

Acknowledgements We would like to thank Ron Rothblum for making us aware of Kilian's ideas which in turn developed into our piecemeal emulation protocol. We would also like to thank Zvika Brakerski and Moni Naor for helpful comments on the presentation of this work.

1 Introduction

Interactive proofs (IPs), introduced by Goldwasser, Micali and Rackoff [11] in 1985 are an important object in the study of complexity and cryptography. An interactive proof is an interactive protocol between two parties, a "prover" and a "verifier", where the prover is trying to convince the verifier of the membership of a string in a language. If this claim is true, then the verifier should be convinced with high probability. Otherwise, if the claim is false, then no matter what the prover does, the verifier should reject the claim with high probability. Since their inception, a central question in the study of interactive proofs has been the connection between private-coin proofs, where the verifier is allowed to hide its randomness from the prover, and public-coin proofs, where hiding information is not allowed. Public-coin protocols are especially appealing since they are easier to analyze and manipulate [4, 1, 2, 3]. Goldwasser and Sipser [12] showed that any private-coin interactive proof can be transformed into a public-coin proof while preserving the number of rounds (up to an additive constant).

One issue with this transformation is that of the honest prover's running time. Vadhan [22] showed that (assuming the existence of one-way functions) there exist protocols that cannot be transformed to be public-coin in a black-box manner while preserving the running time of the prover (up to polynomial factors). While in the classical setting the running time of the prover is considered unbounded, the recent line of works on *doubly-efficient* interactive proofs (deIPs) [10] restricts the honest prover to run in polynomial time. We emphasize that soundness is required to hold against computationally unbounded adversaries. Doubly-efficient interactive proofs apply only to tractable computations, and are therefore of interest when the verifier time can be smaller than the time required to decide the language without the help of a prover. Indeed, the main focus in the literature is on verifiers that run in near-linear time. Goldreich [5] gives a survey on recent work on doubly-efficient interactive proofs.

1.1 This Work

In this work we ask whether transformations of proofs from using private coins to using public coins are applicable to the doubly-efficient setting:

Which private-coin doubly-efficient interactive proofs can be transformed into public-coin doubly-efficient proofs and how can this be done?

We tackle the above question from a number of angles. Some of our results also apply to proofs that are not doubly-efficient.

We extend Vadhan's impossibility result to show that the existence of one-way functions implies that there are no transformations from private-coin deIPs to public-coin deIPs that work in a natural "black-box" way. Note that since deIPs exist only for problems in $\mathcal{BPP}$, one can *always* transform such proofs to using "public-coins" by having the verifier solve the problem on its own. This transformation is not black-box, but it is also not interesting, as the motivation for deIPs is to reduce the verifier's running time to under what is required for it to solve the problem on its own.

Our main result shows that this reliance on one-way functions is essentially tight. Namely, if one-way functions (of a certain type) do not exist, then (essentially) every doubly-efficient proof can be efficiently transformed:

▶ **Theorem 1** (Efficient emulation of deIPs in Pessiland (Informal)). *Suppose that infinitely-often auxiliary-input one-way functions do not exist. Then every language that has a doubly-efficient private-coin interactive proof with "good enough" soundness has a doubly-efficient public-coin interactive proof with the same number of rounds (up to a constant).*

▶ Remark 2. Theorem 1 mentions "good enough" soundness. This is due to the fact that there is a strong degradation in soundness when applying our technique.[1] One could be tempted to amplify the soundness using parallel or sequential repetition, but in the setting of deIPs, the overhead of repeating the protocol in terms of the verifier might be problematic (e.g. it might degrade the verifier's running-time from linear to quadratic).

Viewing this result through the prism of Impagliazzo's worlds [15], it (very) roughly says that in Pessiland (a world where one-way functions do not exist), efficiency-preserving transformation is always possible. We prove Theorem 1 by showing that for every deIP there exists a specific efficiently computable function such that if it is efficiently invertible in the sense of one-way functions[2], then efficiency-preserving transformation is possible (see Section 2.1 for a discussion on the notion of invertibility). We remark that a straightforward implementation of the Goldwasser-Sipser transformation requires exponential running time from the prover, and even an oracle that inverts *any* given function (on random inputs) does not seem sufficient for making their public-coin prover efficient. Indeed, our results require changing the transformation so that the ability to invert becomes sufficient for constructing a public-coin prover.

Using the technique for proving Theorem 1 (and some additional technical work) we show that in Pessiland's "one-way function"-less landscape, standard constant-round proofs (i.e. ones where the honest prover is allowed to run in super-polynomial time) can also be transformed to be public-coin with only a polynomial overhead on the honest prover's running time:

▶ **Theorem 3** (Efficient emulation of constant-round IPs in Pessiland (Informal)). *Suppose that infinitely-often auxiliary-input one-way functions do not exist. Then every language that has a constant-round private-coin interactive proof has a constant-round public-coin interactive proof where the honest prover's running time is polynomially related to that of the private-coin prover.*

1.1.1 Sufficient conditions for efficient transformation

Both the impossibility result of [22] and our extension to doubly-efficient proofs are proved by demonstrating a specific (arguably contrived) protocol that is hard to transform. It is very natural, then, to ask: considering interactive proofs on a case-by-case basis, for which protocols (or families of protocols) *is* efficiency-preserving transformation possible? In other words we wish to identify sufficient conditions that allow for efficiency-preserving transformation of private-coin proofs to public-coin ones.

In particular, Theorem 1 implies one such condition: Every deIP has an efficiently computable function such that if this function is efficiently invertible in the sense of one-way functions, then efficient transformation for this proof system is possible.

We identify an additional, rather natural, sufficient condition for efficient transformation. We show that if it is possible to efficiently count the number of coins that are consistent with transcripts of the protocol, then it is also possible to efficiently emulate the protocol using public-coins. Unlike in Theorems 1 and 3, this result does not preserve the number of rounds, but it applies to general interactive proofs (even when the protocol has an inefficient honest prover and a polynomial number of rounds).

[1] Specifically, in order for the public-coin protocol that we end up with to have constant soundness error, the soundness error of the original (private-coin) protocol should be $O(\text{poly}(n, r, \ell)^{-r})$ where n is the input length, and r and ℓ are the number of rounds and number of random bits used by the verifier in the original private-coin protocol respectively.

[2] The requirement that a specific function is *distributionally* invertible [16] also suffices.

▶ **Theorem 4** (Efficient emulation using approximation (Informal)). *Let $\mathcal{L}$ be a language and suppose that $\mathcal{L}$ has an r-round private-coin interactive proof with communication complexity m, and suppose that for every incomplete transcript it is possible to efficiently approximate the number of verifier random coins that are consistent with the transcript. Then $\mathcal{L}$ has a $2rm$-round public-coin interactive proof with an efficient prover.*

A string of random coins ρ is consistent with an incomplete transcript of an execution of a protocol if for every verifier message α in the transcript, the verifier outputs α when given the transcript prefix leading up to α and using ρ as its random coins. We prove this theorem using a "piecemeal" emulation protocol in which the prover and the verifier together generate a string that is distributed according to the distribution of a random transcript in the private-coin protocol. The soundness error of the resulting protocol is a function of how good the approximation algorithm is. In particular, if one can *exactly* count the number of verifier random coins that are consistent with a transcript efficiently, then soundness is perfectly preserved.

Theorem 4 gives us a condition for efficiently transforming proofs from private-coin to public-coin that is incomparable to the condition implied by Theorems 1 and 3. The condition implied by from Theorems 1 and 3 is efficient distributional inversion for some (efficiently computable) function that depends on the protocol, whereas Theorem 4 uses efficient approximation of the number of verifier coins consistent with a transcript. We demonstrate a natural protocol for which the efficient counting condition of Theorem 4 is satisfied, whereas we don't know how to efficiently invert the function implied by Theorems 1 and 3.

1.1.2 An application

Rothblum, Vandhan and Wigderson [18] show a private-coin proof of proximity for distinguishing between a graph that is bipartite and graphs that are both far from bipartite and well-mixing. Roughly speaking, an interactive proof of proximity (IPP) is an interactive proof where the verifier has sub-linear query access to the input. The problem of distinguishing between a bipartite graph and a well-mixing graph that is far from bipartite has also been studied extensively in the past in the context of property testing [8, 9]. By applying Theorem 4 to the private-coin protocol of [18] we show a new doubly-efficient public-coin proof system for this problem. We describe this below in more detail.

▶ **Theorem 5** (Public-coin IPP for bipartiteness (Informal)). *For every $\varepsilon > 0$, there exists a* public-coin *interactive proof of ε-proximity with an* efficient *prover for the problem of distinguishing between bipartite graphs and graphs that are ε-far from bipartite and are well-mixing.*

We remark that no such proof was previously known (except for ε close to 1).

The main property of the RVW bipartiteness protocol that we use, is the fact that it that the (private coin) verifier uses only a logarithmic number of coins (it has logarithmic *randomness complexity*), and this suffices for soundness error $1-\Omega(\varepsilon)$. Thus, polynomial time suffices for enumerating all possible choices of verifier randomness and for exactly counting how many choices are consistent with the transcript. The transformation of Theorem 4 is soundness preserving when efficient exact counting is possible, and so gives an efficient public-coin protocol with soundness error $1-\Omega(\varepsilon)$, which can be amplified to obtain constant soundness. See the full paper for further discussion and details.

We note that a similar argument applies to every proof system where the verifier's randomness complexity is $O(\log n)$. We show that *every* such private-coin IP (resp. IPP) can be transformed to a public-coin IP (resp. IPP) while the honest prover's running time remains polynomial.

We further note that the Goldwasser-Sipser (GS) approach to transforming private-coin proofs into public-coin ones, and the result behind Theorem 1, can also preserve the prover's efficiency when the verifier's randomness complexity is $O(\log n)$. However, the GS approach degrades soundness significantly, and hence usually requires parallel repetition before applying the transformation. Here, since the starting (private coin) protocol has large soundness error, repeating it to reduce the soundness error to the point where the GS approach can be applied requires super-logarithmic randomness complexity, which means that the GS transformation will not preserve the prover's efficiency. In comparison, in this setting Theorem 4 preserves soundness (see above), and so it can be used even when the soundness error of the private-coin protocol is large.

1.2 Related Work

A number of works have tackled the question of private versus public coins, including Haitner, Mahmoody and Xiao [13] who showed that if the prover is given an $\mathcal{NP}$ oracle it is possible to transform private-coin protocols into public-coin ones where both the prover and the verifier run in polynomial time. Holenstein and Künzler [14] show a public-coin protocol in which the prover helps the verifier sample from a distribution, where in addition to the sampled element the verifier ends up with an approximation of the probability that the element is sampled from the distribution. They then show this can be used for public-coin emulation. Goldreich and Leshkowitz [6] improved upon the soundness requirement of Goldwasser and Sipser. In all of the results described the prover is inefficient and the running time of the verifier incurs a polynomial overhead. We additionally note that the celebrated $\mathcal{IP} = \mathcal{PSPACE}$ Theorem [17, 19], implies a non-black-box transformation of private-coin protocols to public-coin ones. The protocol used to show that $\mathcal{PSPACE} \subseteq \mathcal{IP}$ is public-coin, and so one can use this result to transform private-coin protocols into public-coin ones as follows: Given an interactive proof, use the transformation for $\mathcal{IP} \subseteq \mathcal{PSPACE}$ to convert it into a $\mathcal{PSPACE}$ problem. Then use the reduction and protocol showing that $\mathcal{PSPACE} \subseteq \mathcal{IP}$ to construct a public-coin proof. While this transformation is not black-box, it blows up the complexity of the honest prover and the number of rounds of the protocol.

2 Technical Overview

2.1 Overview of the Round-Efficient Emulation

Goldwasser and Sipser [12] showed not only that it is possible to transform private-coin proofs into public-coin ones, but also that this can be done without significantly increasing the number of rounds in the protocol. We show that, given a distributional inverter for certain functions, the prover can be made efficient. A distributional inverter for a function f is an efficient randomized algorithm that upon receiving an input y drawn from the distribution $f(U_n)$ returns a random element from the set $f^{-1}(y)$.

▶ Remark 6. Proving Theorems 1 and 3 would be significantly simpler if we were to consider a stronger form of inversion, where the input y to the inverter can be any element in the support of f. That is, y need not be given as a sample from the distribution $f(U_n)$. We consider the weaker and more complicated variant, since it allows us to establish Theorem 1, and through it the tight relationship between one-way functions and the (non-)existence of private-coin to public-coin transformations that preserve the prover's running time.

We give three toy cases for protocols of increasing complexity, and then discuss the general case. For each case we show how Goldwasser and Sipser's original protocol can be applied in order to transform it to public-coin, and then discuss how we use distributional inversion in order to make the prover efficient. Eventually, this requires changes to the Goldwasser-Sipser transformation. In all three toy cases consider a language $\mathcal{L}$ and a one round private-coin protocol denoted $\langle P, V\rangle$ with perfect completeness and soundness error s. Since it has one round, the protocol is of the following form: On input x the verifier begins with choosing a random $\rho \leftarrow \{0,1\}^\ell$, then sends $\alpha = V(x,\rho)$ where $\alpha \in \{0,1\}^m$. After receiving β from the prover, the verifier accepts if $V(x,\alpha,\beta;\rho) = 1$. Henceforth throughout this overview we omit the shared input x from notation of the verifier and prover functions.

2.1.1 Case 1: Equally Likely Messages With *Known* Number of Messages

2.1.1.1 The Protocol

In addition to the protocol being one-round and having perfect completeness, we assume the following properties:

- Equally Likely Messages: If x is in the language $\mathcal{L}$, then every pair of messages $\alpha_1, \alpha_2 \in \{0,1\}^m$ that have non-zero probability of being sent by the original verifier V are equally likely: $\Pr_{\rho\leftarrow U_\ell}[V(\rho) = \alpha_1] = \Pr_{\rho\leftarrow U_\ell}[V(\rho) = \alpha_2]$.
- Known Number of Messages For Completeness: If $x \in \mathcal{L}$ then there is a known efficiently computable function $N : \{0,1\}^* \to \mathbb{N}$ such that the total number of messages sent by the verifier with non-zero probability is $N(x)$. That is,

$$N(x) = \left|\left\{\alpha | \exists \rho \in \{0,1\}^\ell \text{ s.t. } V(x;\rho) = \alpha\right\}\right|$$

- Few Messages For Soundness: If $x \notin \mathcal{L}$ then there are significantly fewer than $N(x)$ verifier messages.

As a running example for this case one is encouraged to think of the classical private-coin protocol for the graph non-isomorphism problem [7]. In this language the input is comprised of two n-vertex graphs G_0 and G_1 which are claimed to be non-isomorphic. The protocol is as follows: the verifier chooses a random bit b, and random permutation π. It sends $\tilde{G} = \pi(G_b)$ to the prover who must return some b'. The verifier accepts if $b' = b$. One can easily verify that this protocol has completeness 1 and soundness error $\frac{1}{2}$. Moreover, assuming for simplicity that the graphs have no automorphisms, every verifier message is equally likely and if the graphs are non-isomorphic the number of possible verifier messages is $N = 2n!$. If the graphs are isomorphic then there are only $n!$ different messages.

2.1.1.2 The Goldwasser-Sipser Transformation

The transformation of $\langle P, V\rangle$ into a public-coin protocol hinges on the observation that, in this toy case, in order distinguish whether x is in the language, the prover need only show that the number of possible verifier messages is at least N (since by assumption if $x \notin \mathcal{L}$ there are significantly fewer such messages). Thus a (public-coin) "set lower-bound" protocol is used, showing that the set of all valid verifier messages (ones that are sent by V with non-zero probability over the choice of its random coins) is large . Letting $\mathcal{H}_{m,k}$ be a family of pairwise independent hash functions from $\{0,1\}^m$ to $\{0,1\}^k$, U_k be the uniform distribution over k bits and $k = k(N)$ be a value to be discussed later, the final protocol is as follows:

1. The parties execute a "set lower-bound" protocol proving that the number of verifier messages is at least N:
   a. The verifier chooses a random $h \leftarrow \mathcal{H}_{m,k}$ and $y \leftarrow U_k$ [3]
   b. The prover returns $\alpha \in \{0,1\}^m$.
   c. The verifier tests that $h(\alpha) = y$ and otherwise rejects.
2. The prover sends $\rho \in \{0,1\}^\ell$.[4]
3. The verifier accepts if $V(\rho) = \alpha$.

In the case of completeness, where there are N verifier messages, if k is "small enough" (i.e. the hash function is very compressing) it is likely that there exists some valid message α that hashes to y. Conversely, in the case of soundness, where there are significantly fewer than N legal verifier messages, if k is "large enough" then it is unlikely that there will exist a valid message that hashes to y. Thus, completeness and soundness of the protocol are governed by setting k to a reasonable value, which depends on the gap between N and and the magnitude of the prover's lie in the case of a false claim. Note that in this protocol the prover did not even need to send its message β - it was sufficient to use the fact that there is a large gap in the number of verifier messages between the cases of completeness and soundness. The sub-protocol executed in Step 1 is known as the "set lower-bound" protocol, and can be generalized to show a lower-bound on the size of any set S for which the verifier can efficiently test membership. Specifically, the protocol begins with a claim that $N \leq |S|$ and ends with both parties holding an element x for which the verifier needs to verify that it belongs to S. If $N \leq |S|$, then $x \in S$ with high probability, and if $|S| \ll N$, $x \notin S$ with high probability regardless of the prover strategy. A protocol inspired by the set lower-bound protocol is presented and analysed in the full paper under the name "prover-efficient sampling protocol". Going back to the example of graph non-isomorphism, upon receiving h, y from the verifier, the prover would send some graph $\tilde{G}$, a bit b and a permutation π. The verifier would then accept if $h(\tilde{G}) = y$ and $\tilde{G} = \pi(G_b)$.

2.1.1.3 Prover Efficiency

The prover strategy in the above protocol is inefficient. It receives some h, y and is required to find a legal message α that hashes to y and some choice of randomness ρ that leads to α. However, the prover can be made efficient by giving it oracle access to an inverter for the function $f(h, \rho) = h, h(V(\rho))$. An inverter for a function $f : \{0,1\}^a \to \{0,1\}^b$ is a randomized algorithm that on input y drawn from the distribution $f(U_a)$ returns some element x in the set of preimages of y under f (that is, $x \in f^{-1}(y)$). We stress that the input to the inverter must come from the correct distribution, which in our case is $f(\mathcal{H}_{m,k}, U_\ell) \equiv (\mathcal{H}_{m,k}, \mathcal{H}_{m,k}(V(U_\ell))$. The hash function h is clearly chosen by the verifier from the correct distribution. The image y is drawn by the verifier from the uniform distribution which, if the hash function is compressing enough, will be statistically close to $h(V(U_\ell))$ by the Leftover Hash Lemma. Preimages of (h, y) with respect to f are of the form (h, ρ) where $h(V(\rho)) = y$. The prover can send ρ and use ρ to calculate α. In all of this the prover only needs to make a single oracle call, and to calculate $\alpha = V(\rho)$, and is therefore efficient.

[3] In the classic transformation it suffices to set $y = 0^k$ and is described here thus as it will be required later by our transformation.
[4] The final prover message can be merged with the previous one to save a round and is described here as a separate round for clarity.

2.1.2 Case 2: Equally Likely Messages With *Unknown* Number of Messages

2.1.2.1 The Protocol

We make the same assumptions on the protocol as in Case 1, except that the verifier in the transformation does not know N, the number of verifier messages.

2.1.2.2 The Goldwasser-Sipser Transformation

The protocol is based on two observations made in the case of a cheating prover:

- Since the soundness error is s and the verifier uses ℓ random coins, the number of verifier messages α for which there exist β and ρ such that $V(\alpha, \beta; \rho)$ accepts is at most $s \cdot 2^\ell$.
- For every fixed α and β, the number of coins ρ such that $V(\rho) = \alpha$ and $V(\alpha, \beta; \rho)$ accepts is also bounded from above by $s \cdot 2^\ell$.

If either of the above were not true, it would mean the soundness error is greater than s. Now notice that since all messages are equally likely, if there are N valid verifier messages, then each message has $\frac{2^\ell}{N}$ different coins that are consistent with it. Importantly, if N is small, $\frac{2^\ell}{N}$ is large. This gives rise to the following protocol:

1. Prover sends N, a claim on the number of verifier messages.
2. Prover and verifier execute the set lower-bound protocol to show that the number of legal verifier messages is at least N. The parties end up with some α claimed to be a verifier message.
3. Prover sends some β.
4. The parties execute the set lower-bound protocol to show that the number of coins that are consistent with α and lead the verifier to accept (α, β) is at least $\frac{2^\ell}{N}$. The parties end up with a ρ which is supposed to in the set of random coins that lead the verifier to α.
5. Verifier accepts if $V(\rho) = \alpha$ and $V(\alpha, \beta; \rho) = 1$.

The protocol is complete, since if the prover is honest it is likely to succeed in both the set lower-bound protocols, meaning it samples both a valid message α and valid coins ρ such that $V(\rho) = \alpha$ and $V(\alpha, \beta; \rho) = 1$. We now turn towards soundness. Let S be the set of verifier messages for which the prover has an accepting strategy (messages α for which there exist β and ρ such that $V(\alpha, \beta; \rho)$ accepts). For verifier message α and prover message β, let $T_{\alpha,\beta}$ be the number of random coins ρ such that $V(\alpha, \beta; \rho)$ accepts. Recall from the argument above, that $|S| \leq s \cdot 2^\ell$ and for any α, β, $|T_{\alpha,\beta}| \leq s \cdot 2^\ell$. Now note that if the verifier ends up with a verifier message $\alpha \notin S$ it has a message for which no fixing of β and ρ will make the verifier accept. Thus the prover must try to sample in S. Similarly, after fixing α and β if the verifier has $\rho \notin T_{\alpha,\beta}$ it will reject, and so the cheating prover must try to cause the verifier to end up with an element in $T_{\alpha,\beta}$. To see why the protocol is sound consider the prover's choice of N. If $|S| \leq s \cdot 2^\ell \ll N$, then due to the set lower-bound protocol the prover is unlikely to make the verifier sample from S and so the verifier will reject with high probability. If N is small, then $\frac{2^\ell}{N}$ is large. Fixing α and β, if $|T_{\alpha,\beta}| \leq s \cdot 2^\ell \ll \frac{2^\ell}{N}$, then the verifier is unlikely to end up with such coins, meaning it rejects. Note in the above analysis we have that $s \ll \min\{N, \frac{2^\ell}{N}\}$. Thus if s is small enough to begin with, the prover will be forced to lie and be caught with high probability.

2.1.2.3 Prover Efficiency

Inspecting the above protocol there are three things that the honest prover needs to do: Count N, the number of verifier messages, execute the prover's side of the set lower-bound protocol to show that there are at least N verifier messages, and execute the prover's side of

the set lower-bound protocol to prover that the number of coins that are consistent with α is at least $\frac{2^\ell}{N}$. We explain how to compute each of these efficiently in reverse order:

1. Set Lower-Bound Protocol for Number of Consistent Coins: In this set lower-bound protocol the parties have already computed a verifier message α. The prover receives some hash function h and an image y from the verifier, and must return some ρ such that $V(\rho) = \alpha$ and $h(\rho) = y$. Naively it seems that this can be solved simply if the prover has access to an inverter for the function $f(h, \rho) = h, V(\rho), h(\rho)$ since preimages of (h, α, y) are exactly of the form h', ρ such that $h', V(\rho), h'(\rho) = h, \alpha, y$. The problem with this idea is that to use this inverter it must be that the message α *be drawn from the distribution* $V(U_\ell)$. Thus to use this idea it is imperative that α come from the correct distribution.
2. Set Lower-Bound Protocol for Number of Messages: In order to complete this stage, the prover must find some valid verifier message α that hashes to y, and this (as we did in Case(1))can be done using an inverter for the function $f(h, \rho) = h, h(V(\rho))$. Unfortunately as mentioned in point (1), we need it α to be chosen from the real message distribution $V(U_\ell)$. Unfortunately using an inverter as described, the message α might be drawn from a distribution which is far from the real one.[5] This issue is fixed if we move from using a regular inversion oracle to a *distributional* inversion oracle. Roughly, a distributional inverter for a function $f : \{0,1\}^m \to \{0,1\}^t$ is a randomized algorithm A such if y is drawn from $f(U_m)$, the distribution $A(y)$ is statistically close to a random pre-image of y under f.
3. Computing N: We show that given an inversion oracle for the function $f(h, \rho) = h, h(V(\rho))$ it is possible to efficiently approximate N, the number of verifier messages. The way this is done is inspired by the techniques of [20] for approximate counting using an $\mathcal{NP}$ oracle. The prover chooses $h \leftarrow \mathcal{H}_{m,k}$ and $y \leftarrow \{0,1\}^k$ for increasingly larger values of k, and calls the inversion oracle on input (h, y). When k is small relative to the number of verifier messages, there will likely exist a message α that hashes to y, and thus the inverter will return a set of coins ρ such that $V(\rho) = \alpha$. Once k is set to a relatively large value this is unlikely and the inverter will fail. Thus, given the smallest size of k for which the inverter fails the prover can estimate the size of the set.

2.1.3 Case 3: A Two-Cluster Protocol

2.1.3.1 The Protocol

As in Case 2, we do not assume that the parties know initially the number of verifier messages. Moreover we replace the assumption that all messages are of equal likelihood with the following one:

- Two Clusters: The verifier messages that have non-zero probability can be partitioned into two "clusters" C_0 and C_1, where every message in C_b has equal likelihood p_b. Furthermore, each message in C_1 is significantly more likely than messages in C_0: $p_0 \ll p_1$. Both parties know the values p_0 and p_1 but not $|C_0|$ and $|C_1|$.

[5] Indeed, suppose that the inverter always returns the lexicographically smallest ρ such that $h(V(\rho)) = y$. This will cause a sampling bias towards those messages that have coins that are lexicographically smaller.

2.1.3.2 The Goldwasser-Sipser Transformation

Rather than giving a claim about the number of possible messages, the prover will claim only that the heaviest of the flat clusters is large (the weight of cluster C_b is $p_b \cdot |C_b|$).

1. Prover calculates $|C_0|$ and $|C_1|$ and chooses a bit b such that $p_b \cdot |C_b| \geq p_{1-b} \cdot |C_{1-b}|$. It sends b and $N = |C_b|$.
2. Verifier tests that $p_b \cdot N \geq \frac{1}{2}$ and otherwise rejects.
3. Prover and verifier execute the set lower-bound protocol to show that the size of cluster C_b is at least N. The parties end up with some α claimed to be in cluster b.
4. Prover sends some β.
5. The parties execute the set lower-bound protocol to show that the number of coins that are consistent with α and lead the verifier to accept (α, β) is at least $p_b \cdot 2^\ell$. The parties end up with a ρ which is supposed to be a set of random accepting coins that lead the verifier to α.
6. Verifier accepts $V(\rho) = \alpha$ and $V(\alpha, \beta; \rho) = 1$.

Completeness can be verified by noting that since there are only two clusters, the heaviest cluster must hold at least half of the weight of the distribution and so the verifier's test that $p_b \cdot N \geq \frac{1}{2}$ will pass. Due to the fact that every message α in C_b has likelihood p_b, there are exactly $p_b \cdot 2^\ell$ different coins that would lead the verifier to output α and to accept. For soundness first fix b. Once b is fixed, the smallest the prover can set N to be is $\frac{1}{2p_b}$ because otherwise the verifier will reject in Step 2. If p_b is very small, this value is large. As in the analysis of Case 2, since the total number of verifier messages for which the prover as an accepting strategy is small, if the claim N is large, it will likely not manage to make the verifier accept. If p_b is large, then the value of N can be set to be small, but in this case the value $p_b \cdot 2^\ell$ in Step 5 is large. Noting that for any α and β the number of coins ρ for which $V(\alpha, \beta; \rho)$ accepts is at most $s \cdot 2^\ell$, which we think of as very small, the prover is unlikely to be able to cause the verifier to end up with coins that will make it accept.

2.1.3.3 Prover Efficiency

In the classic transformation as described above the prover must do three things: Calculate the size of each cluster, and take part in both executions of the set lower-bound protocol.

1. Counting $|C_0|$ and $|C_1|$: The approximate counting technique as used in Case 2 to approximate N can be used to approximate the number of coins which would lead to a message. Notice that for $\alpha \in C_b$ there are exactly $p_b \cdot 2^\ell$ coins that lead to α. Thus, by randomly choosing many messages and counting how many are in each cluster, the prover can build a "histogram" of the weights of each cluster - a list of each cluster and its respective weight. This is formally addressed in the full paper where it is shown that using both a sampling and a membership oracle one can build such a histogram. An issue with this approach is that the approximation procedure returns an *approximate* value for the number of coins that lead to a message. That is, for a message in C_b it may claim that the number of coins that lead to the message is anywhere in the range $(1 \pm \varepsilon) \cdot p_b \cdot 2^\ell$ for some (relatively small) ε. Since p_0 and p_1 are far from each other, the ranges $(1 \pm \varepsilon) \cdot p_0 \cdot 2^\ell$ and $(1 \pm \varepsilon) \cdot p_1 \cdot 2^\ell$ do not intersect. Thus it is still easy to recognize to which cluster a message belong.
2. Set Lower-Bound Protocol for Number of Messages in C_b: In this part of the protocol the prover receives a hash function h and image y and needs to return a message α such that $\alpha \in C_b$ and $h(\alpha) = y$. The prover cannot simply use an inverter for a function that samples inside of C_b since there may not be an efficient function for sampling in C_b. We

instead show that given a distributional inverter for $f(h, \rho) = h, h(V(\rho))$ it is possible to find preimages that are in C_b. This is true because the cluster has significant weight with respect to the distribution of messages, and so for a randomly chosen hash function and random image y the weight of elements that belong to the cluster and are preimages to y is unlikely to be very small relative to all other preimages of y.

3. Set Lower-Bound Protocol for Number of Consistent Coins: The prover's strategy can be made efficient in this protocol in exactly the same way as in the same set lower-bound in the previous case: by inverting $f(h, \rho) = h, V(\rho), h(\rho)$. Doing this has the same issue dealt with in Case 2 - the distribution from which the message α is drawn must be close to uniform. In the classical transformation the value of α will be far from random because the protocol always works with the heaviest cluster. Suppose, for example, that $p_0 \cdot |C_0| = p_1 \cdot |C_1| + \varepsilon$ for a very small ε. Then C_0 will always be chosen, even though in the the real message distribution the likelihood of being in each cluster is almost identical. In order to fix this skew in distribution we make the choice of b "smoother". Rather than setting b as the index of the heaviest cluster, we let b be random and sampled from the Bernoulli distribution where 0 is drawn with probability $\frac{p_0 \cdot |C_0|}{p_0 \cdot |C_0| + p_1 \cdot |C_1|}$. This presents a minor issue: now it could be that $p_b \cdot N < \frac{1}{2}$. To fix this, we limit the choice of clusters only to ones that have some noticeable probability of appearing, and so the verifier can make sure that the claimed probability is not smaller than this threshold. Similar smoothing techniques were used in [14] and [6] in different contexts.

2.1.4 Towards the General Case

In the general case, the verifier message distribution cannot be split into a small number of flat clusters and the protocol may have multiple rounds. To keep this overview simple we only consider doubly-efficient proofs. Making the transformation work for constant-round proofs with an inefficient prover requires some slight additional technical work.

2.1.4.1 General Message Distribution

General Message Distribution: The issue for working with a general distribution for the verifier messages is solved in the classical transformation by defining clusters of messages as follows: cluster i is the set of all the messages with weight in the range 2^{-i} and 2^{-i+1}. In our case, we have to work harder. Firstly, due to the way we use distributional inverters, we will need that for every cluster, the distribution of messages when restricted only to messages in the cluster be statistically close to uniform. This can be solved by splitting the distribution into more clusters - cluster i will now be all messages in the range $(1 + 1/\operatorname{poly}(n))^{-i}$ and $(1 + 1/\operatorname{poly}(n))^{-i+1}$. Note that the probabilities of messages in neighbouring clusters are similar. Therefore, the observation made in analysis of Case 3 that we can distinguish to which cluster a message belongs even though the approximation procedure does not return exact values for the number of coins that lead to a message, is false. To solve this issue, we work with "approximate clusters" - cluster i consists of all the verifier messages for which *the approximation procedure claims* the weight is between $(1 + 1/\operatorname{poly}(n))^{-i}$ and $(1 + 1/\operatorname{poly}(n))^{-i+1}$.

2.1.4.2 Imperfect Completeness

In order to accommodate protocols with completeness c, recall that the clusters are defined as sets of accepting coins with some weight. In the classical transformation in Case 3 the final prover's claim is that there are $p_b \cdot 2^{\ell}$ coins which would lead the verifier to accept. Since

now p_b is the probability that these coins are sampled *conditioned* on sampling accepting coins, the end claim is changed to $p_c \cdot c \cdot 2^\ell$. In our case, we will not be able to use inverters to find accepting coins, since we only know how to invert efficient functions, and we do not have an efficient function that returns a random accepting coin. We therefore redefine the clusters to refer to gerenal coins, not just accepting ones. This means that in our protocol, the verifier accepts with almost the same probability as in the original private-coin protocol.

2.1.4.3 Multiple Rounds

The issue of multiple rounds is solved in the original transformation by iteratively emulating each round of the protocol. In the following we ignore the issue of distributions over messages which are not uniform. This is treated as explained under "General Message Distribution". In the Goldwasser-Sipser protocol round i starts with the prefix of a transcript $\gamma_{i-1} = (\alpha_1, \beta_1, \ldots, \alpha_{i-1}, \beta_{i-1})$ and N_{i-1}, a claimed lower bound on the number of coins that are consistent with γ_{i-1}. The prover gives a claim N_i that there are N_i coins consistent with each message possible verifier message conditioned on the transcript prefix γ_{i-1}. The parties next run the set lower-bound protocol. That is, the verifier sends a randomly sampled hash function h and random y. The honest prover sends back α_i such that $h(\alpha_i) = y$ and that α_i is consistent with γ_{i-1} (i.e. there exist ρ such that $\alpha_1 = V(\alpha_1, \beta_1, \ldots, \alpha_{i-1}, \beta_{i-1}; \rho)$). The prover then sends β_i. This process is run iteratively until the parties have a full transcript γ along with a claimed lower-bound on the number of coins consistent with this full transcript, at which point the parties execute a final set lower-bound protocol to sample a set of coins ρ.

To follow Goldwasser and Sipser's formula with an efficient prover, we would like an efficient method such that given a random hash function h, and y find this method outputs a consistent α_i. In the one-round case as explained previously, we noted that $f(h, \rho) = h, h(V(\rho))$ was an efficiently computable function in order to sample $\alpha_1 = V(\rho)$. How can we use the same idea but correlate the output to a transcript? To more easily illustrate, in the following we consider a 2-round protocol so that our goal is to sample α_2 after the transcript (α_1, β_1) has already been set.

We show that if the proof in question is doubly-efficient, it suffices to invert the function f that on inputs h and ρ: Computes $\alpha_1 = V(\rho)$, $\beta_1 = P(\alpha_1)$, $\alpha_2 = V(\alpha_1, \beta_1; \rho)$ and outputs $(\alpha_1, \beta_1, h(\alpha_2))$. Firstly note that since the prover is efficient the function f can be computed in polynomial time. Next, notice that the distribution $f(\mathcal{H}, U_\ell)$ is identical to that of taking α_1, β_1 from a random execution of the protocol and additionally outputting a random hash of the next verifier message. Consider an inverter for f. Given a random pair α_1, β_1 and a random h, y it returns randomness ρ such that $\alpha_1 = V(\rho)$. Given ρ it is easy to compute $\alpha_2 = V(\alpha_1, \beta_1; \rho)$. Since ρ is consistent with α_1, β_1, we have that α_2 is also consistent with α_1, β_1. Moreover, α_2 will hash to y. This is exactly what we needed.

2.2 Overview of the Piecemeal Emulation Protocol

In this section we overview the techniques used to prove Theorem 4. We prove the theorem by constructing a protocol which we call the "piecemeal" emulation protocol, which is inspired by ideas described in [21] that are accredited to Joe Kilian.

The protocol hinges on a sampling protocol where the goal of the honest prover is to help the verifier generate a transcript that is distributed similarly to a random transcript of the protocol the parties are trying to emulate. Let $\mathcal{L}$ be a language and $\langle P, V \rangle$ be an r-round interactive proof for $\mathcal{L}$ with ℓ bits of randomness and message length m. Since there are r rounds, there are $2r$ messages sent in the protocol. Each message is of length m, and so

the length of a complete transcript of the protocol is $2rm$ bits. We assume without loss of generality that the protocol ends with the verifier sending its entire randomness to the prover. To reiterate, our goal is to generate a random transcript of an execution of the proof. We do this bit-by-bit in an iterative manner as follows: Round i begins with a partial transcript prefix γ_{i-1} and a claimed lower bound N_{i-1} on the number of random coins which are consistent with this partial transcript, where $\gamma_0 = \emptyset$ is the empty transcript with the claim that all $N_0 = 2^\ell$ coins are consistent with the empty transcript. By consistent with a partial transcript we mean that had the private-coin verifier received these coins at the beginning of the protocol execution, then this partial transcript would have been generated, with respect to the prover messages. The prover sends two values N_i^0 and N_i^1 where N_i^0 is the number of coins that are consistent with extending the transcript with the bit 0, meaning coins consistent with the transcript $(\gamma_{i-1}, 0)$, and similarly N_i^1 is the number of coins consistent with $(\gamma_{i-1}, 1)$. If the prover can exactly count each of these values, then it should be that $N_{i-1} = N_i^0 + N_i^1$. The verifier tests that indeed $N_{i-1} = N_i^0 + N_i^1$ and chooses a bit b with probability $\frac{N_i^b}{N_{i-1}}$. Both parties set the new transcript to be $\gamma_i = (\gamma_{i-1}, b)$ and the new claim on the number of consistent coins to be $N_i = N_i^b$. This continues on until $i = 2rm$ when a full transcript has been generated, where since we assumed that the verifier ends by outputting its randomness, there can only be one random coin that is consistent with the transcript. Therefore, after the last iteration the verifier tests that the final $N_{2rm} = 1$, and that all verifier messages in the transcript are what V would have sent in an actual execution using randomness from the end of the transcript. Finally, if all these tests pass the verifier and accepts if V accepts given the transcript γ_{2rm}.

For completeness, it can be shown that the protocol described above generates a transcript with the exact same distribution as the original one, since in every stage the next bit of the transcript is chosen with probability equal to the probability that it would appear in a real random transcript conditioned on the part of the transcript that has already been fixed. We now would like to to show that the protocol is sound, i.e. that for $x \notin \mathcal{L}$ a malicious prover cannot cause the verifier to accept in the new protocol with probability greater than in the original protocol. To show this we look the ratio between the number of claimed consistent coins, N and the number of consistent coins that would make the verifier accept in a given round. For a given partial transcript γ we denote by $Acc(\gamma)$ the set of coins ρ such that there exists a legal full transcript of the real execution γ' which begins with γ and in which the verifier accepts.

We begin our inspection of soundness with the final round and work backwards from there. Let $i = 2mr$, and let N_{i-1} and γ_{i-1} be the claim and the partial transcript at the beginning of the iteration. Since the transcript ends with the verifier sending its entire randomness, the number of accepting coins consistent with a transcript with only one bit missing can be 0, 1 or 2. It can be shown that in every case, the probability that the verifier ends up accepting is at most $\frac{|Acc(\gamma_{i-1})|}{N_{i-1}}$. For conciseness we focus in this overview on what happens if $|Acc(\gamma_{i-1})| = 1$. In this case only one of the two options for the final bit will make the verifier accept. Suppose this bit is 0, then the probability that the verifier accepts reduces to the probability that it chooses $b = 0$, which is $\frac{N_i^0}{N_{i-1}}$. Now, since in the end of the protocol the verifier tests that $N_i = N_{2rm} = 1$ in order for the prover to cause the verifier to accept bit 0 it must set $N_i^0 = 1$. Therefore, the probability that the verifier ends up accepting the transcript is at most $\frac{1}{N_{i-1}} = \frac{|Acc(\gamma_{i-1})|}{N_{i-1}}$.

We now look at other rounds of the protocol. Let γ_{i-1} and N_{i-1} be the inputs to iteration i. Suppose, as our induction hypothesis, that upon entering round $i+1$ with γ_i and N_i the probability that the verifier ends up accepting is $\frac{|Acc(\gamma_i)|}{N_i}$. Let N_i^0 and N_i^1 be the values sent

by the prover. By the induction hypothesis, if the verifier chooses bit b, which happens with probability $\frac{N_i^b}{N_{i-1}}$, then it will end up accepting with probability $\frac{|Acc(\gamma_{i-1},b)|}{N_i^b}$. Therefore the probability that the verifier ends up accepting is:

$$\frac{N_i^0}{N_{i-1}} \cdot \frac{|Acc(\gamma_{i-1},0)|}{N_i^0} + \frac{N_i^1}{N_{i-1}} \cdot \frac{|Acc(\gamma_{i-1},1)|}{N_i^1} = \frac{|Acc(\gamma_{i-1},0)| + |Acc(\gamma_{i-1},1)|}{N_{i-1}}$$

Noting that $|Acc(\gamma_{i-1},0)| + |Acc(\gamma_{i-1},1)| = |Acc(\gamma_{i-1})|$ we have that the verifier eventually accepts with probability $\frac{|Acc(\gamma_{i-1})|}{N_{i-1}}$. This inductive argument extends all the way up to γ_0 and N_0 in which case $\frac{|Acc(\gamma_0)|}{N_0}$ is equal to the soundness error of the original protocol.

The actual protocol differs slightly from the one described above. In the real setting, the honest prover cannot exactly calculate N_i^0 and N_i^1, but rather only ε-approximate them. This will mean that the transcript that is sampled is only close to uniform. A further implication of this change is that since the honest prover can err, the verifier now must relax its test that $N_i^0 + N_i^1 = N_{i-1}$. This relaxation turns out to be to test that $\frac{N_{i-1}}{N_i^0+N_i^1} \leq 1 + 3\varepsilon$. This in turn gives the cheating prover some additional leeway, specifically in round i the probability that the verifier ends up accepting changes from $\frac{|Acc(\gamma_{i-1})|}{N_{i-1}}$ to $(1+3\varepsilon)^{2rm-i} \cdot \frac{|Acc(\gamma_{i-1})|}{N_{i-1}}$ (recall that $2rm$ is the number of bits sent in the protocol). If ε is small enough this leeway is insignificant.

References

1 László Babai and Shlomo Moran. Proving properties of interactive proofs by a generalized counting technique. *Inf. Comput.*, 82(2):185–197, 1989. doi:10.1016/0890-5401(89)90053-9.

2 Michael Ben-Or, Oded Goldreich, Shafi Goldwasser, Johan Hρastad, Joe Kilian, Silvio Micali, and Phillip Rogaway. Everything provable is provable in zero-knowledge. In *Advances in Cryptology - CRYPTO '88, 8th Annual International Cryptology Conference, Santa Barbara, California, USA, August 21-25, 1988, Proceedings*, pages 37–56, 1988. doi:10.1007/0-387-34799-2_4.

3 Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *Advances in Cryptology - CRYPTO '86, Santa Barbara, California, USA, 1986, Proceedings*, pages 186–194, 1986. doi:10.1007/3-540-47721-7_12.

4 Martin Fürer, Oded Goldreich, Yishay Mansour, Michael Sipser, and Stathis Zachos. On completeness and soundness in interactive proof systems. *Advances in Computing Research*, 5:429–442, 1989.

5 Oded Goldreich. On doubly-efficient interactive proof systems. *Foundations and Trends in Theoretical Computer Science*, 13(3):158–246, 2018. doi:10.1561/0400000084.

6 Oded Goldreich and Maya Leshkowitz. On emulating interactive proofs with public coins. *Electronic Colloquium on Computational Complexity (ECCC)*, 23:66, 2016. URL: http://eccc.hpi-web.de/report/2016/066.

7 Oded Goldreich, Silvio Micali, and Avi Wigderson. Proofs that yield nothing but their validity and a methodology of cryptographic protocol design (extended abstract). In *27th Annual Symposium on Foundations of Computer Science, Toronto, Canada, 27-29 October 1986*, pages 174–187, 1986. doi:10.1109/SFCS.1986.47.

8 Oded Goldreich and Dana Ron. A sublinear bipartiteness tester for bounded degree graphs. *Comb.*, 19(3):335–373, 1999. doi:10.1007/s004930050060.

9 Oded Goldreich and Dana Ron. Property testing in bounded degree graphs. *Algorithmica*, 32(2):302–343, 2002. doi:10.1007/s00453-001-0078-7.

10 Shafi Goldwasser, Yael Tauman Kalai, and Guy N. Rothblum. Delegating computation: interactive proofs for muggles. In *Proceedings of the 40th Annual ACM Symposium on Theory of Computing, Victoria, British Columbia, Canada, May 17-20, 2008*, pages 113–122, 2008. doi:10.1145/1374376.1374396.

11 Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof-systems (extended abstract). In *Proceedings of the 17th Annual ACM Symposium on Theory of Computing, May 6-8, 1985, Providence, Rhode Island, USA*, pages 291–304, 1985. doi:10.1145/22145.22178.

12 Shafi Goldwasser and Michael Sipser. Private coins versus public coins in interactive proof systems. In *Proceedings of the 18th Annual ACM Symposium on Theory of Computing, May 28-30, 1986, Berkeley, California, USA*, pages 59–68, 1986. doi:10.1145/12130.12137.

13 Iftach Haitner, Mohammad Mahmoody, and David Xiao. A new sampling protocol and applications to basing cryptographic primitives on the hardness of NP. *Electronic Colloquium on Computational Complexity (ECCC)*, 17:1, 2010. URL: http://eccc.hpi-web.de/report/2010/001.

14 Thomas Holenstein and Robin Künzler. A protocol for generating random elements with their probabilities. *CoRR*, abs/1312.2483, 2013. arXiv:1312.2483.

15 Russell Impagliazzo. A personal view of average-case complexity. In *Proceedings of the Tenth Annual Structure in Complexity Theory Conference, Minneapolis, Minnesota, USA, June 19-22, 1995*, pages 134–147, 1995. doi:10.1109/SCT.1995.514853.

16 Russell Impagliazzo and Michael Luby. One-way functions are essential for complexity based cryptography (extended abstract). In *30th Annual Symposium on Foundations of Computer Science, Research Triangle Park, North Carolina, USA, 30 October - 1 November 1989*, pages 230–235, 1989. doi:10.1109/SFCS.1989.63483.

17 Carsten Lund, Lance Fortnow, Howard J. Karloff, and Noam Nisan. Algebraic methods for interactive proof systems. *J. ACM*, 39(4):859–868, 1992. doi:10.1145/146585.146605.

18 Guy N. Rothblum, Salil P. Vadhan, and Avi Wigderson. Interactive proofs of proximity: delegating computation in sublinear time. In *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013*, pages 793–802, 2013. doi:10.1145/2488608.2488709.

19 Adi Shamir. IP = PSPACE. *J. ACM*, 39(4):869–877, 1992. doi:10.1145/146585.146609.

20 Larry J. Stockmeyer. The complexity of approximate counting. In *Proceedings of the 15th Annual ACM Symposium on Theory of Computing, 25-27 April, 1983, Boston, Massachusetts, USA*, pages 118–126, 1983. doi:10.1145/800061.808740.

21 Madhu Sudan. Advanced Complexity Theory (Lecture 14): http://people.csail.mit.edu/madhu/ST07/scribe/lect14.pdf. Scribe notes by Nate Ince and Krzysztof Onak, 2007.

22 Salil P. Vadhan. On transformation of interactive proofs that preserve the prover's complexity. In *Proceedings of the Thirty-Second Annual ACM Symposium on Theory of Computing, May 21-23, 2000, Portland, OR, USA*, pages 200–207, 2000. doi:10.1145/335305.335330.