A PCP Theorem for Interactive Proofs and Applications

Gal Arnon[1⋆] and Alessandro Chiesa[2⋆⋆] and Eylon Yogev[3⋆⋆⋆]

[1] gal.arnon@weizmann.ac.il
Weizmann Institute
[2] alessandro.chiesa@epfl.ch
EPFL
[3] eylon.yogev@biu.ac.il
Bar-Ilan University

Abstract. The celebrated PCP Theorem states that any language in NP can be decided via a verifier that reads $O(1)$ bits from a polynomially long proof. Interactive oracle proofs (IOP), a generalization of PCPs, allow the verifier to interact with the prover for multiple rounds while reading a small number of bits from each prover message. While PCPs are relatively well understood, the power captured by IOPs (beyond NP) has yet to be fully explored.
We present a generalization of the PCP theorem for interactive languages. We show that any language decidable by a $k(n)$-round IP has a $k(n)$-round public-coin IOP, where the verifier makes its decision by reading only $O(1)$ bits from each (polynomially long) prover message and $O(1)$ bits from each of its own (random) messages to the prover.
Our result and the underlying techniques have several applications. We get a new hardness of approximation result for a stochastic satisfiability problem, we show IOP-to-IOP transformations that previously were known to hold only for IPs, and we formulate a new notion of PCPs (index-decodable PCPs) that enables us to obtain a commit-and-prove SNARK in the random oracle model for nondeterministic computations.

Keywords: interactive proofs; probabilistically checkable proofs; interactive oracle proofs

1 Introduction

Probabilistic proofs play a central role in complexity theory and cryptography. In the past decades, probabilistic proofs have become powerful and versatile

⋆ Supported in part by a grant from the Israel Science Foundation (no. 2686/20) and by the Simons Foundation Collaboration on the Theory of Algorithmic Fairness.
⋆⋆ Funded by the Ethereum Foundation.
⋆⋆⋆ Part of this project was performed when Eylon Yogev was in Tel Aviv University where he was funded by the ISF grants 484/18, 1789/19, Len Blavatnik and the Blavatnik Foundation, and The Blavatnik Interdisciplinary Cyber Research Center at Tel Aviv University.

tools in these fields, leading to breakthroughs in zero-knowledge, delegation of computation, hardness of approximation, and other areas.

As an example, interactive proofs (IPs) [40] allow proof-verification to be randomized and interactive, which seemingly confers them much more power than their deterministic (and non-interactive) counterparts. In a k-round IP, a probabilistic polynomial-time verifier exchanges k messages with an all-powerful prover and then accepts or rejects; IP[k] is the class of languages decidable via a k-round interactive proof. Seminal results characterize the power of IPs (IP[poly(n)] = PSPACE) [48, 54] and also achieve zero-knowledge [40, 38].

The development of IPs, in turn, led to probabilistically checkable proofs (PCPs) [4, 36], where a probabilistic polynomial-time verifier has query access to a proof string. Here PCP[r, q] denotes the class of languages decidable by a PCP verifier that uses at most r bits of randomness and queries at most q bits of the proof string. A line of works culminated in the PCP Theorem [2, 1], which can be stated as NP = PCP[$O(\log n), O(1)$]; that is, every language in NP can be decided, with constant soundness error, by probabilistically examining only a constant number of bits in a polynomially long proof.

These advances in probabilistic proofs have reshaped theoretical computer science.

Interactive oracle proofs. More recently, researchers formulated *interactive oracle proofs* (IOPs) [12, 52], a model of probabilistic proof that combines aspects of the IP and PCP models. A k-round IOP is a k-round IP where the verifier has PCP-like access to each prover message: the prover and verifier interact for k rounds, and after the interaction the verifier probabilistically reads a small number of bits from each prover message and decides to accept or reject based on the examined locations. The randomness used in the final phase is called *decision randomness* (which we distinguish from the random messages that the verifier sends to the prover during the interaction).

Recent work has constructed highly-efficient IOPs [9, 7, 18, 11, 10, 8, 13, 26, 53, 19–21]. While the shortest PCPs known to date have quasi-linear length [16, 30], IOPs can achieve linear proof length and fast provers. These developments are at the heart of recent constructions of non-interactive succinct arguments (SNARGs), and have facilitated their deployment in numerous real-world systems. IOPs are also used to construct IPs for delegating computation [52].

IOPs beyond NP*?* Most research regarding IOPs has focused on understanding IOPs for languages in NP (and more generally various forms of non-deterministic computations) while using the additional rounds of interaction to achieve better efficiency compared to PCPs for those languages.

However, the power of IOPs for languages beyond NP is not well understood. We do know that IPs can express all languages in PSPACE for sufficiently large round complexity [48, 54]; moreover more rounds lead to more languages because, under plausible complexity assumptions, it holds that IP[k] $\not\subseteq$ IP[$o(\mathsf{k})$] (while restricting to polynomial communication complexity) [39]. But what can we say

about the power of IOPs *with small query complexity (over the binary alphabet)*?[4] Not much is known about the power of general k-round IOPs, which leads us to ask:

What languages have a k-round IOP where the verifier decides by reading $O(1)$ bits from each prover message and from each verifier message?

1.1 Main results

We answer the above question by showing that (informally) the power of IOPs with k rounds where the verifier reads $O(1)$ bits from each communication round (both prover and verifier messages) is the same as if the verifier reads the entire protocol transcript (as in an IP). This can be seen as extending the PCP Theorem to interactive proofs, interpreted as "*you can be convinced by a conversation while barely listening (even to yourself)*".

To achieve this, our main result is a transformation from IPs to IOPs: we transform any IP into a corresponding IOP where the verifier reads $O(1)$ bits from each communication round and uses a total of $O(\log n)$ bits of decision randomness.[5] The round complexity is preserved, and other parameters are preserved up to polynomial factors. (A round is a verifier message followed by a prover message; after the interaction, the verifier's decision is probabilistic.)

Theorem 1 (IP $\to$ IOP)**.** *Let L be a language with a public-coin IP with k rounds and constant soundness error. Then L has an IOP with k rounds, constant soundness error, where the verifier decides by using $O(\log n)$ bits of decision randomness and reading $O(1)$ bits from each prover message and each verifier message. All other parameters are polynomially related.*

Prior work on IOPs beyond NP. The PCP Theorem can be viewed as a "half-round" IOP with query complexity $O(1)$ and decision randomness $O(\log n)$ for NP. For languages above NP, prior works imply certain facts about k-round IOPs for extreme settings of k.

- For languages that have a public-coin IP with $\mathsf{k} = 1$ round (a verifier message followed by a prover message), Drucker [33] proves a hardness of approximation result in the terminology of CSPs. His result can be re-interpreted showing that these languages have a one-round IOP where the verifier reads $O(1)$ bits from each message and decides (using $O(\log n)$ bits of decision randomness). However, Drucker's result does not extend to arbitrary many rounds.[6]

[4] An IP is an IOP where the verifier has large query complexity over the binary alphabet.

[5] After the interaction, the verifier uses $O(\log n)$ random bits to decide which locations to read from all k rounds.

[6] Round reduction [5] can reduce the number of rounds from any k to 1 with a blow-up in communication that is exponential in k. This does not work when k is super constant; see Section 2.2 for further discussion.

– When k can be polynomially large, we observe that constant-query IOPs for PSPACE can be obtained from [27, 28],[7] which in turn provides such an IOP for every language having an IP. Other analogues of PCP have been given (e.g., [43] applies to the polynomial hierarchy, [32] is also for PSPACE) but they do not seem to translate to IOPs.
– For general k, one can use the fact that AM[k] $\subseteq$ NEXP, and obtain a PCP where the prover sends a single *exponentially-long* message from which the (polynomial-time) verifier reads $O(1)$ bits. However, this does not help if we require the prover to send messages of *polynomial length*.

See Figure 1 for a table summarizing these results and ours.

	complexity class	**model**	**proof length**	**alphabet**	**query complexity**	**round complexity**
[14]	NEXP	PCP	$\exp(\lvert\mathbb{x}\rvert)$	$\{0,1\}$	$O(1)$	1
[28]	PSPACE	IOP	$\text{poly}(\lvert\mathbb{x}\rvert)$	$\{0,1\}$	$O(1)$	$\text{poly}(\lvert\mathbb{x}\rvert)$
implied by [14]	AM[k]	PCP	$\exp(\lvert\mathbb{x}\rvert)$	$\{0,1\}$	$O(1)$	1
[3, 40]	AM[k]	IP	k	$\{0,1\}^{\text{poly}(\lvert\mathbb{x}\rvert)}$	1 per round	k
[this work]	AM[k]	IOP	$\text{poly}(\lvert\mathbb{x}\rvert)$	$\{0,1\}$	$O(1)$ per round	k
[33]	AM	IOP	$\text{poly}(\lvert\mathbb{x}\rvert)$	$\{0,1\}$	$O(1)$	1
[1, 2]	NP	PCP	$\text{poly}(\lvert\mathbb{x}\rvert)$	$\{0,1\}$	$O(1)$	1

Fig. 1. Classes captured by different types of probabilistic proofs (in the regime of constant soundness error). Here, $\mathbb{x}$ denotes the instance whose membership in the language the verifier is deciding. Here, AM stands for two-message public-coin protocols (a verifier random message followed by a prover message), and AM[k] is a k-round public-coin protocol.

Hardness of approximation for stochastic satisfiability We use Theorem 1 to prove the hardness of approximating the value of an instance of the *stochastic satisfiability* (SSAT) problem, which we now informally define.

SSAT is a variant of TQBF (true quantified boolean formulas) where the formula is in 3CNF and the variables are quantified by alternating existential quantifiers and *random* quantifiers (the value of the quantified variable is chosen uniformly at random). A formula ϕ is in the language SSAT if the probability that there is a setting of the existential variables that cause ϕ to be satisfied is greater than 1/2. The *value* of an SSAT instance ϕ is the expected number

[7] Their result shows that PSPACE has what is known as a *probabilistically checkable debate system*. In their system, one prover plays a uniform random strategy. Thus one can naturally translate the debate system into an IOP.

of satisfied clauses in ϕ if the existential variables are chosen to maximize the number of satisfied clauses in ϕ. We denote by k-SSAT the SSAT problem when there are k alternations between existential and random quantifiers.

SSAT can be viewed as a restricted "game against nature" [51] where all parties are binary, and "nature's" moves are made uniformly at random. Variations of SSAT are related to areas of research in artificial intelligence, specifically planning and reasoning under uncertainty [47]. Previous research on SSAT has studied complexity-theoretical aspects [28, 33, 34] and SAT-solvers for it (e.g., [47, 49, 45]).

Our result on the hardness of approximation for k-SSAT is as follows.

Theorem 2. *For every* k*, it is* AM[k]*-complete to distinguish whether a* k-SSAT *instance has value* 1 *or value at most* $1 - \frac{1}{O(\mathsf{k})}$.

We compare Theorem 2 with prior ones about k-SSAT. For $\mathsf{k} = 1$, our result matches that of [33] who showed that the value of 1-SSAT is AM[1]-hard to approximate up to a constant factor. Condon, Feigenbaum, Lund, and Shor [28] show that there exists a constant $c > 1$ such that for every language L in IP = PSPACE, one can reduce an instance $\mathbb{x}$ to a poly($|\mathbb{x}|$)-SSAT instance ϕ such that if $\mathbb{x} \in L$ then the value of ϕ is 1, and otherwise the value of ϕ is $1/c$. The approaches used in both prior works do not seem to extend to other values of k.

This state of affairs motives the following natural question:

How hard is it to approximate the value of k-SSAT *to a constant factor independent of* k*?*

While PCPs have well-known applications to the hardness of approximation of numerous NP problems, no similar connection between IOPs and hardness of approximation was known. (Indeed, this possibility was raised as an open problem in prior work.) The works of Drucker [33] and Condon et al. [28] can be reinterpreted as giving such results for stochastic satisfiability problems. In this paper we make this connection explicit and extend their results.

Transformations for IOPs We obtain IOP analogues of classical IP theorems, as a corollary of Theorem 1. We show IOP-to-IOP transformations, with small query complexity, and achieve classical results that were known for IPs, including: a private-coin to public-coin transformation (in the style of [41]); a round reduction technique (in the style of [5]); and a method to obtain perfect completeness (in the style of [37]). A graphic of this corollary is displayed in Figure 2.

Corollary 1. *Let* L *be a language with a* k*-round IOP with polynomial proof length over a binary alphabet. Then the following holds:*

1. **private-coins to public-coins:** *L has a $O(\mathsf{k})$-round public-coin IOP;*
2. **round reduction:** *for every constant $c \leq \mathsf{k}$, L has a k/c-round IOP;*
3. **perfect completeness:** *L has a perfectly complete k-round IOP.*

All resulting IOPs have polynomial proof length and $O(1)$ per-round query complexity over a binary alphabet; all other parameters are polynomially related to the original IOP.

Similar to the case with IPs, one can combine these transformations to get all properties at once. In particular, one can transform any IOP to be public-coin and have perfect completeness while preserving the round complexity.

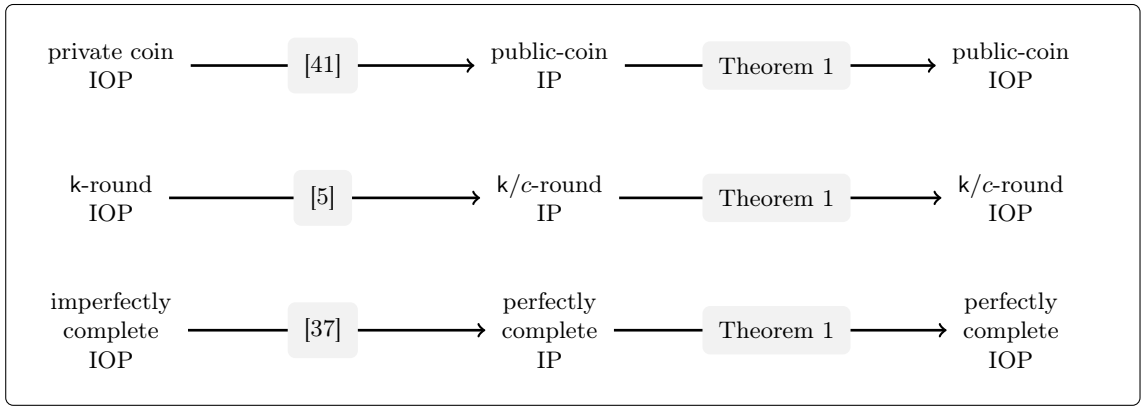

Fig. 2. Corollary 1 provides IOP analogues of classical IP theorems.

1.2 A cryptographic application to SNARKs

A building block that underlies Theorem 1 is a new notion of PCP that we call *index-decodable PCPs*. We informally describe this object in Section 1.2 below (and postpone the definition and a comparison with other PCP notions to Section 2.3). Moreover, we prove that index-decodable PCPs are a useful tool beyond the aforementioned application to Theorem 1, by establishing a generic transformation from index-decodable PCPs to commit-and-prove SNARKs. We discuss these SNARKs and our result in Section 1.2 below (and postpone further discussion to Section 2.6).

Index-decodable PCPs An index-decodable PCP can be seen as a PCP on *maliciously encoded data*. The prover wishes to convince the verifier about a statement that involves k data segments $\mathtt{i}[1], \ldots, \mathtt{i}[\mathsf{k}]$ and an instance $\mathtt{x}$, for example, that it knows a witness $\mathtt{w}$ such that $(\mathtt{i}[1], \ldots, \mathtt{i}[\mathsf{k}], \mathtt{x}, \mathtt{w}) \in R$ for some relation R. The prover outputs a PCP string Π for this statement. The verifier receives as input only the instance $\mathtt{x}$, and is given query access to an *encoding* of each data segment $\mathtt{i}[i]$ and query access to the PCP string Π. This means that the verifier has query access to a total of $\mathsf{k}+1$ oracles.

The definition of an index-decodable PCP, to be useful, needs to take into account several delicate points (which, in fact, are crucial for our proof of Theorem 1).

First, the encoding of each data segment must be computed independently of other data segments and even the instance. (Though the PCP string Π can depend on all data segments and the instance.)

Second, the verifier is not guaranteed that the k data oracles are valid encodings, in the sense that "security" is required to hold even against malicious provers that have full control of all $\mathsf{k}+1$ oracles (not just the PCP string oracle). In other words, we wish to formulate a security notion that is meaningful even for data that has been maliciously encoded.

The security notion that we use is *decodability*. Informally, we require that if the verifier accepts with high-enough probability a given set of (possibly malicious) data oracles and PCP string, then each data oracle can be individually decoded into a data segment and the PCP string can be decoded into a witness such that, collectively, all the data segments, the instance, and the witness form a true statement. We stress that the decoder algorithms must run on each data oracle separately from other data oracles and the instance (similarly as the encoder).

Commit-and-prove SNARKs A *commit-and-prove SNARK* (CaP-SNARK) is a SNARK that enables proving statements about previously committed data, and commitments can be reused across different statements. CaP-SNARKs have been studied in a line of work [35, 29, 46, 22, 17], where constructions have been achieved assuming specific computational assumptions (e.g., knowledge of exponent assumptions) and usually with the added property of zero-knowledge.

We show how to use index-decodable PCPs to unconditionally achieve CaP-SNARKs in the random oracle model (ROM);[8] in more detail we need the index-decodable PCP to have efficient indexing/decoding and certain proximity properties. Our transformation can be seen as an index-decodable PCP analogue of the Micali construction of SNARKs in the ROM from PCPs [50].

Theorem 3. *There is a transformation that takes as input an index-decodable PCP (that has an efficient indexer and decoder and satisfies certain proximity properties) for a relation R with proof length l and query complexity q, and outputs a CaP-SNARK in the ROM for R with argument size $O_\lambda(\mathsf{q} \cdot \log \mathsf{l})$. (Here λ is the output size of the random oracle.)*

We obtain a concrete construction of a CaP-SNARK in the ROM (for nondeterministic computations) by applying the above theorem to our construction of an index-decodable PCP system.

We conclude by noting that the ROM supports several well-known constructions of succinct arguments that can be heuristically instantiated via lightweight cryptographic hash functions, are plausibly post-quantum secure [25], and have

[8] In this model, all parties (honest and malicious) receive query access to the same random function.

led to realizations that are useful in practice. It is plausible that our construction can be shown to have these benefits as well — we leave this, and constructing zero-knowledge CaP-SNARKs in the ROM, to future work.

2 Techniques

We summarize the main ideas underlying our results.

We begin by discussing the question of transforming IPs to IOPs. In Section 2.1, we describe a solution in [33] that works for a single round and explain why it is challenging to extend it for multiple rounds. Then, we describe our transformation for many rounds in two steps. First, in Section 2.2, we describe how to make a verifier query each of its random messages at few locations. Next, in Section 2.3, we define our new notion of *index-decodable PCPs* and, in Section 2.4, describe how to use these to make the verifier query each prover message at few locations (without affecting the first step). In Section 2.5, we explain how to construct index-decodable PCPs with good parameters.

We conclude by describing applications of our results and constructions: (i) in Section 2.6, we construct commit-and-prove SNARKs in the random oracle model from index-decodable PCPs; and (ii) in Section 2.7, we show that Theorem 1 has implications on the hardness of approximating the value of certain stochastic problems.

Throughout, we call *interaction randomness* (or verifier random messages) the randomness sent by the verifier to the prover during the interaction, and *decision randomness* the randomness used by the verifier in the post-interaction decision stage.

2.1 Towards transforming IPs to IOPs

We discuss the problem of transforming IPs into IOPs. We begin by describing a solution in [33] that transforms a single-round IP into a single-round IOP. Following that, we describe the challenges of extending this approach to work for multi-round IPs.

The case of a single-round IP The case of a single-round was settled by Drucker [33], whose work implies a transformation from a public-coin single-round IP to a single-round IOP where the verifier reads $O(1)$ bits from the communication transcript (here consisting of the prover message and the verifier message). His construction uses as building blocks the randomness-efficient amplification technique of [6] and *PCPs of proximity* (PCPPs) [31, 15].[9] We give a high-level overview of his construction.

[9] A PCPP is a PCP system where the verifier has oracle access to its input in addition to the prover's proof; the soundness guarantee is that if the input is *far* (in Hamming distance) from any input in the language, then the verifier accepts with small probability.

In a public-coin single-round IP, given a common input instance $\mathbb{x}$, the verifier $\mathbf{V}_{\mathsf{IP}}$ sends randomness ρ, the prover $\mathbf{P}_{\mathsf{IP}}$ sends a message a, and the verifier $\mathbf{V}_{\mathsf{IP}}$ decides whether to accept by applying a predicate to $(\mathbb{x}, \rho, a)$. Consider the non-deterministic machine M such that $M(\mathbb{x}, \rho) = 1$ if and only if there exists a such that $\mathbf{V}_{\mathsf{IP}}$ accepts $(\mathbb{x}, \rho, a)$. The constructed IOP works as follows:

1. the IOP verifier sends $\mathbf{V}_{\mathsf{IP}}$'s randomness ρ;
2. the IOP prover computes $\mathbf{P}_{\mathsf{IP}}$'s message a and produces a PCPP string Π for the claim "$M(\mathbb{x}, \rho) = 1$";
3. the IOP verifier checks Π using the PCPP verifier with explicit inputs M and $\mathbb{x}$ and implicit input ρ.

This IOP is sound if the underlying IP is "randomness-robust", which means that if $\mathbb{x}$ is not in the language then with high probability over ρ it holds that ρ is *far* from any accepting input for $M(\mathbb{x}, \cdot)$. Drucker achieves this property by using an amplification technique in [6] that achieves soundness error $2^{-|\rho|}$ while using $O(|\rho|)$ random bits (standard amplification would, when starting with a constant-soundness protocol, result in $\omega(|\rho|)$ random bits). Thus, with high probability, ρ is not only a "good" random string (which holds for any single-round IP) but also is δ-far from any "bad" random string, for some small constant $\delta > 0$. This follows since the ball of radius δ around any bad random string has size $2^{\delta'|\rho|}$, for some small constant δ' that depends only on δ.

Challenges of extending the single-round approach to multi-round IPs We wish to obtain a similarly efficient transformation for a public-coin k-round IP where $\mathsf{k} = \mathrm{poly}(n)$.

One possible approach would be to reduce the number of rounds of the given IP from k to 1 and then apply the transformation for single-round IPs. The round reduction of Babai and Moran [5] shows that any public-coin k-round IP can be transformed into a one-round IP where efficiency parameters grow by $n^{O(\mathsf{k})}$. This transformation, however, is not efficient for super-constant values of k. Moreover, it is undesirable even when k is constant because the transformation overhead is not a fixed polynomial (the exponent depends on k rather than being a fixed constant).

Therefore, we seek an approach that directly applies to a multi-round IP. Unfortunately, Drucker's approach for one-round IPs does not generalize to multiple-round IPs for several reasons. First, the corresponding machine $M(\mathbb{x}, \rho_1, \ldots, \rho_{\mathsf{k}})$ (which accepts if and only if there exist prover messages $a_1, \ldots, a_{\mathsf{k}}$ such that $\mathbf{V}_{\mathsf{IP}}$ accepts $(\mathbb{x}, \rho_1, a_1, \ldots, \rho_{\mathsf{k}}, a_{\mathsf{k}})$) does not capture the soundness of the interactive proof because it fails to capture interaction (a protocol may be sound according to the IP definition and, yet, for every $\mathbb{x}$ and $\rho_1, \ldots, \rho_{\mathsf{k}}$ it could be that $M(\mathbb{x}, \rho_1, \ldots, \rho_{\mathsf{k}}) = 1$). Moreover, it is not clear how to perform a randomness-efficient amplification for multiple rounds that makes the protocol sufficiently "randomness robust" for the use of a PCPP. The main reason is that to get soundness error 2^{-m} (as in [33]), the techniques of [6] add $O(m)$ bits *per round*, which is too much when the protocol has many rounds (see Section 2.2 for a more detailed discussion on why this approach fails for many rounds).

We give a different solution that circumvents this step and works for any number of rounds. Our transformation from k-round IP to an IOP in two stages. In the first stage, we transform the IP into one in which the verifier reads only $O(1)$ bits from each random message it sends. In the second stage, we transform the IP into an IOP with $O(1)$ per-round query complexity, simultaneously for each prover message and each verifier message. We achieve this via a new notion of PCPs that we call *index-decodable PCPs*, and we describe in Section 2.3. First, we explain how to achieve the property that the verifier reads $O(1)$ bits from each of its random messages to the prover.

2.2 Local access to randomness

We transform a public-coin IP $(\mathbf{P}_{\mathsf{IP}}, \mathbf{V}_{\mathsf{IP}})$ into an IP $(\mathbf{P}'_{\mathsf{IP}}, \mathbf{V}'_{\mathsf{IP}})$ whose verifier (i) reads $O(1)$ bits from each of its random messages to the prover, and (ii) has logarithmic decision randomness (the randomness used by the verifier in the post-interaction decision stage). For now, the verifier reads in full every message received from the prover, and only later we discuss how to reduce the query complexity to prover messages while preserving the query complexity to the verifier random messages.

One-round public-coin proofs In order to describe our ideas we begin with the simple case of one-round public-coin interactive proofs. Recall from Section 2.1 that this case is solved in [33], but we nevertheless first describe our alternative approach for this case and after that we will discuss the multiple-round case.

A strawman protocol. Recall that in a one-round public-coin IP the verifier sends a uniformly random message, the prover replies with some answer, and the verifier uses both of these messages to decide whether to accept. An idea to allow the verifier to not read in full its own random message would be for the prover to send the received random message back to the verifier, and the verifier to use this latter and test consistency with its own randomness. Given an instance $\mathbb{x}$: $\mathbf{V}'_{\mathsf{IP}}$ sends $\mathbf{V}_{\mathsf{IP}}$'s random message $\rho \in \{0,1\}^{\mathsf{r}}$; $\mathbf{P}'_{\mathsf{IP}}$ replies with $\rho' := \rho$ and the message $a := \mathbf{P}_{\mathsf{IP}}(\mathbb{x}, \rho)$; and $\mathbf{V}'_{\mathsf{IP}}$ checks that ρ and ρ' agree on a random location and that $\mathbf{V}_{\mathsf{IP}}(\mathbb{x}, \rho', a) = 1$.

This new IP is complete, and its verifier queries its random message at one location to conduct the consistency test. However, the protocol might not be sound, as we explain. Suppose that $\mathbb{x} \notin L$. Let r be the length of ρ, let β be the soundness error of the original IP, and let ν_{r} be the volume of the Hamming sphere of radius $\mathsf{r}/3$ in $\{0,1\}^{\mathsf{r}}$. A choice of verifier message ρ is *bad* if there exists a such that $\mathbf{V}_{\mathsf{IP}}(\mathbb{x}, \rho, a) = 1$. By the soundness guarantee of $\mathbf{V}_{\mathsf{IP}}$, the fraction of bad choices of random verifier messages is at most β. A choice of verifier message ρ is *ball-bad* if there exist a bad ρ' that is $1/3$-close to ρ. By the union bound, the fraction of ball-bad coins is at most $\gamma = \beta \cdot \nu_{\mathsf{r}}$.

Let E be the event over the choice of ρ that the prover sends ρ' that is $1/3$-far from ρ.

– Conditioned on E occurring, $\mathbf{V}'_{\mathsf{IP}}$ rejects with probability at least $1/3$ (whenever $\mathbf{V}'_{\mathsf{IP}}$ chooses a location on which ρ and ρ' disagree).
– Conditioned on E not occurring, $\mathbf{P}'_{\mathsf{IP}}$ cannot send any ρ' and a such that $\mathbf{V}_{\mathsf{IP}}(\mathbb{x}, \rho', a) = 1$ unless ρ is ball-bad, and so $\mathbf{V}'_{\mathsf{IP}}$ rejects with probability at least $1 - \gamma$.

Therefore, for the new IP to be sound, we need $\gamma = \beta \cdot \nu_{\mathsf{r}}$ to be small. Notice that $\nu_{\mathsf{r}} = 2^{c \cdot \mathsf{r}}$ (for some constant $0 < c < 1$) depends on r but not on β. Thus we need to achieve $\log 1/\beta > c \cdot \mathsf{r}$. As in Drucker's transformation, this can be done using the randomness-efficient soundness amplification of [6], *but we deliberately take a different approach that will generalize for multiple rounds.*

Shrinking γ using extractors. Let Ext be an extractor with output length r, seed length $O(\log 1/\beta)$, and error β;[10] such extractors are constructed in [42]. Suppose that the prover and verifier have access to a sample z from a source D with high min-entropy. Consider the following IP: $\mathbf{V}'_{\mathsf{IP}}$ sends s; $\mathbf{P}'_{\mathsf{IP}}$ replies with $s' := s$ and $a := \mathbf{P}_{\mathsf{IP}}(\mathbb{x}, \mathsf{Ext}(z, s'))$; $\mathbf{V}'_{\mathsf{IP}}$ checks that s and s' agree on a random location and that $\mathbf{V}_{\mathsf{IP}}(\mathbb{x}, \mathsf{Ext}(z, s'), a) = 1$.

At most a 2β-fraction of the seeds s are such that there exists a such that $\mathbf{V}_{\mathsf{IP}}(\mathbb{x}, \mathsf{Ext}(z, s), a) = 1$, because Ext is an extractor with error β and D is a distribution with high min-entropy. By an identical argument to the one done previously, either $\mathbf{P}'_{\mathsf{IP}}$ sends s' that is far from s and so $\mathbf{V}'_{\mathsf{IP}}$ rejects with constant probability, or $\mathbf{V}'_{\mathsf{IP}}$ rejects with probability at least $\gamma = 2\beta \cdot \nu_{\mathsf{r}'}$ where $\mathsf{r}' = |s| = O(\log 1/\beta)$. Thus we have that $\gamma = 2 \cdot \beta^{1-c}$, which is a constant fraction for small enough values of β (which can be achieved with standard parallel repetition).

Generating a source of high min-entropy. We describe how the prover and verifier can agree on a sample from a high-entropy source by leveraging the following observation: if z is a uniformly random string and z' is an arbitrary string that is close in Hamming distance to z, then z' has high min-entropy. Thus we can sample via similar ideas as above: $\mathbf{V}'_{\mathsf{IP}}$ samples and sends z; $\mathbf{P}'_{\mathsf{IP}}$ replies with $z' := z$; and $\mathbf{V}'_{\mathsf{IP}}$ checks that z and z' agree on a random location. (So $\mathbf{V}'_{\mathsf{IP}}$ reads one bit of its random message z.) If, with constant probability over z, $\mathbf{P}'_{\mathsf{IP}}$ sends z' that is far from z, then $\mathbf{V}'_{\mathsf{IP}}$ rejects with constant probability. Otherwise, we show that z' has high min-entropy because with high probability it agrees with z on most of its locations.

Putting it all together. Let $(\mathbf{P}_{\mathsf{IP}}, \mathbf{V}_{\mathsf{IP}})$ be a public-coin single-round IP with soundness error β and randomness complexity r, and let Ext be an extractor with output length r, seed length $O(\log 1/\beta)$, and error β. The new IP $(\mathbf{P}'_{\mathsf{IP}}, \mathbf{V}'_{\mathsf{IP}})$ is as follows.
– *Sample high min-entropy source:* $\mathbf{V}'_{\mathsf{IP}}$ sends z and $\mathbf{P}'_{\mathsf{IP}}$ replies with $z' := z$.
– *Sample extractor seed:* $\mathbf{V}'_{\mathsf{IP}}$ sends s and $\mathbf{P}'_{\mathsf{IP}}$ replies with $s' := s$.

[10] A function $\mathsf{Ext}\colon \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ is a (k, ε)-extractor if, for every random variable X over $\{0,1\}^n$ with min-entropy at least k, the statistical distance between $\mathsf{Ext}(X, U_d)$ and U_m is at most ε.

– *Prover message:* $\mathbf{P}'_{\mathsf{IP}}$ sends $a := \mathbf{P}_{\mathsf{IP}}(\mathbb{x}, \mathsf{Ext}(z', s'))$.
– *Verification:* $\mathbf{V}'_{\mathsf{IP}}$ checks that z and z' agree on a random location, s and s' agree on a random location, and $\mathbf{V}_{\mathsf{IP}}(\mathbb{x}, \mathsf{Ext}(z', s'), a) = 1$.

Extending to multiple rounds In order to extend the previously described protocol to multiple rounds, we leverage the notion of *round-by-round soundness*. An IP for a language L has round-by-round soundness error β_{rbr} if there exists a "state" function such that: (i) for $\mathbb{x} \notin L$, the starting state is "doomed"; (ii) for every doomed state and next message that a malicious prover might send, with probability β_{rbr} over the verifier's next message, the protocol state will remain doomed; (iii) if at the end of interaction the state is doomed then the verifier rejects.

In the analysis of the one-round case there was an event (called *bad*) over the IP verifier's random message ρ such that if this event does not occur then the prover has no accepting strategy. This event can be replaced, in the round-by-round case, by the event that, in a given round, the verifier chooses randomness where the transcript remains doomed. This idea leads to a natural extension of the one-round protocol described in Section 2.2 to the multi-round case, which is our final protocol.

Let $(\mathbf{P}_{\mathsf{IP}}, \mathbf{V}_{\mathsf{IP}})$ be a public-coin k-round IP with round-by-round soundness error β_{rbr} and randomness complexity r, and Ext an extractor with output length r, seed length $O(\log 1/\beta_{\text{rbr}})$, and error β_{rbr}.

– For each round $j \in [\mathsf{k}]$ of the original IP:
  1. *Sample high min-entropy source:* $\mathbf{V}'_{\mathsf{IP}}$ sends z_j and $\mathbf{P}'_{\mathsf{IP}}$ replies with $z'_j := z_j$.
  2. *Sample extractor seed:* $\mathbf{V}'_{\mathsf{IP}}$ sends s_j and $\mathbf{P}'_{\mathsf{IP}}$ replies with $s'_j := s_j$.
  3. *Prover message:* $\mathbf{P}'_{\mathsf{IP}}$ sends $a_j := \mathbf{P}_{\mathsf{IP}}(\mathbb{x}, \rho_1, \ldots, \rho_j)$ where $\rho_i := \mathsf{Ext}(z_i, s_i)$.
– $\mathbf{V}'_{\mathsf{IP}}$ accepts if and only if the following tests pass:
  1. Choose a random location and, for every $j \in [\mathsf{k}]$, test that z_j and z'_j agree on this location.
  2. Choose a random location and, for every $j \in [\mathsf{k}]$, test that s_j and s'_j agree on this location.
  3. For every $j \in [\mathsf{k}]$, compute $\rho_j := \mathsf{Ext}(z'_j, s'_j)$. Check that $\mathbf{V}_{\mathsf{IP}}(\mathbb{x}, \rho_1, a_1, \ldots, \rho_{\mathsf{k}}, a_{\mathsf{k}}) = 1$.

The soundness analysis of this protocol is similar to the one-round case. Suppose that $\mathbb{x} \notin L$. Then the empty transcript is "doomed". By an analysis similar to the one-round case, except where we set "bad" verifier messages to be ones where the transcript state switches from doomed to not doomed, if a round begins with a doomed transcript then except with probability $\gamma = 2 \cdot \beta_{\text{rbr}}^{1-c}$ (for some constant c) the transcript in the next round is also doomed. Thus, by a union bound, the probability that the transcript ends up doomed, and as a result the verifier rejects, is at least $1 - 2 \cdot \mathsf{k} \cdot \beta_{\text{rbr}}^{1-c}$. As shown in [23] round-by-round soundness error can be reduced via parallel repetition, albeit at a lower rate than regular soundness error. Thus, by doing enough parallel repetition before applying our transformation, the round-by-round soundness error β_{rbr} can be reduced enough so that the verifier rejects with constant probability.

The above protocol has $2\mathsf{k}_{\mathsf{IP}}$ rounds. The verifier reads 1 bit from each of its random messages, and has $O(\log|\mathbb{x}|)$ bits of decision randomness (to sample random locations for testing consistency between each z'_j and z_j and between each s'_j and s_j). To achieve k_{IP} rounds, we first apply the round reduction of [5] on the original IP to reduce to $\mathsf{k}_{\mathsf{IP}}/2$ rounds, and then apply our transformation.

Why randomness-efficient soundness amplification is insufficient We briefly sketch why applying randomness-efficient soundness amplification in the style of [6] is insufficient in the multi-round case, even if we were to consider round-by-round soundness. Recall that we wish for $\beta_{\text{rbr}} \cdot 2^{\Theta(\mathsf{r})}$ to be small, where β_{rbr} is the round-by-round soundness of the protocol and r is the number of random bits sent by the verifier in a single round. Bellare, Goldreich and Goldwasser [6] show that, starting with constant soundness and randomness r, one can achieve soundness error 2^{-m} using $\mathsf{r}' = O(\mathsf{r}+m)$ random bits; they do this via m parallel repetitions where the randomness between repetitions is shared in a clever way. Using parallel repetition, achieving round-by-round soundness error 2^{-m} requires m/k repetitions (see [23]). Thus, even if we were to show that the transformation of [6] reduces round-by-round soundness error at the same rate as standard parallel repetition (as it does for standard soundness), in order to get round-by-round soundness error 2^{-m}, we would need $\mathsf{r}' = O(\mathsf{r} + m \cdot \mathsf{k})$ bits of randomness. This would achieve $\beta_{\text{rbr}} \cdot 2^{\Theta(\mathsf{r}')} = 2^{-m} \cdot 2^{\Theta(\mathsf{r}+m\mathsf{k})}$, which, for super-constant values of k, is greater than 1 regardless of r.

2.3 Index-decodable PCPs

We introduce *index-decodable PCPs*, a notion of PCP that works on *multi-indexed relations*. A multi-indexed relation R is a set of tuples $(\mathbb{i}[1], \ldots, \mathbb{i}[\mathsf{k}], \mathbb{x}, \mathbb{w})$ where $(\mathbb{i}[1], \ldots, \mathbb{i}[\mathsf{k}])$ is the index vector, $\mathbb{x}$ the instance, and $\mathbb{w}$ the witness. As seen in the following definition, an index-decodable PCP treats the index vector $(\mathbb{i}[1], \ldots, \mathbb{i}[\mathsf{k}])$ and the instance $\mathbb{x}$ differently, which is why they are not "merged" into an instance $\mathbb{x}' = (\mathbb{i}[1], \ldots, \mathbb{i}[\mathsf{k}], \mathbb{x})$ (and why we do not consider standard relations).

Definition 1. *An* **index-decodable PCP** *for a multi-indexed relation* $R = \{ (\mathbb{i}[1], \ldots, \mathbb{i}[\mathsf{k}], \mathbb{x}, \mathbb{w}) \}$ *is a tuple of algorithms* $(\mathbf{I}_{\mathsf{PCP}}, \mathbf{P}_{\mathsf{PCP}}, \mathbf{V}_{\mathsf{PCP}}, \mathbf{iD}_{\mathsf{PCP}}, \mathbf{wD}_{\mathsf{PCP}})$*, where* $\mathbf{I}_{\mathsf{PCP}}$ *is the (honest) indexer,* $\mathbf{P}_{\mathsf{PCP}}$ *the (honest) prover,* $\mathbf{V}_{\mathsf{PCP}}$ *the verifier,* $\mathbf{iD}_{\mathsf{PCP}}$ *the index decoder, and* $\mathbf{wD}_{\mathsf{PCP}}$ *the witness decoder. The system has (perfect completeness and)* decodability bound κ_{PCP} *if the following conditions hold.*

– ***Completeness.*** *For every* $(\mathbb{i}[1], \ldots, \mathbb{i}[\mathsf{k}], \mathbb{x}, \mathbb{w}) \in R$*,*

$$\Pr_{\rho}\left[\mathbf{V}_{\mathsf{PCP}}^{\pi_1,\ldots,\pi_{\mathsf{k}},\Pi}(\mathbb{x};\rho) = 1 \;\middle|\; \begin{array}{r} \pi_1 \leftarrow \mathbf{I}_{\mathsf{PCP}}(\mathbb{i}[1]) \\ \vdots \\ \pi_{\mathsf{k}} \leftarrow \mathbf{I}_{\mathsf{PCP}}(\mathbb{i}[\mathsf{k}]) \\ \Pi \leftarrow \mathbf{P}_{\mathsf{PCP}}(\mathbb{i}[1], \ldots, \mathbb{i}[\mathsf{k}], \mathbb{x}, \mathbb{w}) \end{array}\right] = 1 \ .$$

– ***Decodability.*** *For every* $\mathbb{x}$*, indexer proofs* $\tilde{\pi}_1, \ldots, \tilde{\pi}_{\mathsf{k}}$*, and malicious prover proof* $\tilde{\Pi}$*, if*

$$\Pr_{\rho}\left[\, \mathbf{V}_{\mathsf{PCP}}^{\tilde{\pi}_1,\ldots,\tilde{\pi}_{\mathsf{k}},\tilde{\Pi}}(\mathbb{x};\rho) = 1 \,\right] > \kappa_{\mathsf{PCP}}(|\mathbb{x}|)$$

then $\left(\mathbf{iD}_{\mathsf{PCP}}(\tilde{\pi}_1), \ldots, \mathbf{iD}_{\mathsf{PCP}}(\tilde{\pi}_{\mathsf{k}}), \mathbb{x}, \mathbf{wD}_{\mathsf{PCP}}(\tilde{\Pi})\right) \in R$.

The indexer $\mathbf{I}_{\mathsf{PCP}}$ separately encodes each index, independent of indices and the instance, to obtain a corresponding *indexer proof*. The prover $\mathbf{P}_{\mathsf{PCP}}$ gets all the data as input (index vector, instance, and witness) and outputs a *prover proof*. The verifier $\mathbf{V}_{\mathsf{PCP}}$ gets the instance as input and has query access to $\mathsf{k}+1$ oracles (k indexer proofs and 1 prover proof), and outputs a bit.

The decodability condition warrants some discussion. The usual soundness condition of a PCP for a standard relation R has the following form: "if $\mathbf{V}_{\mathsf{PCP}}^{\tilde{\Pi}}(\mathbb{x})$ accepts with high-enough probability then there exists a witness $\mathbb{w}$ such that $(\mathbb{x}, \mathbb{w}) \in R$". For a multi-indexed relation it could be that for any given instance $\mathbb{x}$ there exist indexes $\mathbb{i}[1], \ldots, \mathbb{i}[\mathsf{k}]$ and a witness $\mathbb{w}$ such that $(\mathbb{i}[1], \ldots, \mathbb{i}[\mathsf{k}], \mathbb{x}, \mathbb{w}) \in R$. Since we do not trust the indexer's outputs, a soundness condition is not meaningful.

Instead, the decodability condition that we consider has the following form: "if $\mathbf{V}_{\mathsf{PCP}}^{\tilde{\pi}_1,\ldots,\tilde{\pi}_{\mathsf{k}},\tilde{\Pi}}(\mathbb{x})$ accepts with high-enough probability then $(\mathbb{i}[1], \ldots, \mathbb{i}[\mathsf{k}], \mathbb{x}, \mathbb{w}) \in R$ where $\mathbb{i}[1], \ldots, \mathbb{i}[\mathsf{k}]$ and $\mathbb{w}$ are the decoded indices and witness respectively found in $\tilde{\pi}_1, \ldots, \tilde{\pi}_{\mathsf{k}}$ and $\tilde{\Pi}$". It is crucial that the index decoder receives as input the relevant indexer proof but not also the instance, or else the decodability condition would be trivially satisfied (the index decoder could output the relevant index of the lexicographically first index vector putting the instance in the relation). This ensures that the proofs collectively convince the verifier not only that there exists an index vector and witness that place the instance in the relation, but that the prover encoded a witness that, along with index vector obtainable from the index oracles via the index decoder, places the instance in the relation.

We do not require the indexer or the decoders to be efficient. However, in some applications, it is useful to have an efficient indexer and decoders, and indeed we construct an index-decodable PCP with an efficient indexer and decoders.

Remark 1 (comparison with holography). We compare index-decodable PCPs and holographic PCPs, which also work for indexed relations (see [24] and references therein). In both cases, an indexer produces an encoding of the index (independent of the instance). However, there are key differences between the two: (i) in an index-decodable PCP the indexer works separately on each entry of the index vector, while in a holographic PCP there is a single index; moreover, (ii) in a holographic PCP the indexer is trusted in the sense that security is required to hold only when the verifier has oracle access to the honest indexer's output, but in an index-decodable PCP, the indexer is **not trusted** in the sense that the malicious prover can choose encodings for all of the indices. *Both differences are essential properties for our transformation of IPs into IOPs.*

We construct a binary index-decodable PCPs with $O(1)$ query complexity per oracle.

Theorem 4. *Any multi-indexed relation* $R = \{(\mathtt{i}[1], \ldots, \mathtt{i}[\mathsf{k}], \mathbb{x}, \mathbb{w})\}$ *to which membership can be verified in nondeterministic time* T *has a non-adaptive index-decodable PCP with the following parameters:*

Index-Decodable PCP for $(\mathtt{i}[1], \ldots, \mathtt{i}[\mathsf{k}], \mathbb{x}, \mathbb{w}) \in R$	
Indexer proof length (per proof)	$O(\lvert\mathtt{i}[i]\rvert)$
Prover proof length	$\mathrm{poly}(T)$
Alphabet size	2
Queries per oracle	$O(1)$
Randomness	$O(\log \lvert\mathbb{x}\rvert)$
Decodability bound	$O(1)$
Indexer running time	$\tilde{O}(\lvert\mathtt{i}[i]\rvert)$
Prover running time	$\mathrm{poly}(T)$
Verifier running time	$\mathrm{poly}(\lvert\mathbb{x}\rvert, \mathsf{k}, \log T)$
Index decoding running	$\tilde{O}(\lvert\mathtt{i}[i]\rvert)$
Witness decoding time	$\mathrm{poly}(T)$

Our construction achieves optimal parameters similar to the PCP theorem: it has $O(1)$ query complexity (per oracle) over a binary alphabet, and the randomness complexity is logarithmic, *independent* of the number of indexes k. Achieving small randomness complexity is challenging and useful. First, it facilitates proof composition (where a prover writes a proof for every possible random string), which is common when constructing zero-knowledge PCPs (e.g., [44]). Second, small randomness complexity is necessary for our hardness of approximation results (see Section 2.7).

A similar notion is (implicitly) considered in [1] but their construction does not achieve the parameters we obtain in Theorem 4 (most crucially, they do not achieve small randomness).

2.4 Local access to prover messages

We show how to transform an IP into an IOP by eliminating the need of the verifier to read more than a few bits of each prover message. This transformation preserves the number of bits read by the verifier to its own interaction randomness. Thus, combining it with the transformation described in Section 2.2, this completes the proof (overview) of Theorem 1.

We transform any public-coin IP into an IOP by using an index-decodable PCP. In a public-coin k-round IP, the prover $\mathbf{P}_{\mathsf{IP}}$ and verifier $\mathbf{V}_{\mathsf{IP}}$ receive as input an instance $\mathbb{x}$ and then, in each round i, the verifier $\mathbf{V}_{\mathsf{IP}}$ sends randomness ρ_i and the prover replies with a message $a_i \leftarrow \mathbf{P}_{\mathsf{IP}}(\mathbb{x}, \rho_1, \ldots, \rho_i)$; after the interaction, the verifier $\mathbf{V}_{\mathsf{IP}}$ runs an efficient probabilistic algorithm with decision randomness ρ_{dc} on the transcript $(\mathbb{x}, \rho_1, a_1, \ldots, \rho_{\mathsf{k}}, a_{\mathsf{k}})$ to decide whether to accept or reject.

The IP verifier $\mathbf{V}_{\mathsf{IP}}$ defines a multi-indexed relation $R(\mathbf{V}_{\mathsf{IP}})$ consisting of tuples

$$\Big(\mathtt{i}[1], \ldots, \mathtt{i}[\mathsf{k}], \mathbb{x}', \mathbb{w}\Big) = \Big(a_1, \ldots, a_{\mathsf{k}}, (\mathbb{x}, \rho_1, \ldots, \rho_{\mathsf{k}}, \rho_{\mathsf{dc}}), \bot\Big)$$

such that the IP verifier $\mathbf{V}_{\mathsf{IP}}$ accepts the instance $\mathbb{x}$, transcript $(\rho_1, a_1, \ldots, \rho_{\mathsf{k}}, a_{\mathsf{k}})$, and decision randomness ρ_{dc}. (Here we do not rely on witnesses although the definition of index-decodable PCPs supports this.)

From IP to IOP. Let $(\mathbf{I}_{\mathsf{PCP}}, \mathbf{P}_{\mathsf{PCP}}, \mathbf{V}_{\mathsf{PCP}}, \mathbf{iD}_{\mathsf{PCP}}, \mathbf{wD}_{\mathsf{PCP}})$ be an index-decodable PCP for the relation $R(\mathbf{V}_{\mathsf{IP}})$. We construct the IOP as follows. The IOP prover and IOP verifier receive an instance $\mathbb{x}$. In round $i \in [\mathsf{k}]$, the IOP verifier sends randomness ρ_i (just like the IP verifier $\mathbf{V}_{\mathsf{IP}}$) and the (honest) IOP prover sends the indexer proof $\pi_i := \mathbf{I}_{\mathsf{PCP}}(a_i)$ where $a_i \leftarrow \mathbf{P}_{\mathsf{IP}}(\mathbb{x}, \rho_1, \ldots, \rho_i)$. In a final additional message (which can be sent at the same time as the last indexer proof π_{k}), the IOP prover sends $\Pi := \{\Pi_{\rho_{\mathsf{dc}}}\}_{\rho_{\mathsf{dc}}}$ where, for every possible choice of decision randomness ρ_{dc}, $\Pi_{\rho_{\mathsf{dc}}}$ is an index-decodable PCP prover proof to the fact that $\big(a_1, \ldots, a_{\mathsf{k}}, (\mathbb{x}, \rho_1, \ldots, \rho_{\mathsf{k}}, \rho_{\mathsf{dc}}), \bot\big) \in R(\mathbf{V}_{\mathsf{IP}})$. After the interaction, the IOP verifier samples IP decision randomness ρ_{dc} and checks that $\mathbf{V}_{\mathsf{PCP}}^{\tilde{\pi}_1, \ldots, \tilde{\pi}_{\mathsf{k}}, \tilde{\Pi}_{\rho_{\mathsf{dc}}}}\big(\mathbb{x}, \rho_1, \ldots, \rho_{\mathsf{k}}, \rho_{\mathsf{dc}}\big) = 1$.

Proof sketch. Completeness follows straightforwardly from the construction. We now sketch a proof of soundness. Letting L be the language decided by $(\mathbf{P}_{\mathsf{IP}}, \mathbf{V}_{\mathsf{IP}})$, fix an instance $\mathbb{x} \notin L$ and a malicious IOP prover $\tilde{\mathbf{P}}_{\mathsf{IOP}}$. Given interaction randomness $\rho_1, \ldots, \rho_{\mathsf{k}}$, consider the messages $\tilde{\pi}_1, \ldots, \tilde{\pi}_{\mathsf{k}}$ output by $\tilde{\mathbf{P}}_{\mathsf{IOP}}$ in the relevant rounds ($\tilde{\pi}_i$ depends on $\rho_1, \ldots, \rho_i$) and the message $\tilde{\Pi} = \{\tilde{\Pi}_{\rho_{\mathsf{dc}}}\}_{\rho_{\mathsf{dc}}}$ output by $\tilde{\mathbf{P}}_{\mathsf{IOP}}$ in the last round (this message depends on $\rho_1, \ldots, \rho_{\mathsf{k}}$). We consider two complementary options of events over the IOP verifier's randomness $(\rho_1, \ldots, \rho_{\mathsf{k}}, \rho_{\mathsf{dc}})$.

1. With high probability the proofs $\tilde{\pi}_1, \ldots, \tilde{\pi}_{\mathsf{k}}$ and $\tilde{\Pi}_{\rho_{\mathsf{dc}}}$ generated while interacting with $\tilde{\mathbf{P}}_{\mathsf{IOP}}$ using randomness $\rho_1, \ldots, \rho_{\mathsf{k}}$ and ρ_{dc} are such that

$$\Big(\mathbf{iD}_{\mathsf{PCP}}(\tilde{\pi}_1), \ldots, \mathbf{iD}_{\mathsf{PCP}}(\tilde{\pi}_{\mathsf{k}}), (\mathbb{x}, \rho_1, \ldots, \rho_{\mathsf{k}}, \rho_{\mathsf{dc}}), \bot\Big) \notin R(\mathbf{V}_{\mathsf{IP}}) \ .$$

   If this is true, then, by the decodability property of the index-decodable PCP, the IOP verifier must reject with high probability over the choice of randomness for $\mathbf{V}_{\mathsf{PCP}}$.
2. With high probability the proofs $\tilde{\pi}_1, \ldots, \tilde{\pi}_{\mathsf{k}}$ and $\tilde{\Pi}_{\rho_{\mathsf{dc}}}$ generated while interacting with $\tilde{\mathbf{P}}_{\mathsf{IOP}}$ using randomness $\rho_1, \ldots, \rho_{\mathsf{k}}$ and ρ_{dc} are such that

$$\Big(\mathbf{iD}_{\mathsf{PCP}}(\tilde{\pi}_1), \ldots, \mathbf{iD}_{\mathsf{PCP}}(\tilde{\pi}_{\mathsf{k}}), (\mathbb{x}, \rho_1, \ldots, \rho_{\mathsf{k}}, \rho_{\mathsf{dc}}), \bot\Big) \in R(\mathbf{V}_{\mathsf{IP}}) \ .$$

   We prove that this case cannot occur by showing that it contradicts the soundness of the original IP. Suppose towards contradiction that the above is true. We use $\tilde{\mathbf{P}}_{\mathsf{IOP}}$ and the index decoder of the index-decodable PCP, $\mathbf{iD}_{\mathsf{PCP}}$, to construct a malicious IP prover for the original IP as follows.
   In round i, the transcript $(\rho_1, a_1, \ldots, \rho_{i-1}, a_{i-1})$ has already been set during previous interaction. The IP verifier sends randomness ρ_i. The IP prover sends $a_i := \mathbf{iD}_{\mathsf{PCP}}(\tilde{\pi}_i)$ to the IP verifier, where $\tilde{\pi}_i := \tilde{\mathbf{P}}_{\mathsf{IOP}}(\rho_1, \ldots, \rho_i)$. Recall that $\big(\mathbf{D}_{\mathsf{PCP}}(\tilde{\pi}_1), \ldots, \mathbf{D}_{\mathsf{PCP}}(\tilde{\pi}_{\mathsf{k}}), (\mathbb{x}, \rho_1, \ldots, \rho_{\mathsf{k}}, \rho_{\mathsf{dc}}), \bot\big) \in R(\mathbf{V}_{\mathsf{IP}})$ if and only if the

IP verifier accepts given instance $\mathbb{x}$, randomness $(\rho_1, \ldots, \rho_{\mathsf{k}}, \rho_{\mathsf{dc}})$, and prover messages $\mathbf{D}_{\mathsf{PCP}}(\tilde{\pi}_1), \ldots, \mathbf{D}_{\mathsf{PCP}}(\tilde{\pi}_{\mathsf{k}})$, which is precisely what the IP prover supplies it with. Since the event that $\left(\mathbf{D}_{\mathsf{PCP}}(\tilde{\pi}_1), \ldots, \mathbf{D}_{\mathsf{PCP}}(\tilde{\pi}_{\mathsf{k}}), (\mathbb{x}, \rho_1, \ldots, \rho_{\mathsf{k}}, \rho_{\mathsf{dc}}), \bot\right) \in R(\mathbf{V}_{\mathsf{IP}})$ happens with high probability, this implies that with high probability the IP verifier will accept, contradicting soundness of the original IP. Here we crucially used the fact that the decoder $\mathbf{D}_{\mathsf{PCP}}$ does not depend on the instance of the index-decodable PCP (which consists of $\mathbb{x}$ and all of the IP verifier's randomness $\rho_1, \ldots, \rho_{\mathsf{k}}, \rho_{\mathsf{dc}}$) or on the other indexer messages.

The resulting IOP has k rounds, exactly as in the original IP. The IOP verifier uses as much randomness as the original IP verifier with the addition of the randomness used by the index-decodable PCP. The query complexity is that of the underlying verifier of the index-decodable PCP. The proof length and alphabet are the same as those of the index-decodable PCP.

Preserving local access to randomness. The transformation described above can be modified to preserve the query complexity of the verifier to its own interaction randomness if the verifier is non-adaptive with respect to its queries to its random messages (i.e., the choice of bits that it reads depends only on $\mathbb{x}$ and ρ_{dc}). We can redefine the multi-indexed relation $R(\mathbf{V}_{\mathsf{IP}})$ to have as explicit inputs the instance $\mathbb{x}$, decision randomness ρ_{dc}, and the bits of $\rho_1, \ldots, \rho_{\mathsf{k}}$ that the verifier needs to read to decide whether to accept or reject (rather than the entire interaction randomness strings). In more detail, suppose that the verifier reads q bits from its own interaction randomness. Then the new multi-indexed relation consists of tuples:

$$\left(\mathbb{i}[1], \ldots, \mathbb{i}[\mathsf{k}], \mathbb{x}', \mathbb{w}\right) = \left(a_1, \ldots, a_{\mathsf{k}}, (\mathbb{x}, b_1, \ldots, b_{\mathsf{q}}, \rho_{\mathsf{dc}}), \bot\right)$$

such that given decision randomness ρ_{dc} the IP verifier $\mathbf{V}_{\mathsf{IP}}$ accepts given instance $\mathbb{x}$, decision randomness ρ_{dc}, prover messages $(a_1, \ldots, a_{\mathsf{k}})$, and $(b_1, \ldots, b_{\mathsf{q}})$ as answers to its q queries to $\rho_1, \ldots, \rho_{\mathsf{k}}$.

Given a multi-indexed PCP for this relation, the IP to IOP transformation is identical to the one described above, except that after the interaction, the IOP verifier samples IP decision randomness, queries its own interaction randomness to get answers $b_1, \ldots, b_{\mathsf{q}}$, and these replace $\rho_1, \ldots, \rho_{\mathsf{k}}$ as explicit inputs to the index-decodable PCP verifier $\mathbf{V}_{\mathsf{PCP}}$.

2.5 Constructing index-decodable PCPs

We describe how to construct index-decodable PCPs: in Section 2.5 we outline a randomness-efficient index-decodable PCP that makes $O(1)$ queries to each of its oracles, where the indexer proofs are over the binary alphabet and the prover proof is over a large alphabet; then in Section 2.5 we use proof composition to reduce the alphabet size of the latter.

Basic construction from PCPPs We outline a construction of an index-decodable PCP with $O(1)$ query complexity to each indexer proof and to the

prover proof, and where the prover proof is over a large alphabet (of size 2^{k}). For a later proof composition while preserving polynomial proof length, here we additionally require that the verifier has logarithmic randomness complexity.

Building blocks. In our construction we rely on variants of PCPPs. Recall that a PCPP is a PCP system where the verifier has oracle access to its input in addition to the prover's proof; the soundness guarantee is that if the input is *far* (in Hamming distance) from any input in the language, then the verifier accepts with small probability.

We use PCPPs that are *multi-input* and *oblivious*. We explain each of these properties.

- A PCPP is multi-input if the verifier has oracle access to multiple (oracle) inputs. The soundness guarantee is that, for every vector of inputs that satisfy the machine in question, if at least one input oracle is far from the respective satisfying input, then the verifier accepts with small probability.
- A (non-adaptive) PCPP is oblivious for a family of nondeterministic machines $\mathcal{M} = \{M_i\}_{i \in [\mathsf{k}]}$ if the queries made by the verifier to its oracles depend only on $\mathcal{M}$ and its randomness. In particular they do not depend on i. This property will be used later to facilitate bundling queries. We will have k PCPs, each with a different M_i, but the verifier will use the same randomness in each test. Since the PCPPs are oblivious, this means that the verifier makes the same queries for every test. Thus we can group together the k proofs into a single proof with larger alphabet and maintain good query complexity on this proof. This property is important in order to achieve our final parameters.

See the full version of this paper for formal definitions for the above notions, and how to obtain them from standard PCPPs. Henceforth, all PCPPs that we use will be over the binary alphabet and have constant proximity, constant soundness error, constant query complexity, and logarithmic randomness complexity.

The construction. We construct an index-decodable PCP for a multi-indexed relation $R = \{(\mathbb{i}[1], \ldots, \mathbb{i}[\mathsf{k}], \mathbb{x}, \mathbb{w})\}$ whose membership can be verified efficiently.

The indexer encodes each index via an error-correcting code with (constant) relative distance greater than the (constant) proximity parameter of the PCPP used later. The prover uses PCPPs to prove that there exist indexes and a witness that put the given instance in the relation and adds consistency checks to prove that the indices are consistent with those encoded by the indexer. The verifier checks each of these claims. The index decoder decodes the indexer proofs using the same code.

In slightly more detail, the index-decodable PCP is as follows.

- $\mathbf{I}_{\mathsf{PCP}}(\mathbb{i}[i])$: Encode the index $\mathbb{i}[i]$ as π_i using an error-correcting code.
- $\mathbf{P}_{\mathsf{PCP}}(\mathbb{i}[1], \ldots, \mathbb{i}[\mathsf{k}], \mathbb{x}, \mathbb{w})$:
  1. *Encoding the indexes.* Compute Π_*, an encoding of the string $(\mathbb{i}[1], \ldots, \mathbb{i}[\mathsf{k}], \mathbb{w})$.
  2. *Membership of encoding.* Compute a PCPP string Π_{mem} for the claim that $M_*(\mathbb{x}, \Pi_*) = 1$ where M_* checks that Π_* is a valid encoding of indexes and a witness that put $\mathbb{x}$ in R.

3. *Consistency of encoding.* For every $j \in [\mathsf{k}]$, compute a PCPP string Π_j for the claim that $M_j(\pi_j, \Pi_*) = 1$ where M_j checks that π_j and Π_* are valid encodings and that the string $\mathbb{i}[j]$ encoded within π_j is equal to the matching string encoded within Π_*.
4. Output $(\Pi_*, \Pi_{\mathsf{mem}}, \Pi_{\mathbb{i}})$ where $\Pi_{\mathbb{i}}$ are the proofs $\Pi_1, \ldots, \Pi_{\mathsf{k}}$ "bundled" together into symbols of k bits such that $\Pi_{\mathbb{i}}[q] = (\Pi_1[q], \ldots, \Pi_{\mathsf{k}}[q])$.

– $\mathbf{V}_{\mathsf{PCP}}^{\tilde{\pi}_1,\ldots,\tilde{\pi}_{\mathsf{k}},(\tilde{\Pi}_*,\tilde{\Pi}_{\mathsf{mem}},\tilde{\Pi}_{\mathbb{i}})}(\mathbb{x})$: Check that all the tests below pass.
  1. *Membership.* Run the PCPP verifier on the claim that $M_*(\mathbb{x}, \tilde{\Pi}_*) = 1$ using proof oracle $\tilde{\Pi}_{\mathsf{mem}}$.
  2. *Consistency.* For every $j \in [\mathsf{k}]$, run the PCPP verifier on the claim that $M_j(\tilde{\pi}_j, \tilde{\Pi}_*) = 1$ using proof oracle $\tilde{\Pi}_j$. These k tests are run *with the same randomness*. Since the PCPP is oblivious and randomness is shared, the queries made by the PCPP verifier in each test are identical, and so each query can be made by reading the appropriate k-bit symbols from $\tilde{\Pi}_{\mathbb{i}}$.
– $\mathbf{iD}_{\mathsf{PCP}}(\tilde{\pi}_j)$: output the codeword closest to $\tilde{\pi}_j$ in the error-correcting code.
– $\mathbf{wD}_{\mathsf{PCP}}(\tilde{\Pi}_*, \tilde{\Pi}_{\mathsf{mem}}, \tilde{\Pi}_{\mathbb{i}})$: Let $(\tilde{\mathbb{i}}[1], \ldots, \tilde{\mathbb{i}}[\mathsf{k}], \tilde{\mathbb{w}})$ be the codeword closest to $\tilde{\Pi}_*$ in the error-correcting code and output $\tilde{\mathbb{w}}$.

Completeness follows straightforwardly from the construction. We now sketch decodability.

Decodability. Fix an instance $\mathbb{x}$, indexer proofs $\tilde{\pi}_1, \ldots, \tilde{\pi}_{\mathsf{k}}$, and prover proof $(\tilde{\Pi}_*, \tilde{\Pi}_{\mathsf{mem}}, \tilde{\Pi}_{\mathbb{i}})$. Suppose that the verifier accepts with high-enough probability. We argue that this implies that there exists $\mathbb{w}$ such that $\left(\mathbf{iD}_{\mathsf{PCP}}(\tilde{\pi}_1), \ldots, \mathbf{iD}_{\mathsf{PCP}}(\tilde{\pi}_{\mathsf{k}}), \mathbb{x}, \mathbf{wD}_{\mathsf{PCP}}(\tilde{\Pi})\right) \in R$. Specifically, we argue that $\tilde{\Pi}_*$ encodes indices $\tilde{\mathbb{i}}[1], \ldots, \tilde{\mathbb{i}}[\mathsf{k}]$ and witness $\tilde{\mathbb{w}}$ that place $\mathbb{x}$ in R and, additionally, each $\tilde{\pi}_j$ is an encoding of $\tilde{\mathbb{i}}[j]$. This completes the proof of decodability because $\mathbf{iD}_{\mathsf{PCP}}$ decodes each $\tilde{\pi}_j$ to $\tilde{\mathbb{i}}[j]$, and these strings together with $\tilde{\mathbb{w}}$ put $\mathbb{x}$ in the multi-indexed relation R.

Let δ_{PCPP} be the PCPP's proximity and δ_{ECC} the code's (relative) distance; recall that $\delta_{\mathsf{PCPP}} \leq \delta_{\mathsf{ECC}}$.

– *Membership*: We claim that there exist strings $\tilde{\mathbb{i}}[1], \ldots, \tilde{\mathbb{i}}[\mathsf{k}]$ and $\tilde{\mathbb{w}}$ that place $\mathbb{x}$ in R and whose encoding has Hamming distance at most δ_{PCPP} from $\tilde{\Pi}_*$; since $\delta_{\mathsf{PCPP}} \leq \delta_{\mathsf{ECC}}$, this implies that $\tilde{\Pi}_*$ decodes to $(\tilde{\mathbb{i}}[1], \ldots, \tilde{\mathbb{i}}[\mathsf{k}], \tilde{\mathbb{w}})$. Suppose towards contradiction that there are no such strings. In other words, for every codeword $\widehat{\Pi}_*$ that is close in Hamming distance to $\tilde{\Pi}_*$ we have that $M_{*,\mathbb{x}}(\mathbb{x}, \widehat{\Pi}_*) = 0$. As a result the PCPP verifier must reject with high probability, which contradicts our assumption that $\mathbf{V}_{\mathsf{PCP}}$ (which runs the PCPP verifier) *accepts* with high probability.
– *Consistency*: We claim that there exist strings $\tilde{\mathbb{i}}[1], \ldots, \tilde{\mathbb{i}}[\mathsf{k}]$ and $\tilde{\mathbb{w}}$ such that their collective encoding is close to $\tilde{\Pi}_*$ and that, for every $j \in [\mathsf{k}]$, $\tilde{\pi}_j$ is close to the encoding of $\tilde{\mathbb{i}}[j]$. As before, since the proximity parameter of the PCPP is smaller than the distance of the code, this implies that $\tilde{\Pi}_*$ decodes to $(\tilde{\mathbb{i}}[1], \ldots, \tilde{\mathbb{i}}[\mathsf{k}], \tilde{\mathbb{w}})$ and that $\tilde{\pi}_j$ decodes to $\tilde{\mathbb{i}}[j]$. Suppose towards contradiction that for some $j \in [\mathsf{k}]$ the above condition does not hold: for every $\widehat{\pi}_j$ and $\widehat{\Pi}$ such that $\widehat{\pi}_j$ is close to $\tilde{\pi}_j$ and $\widehat{\Pi}_*$ is close to $\tilde{\Pi}_*$ it holds that $M_j(\widehat{\pi}_j, \widehat{\Pi}_*) = 0$.

By the soundness of the (*multi-input*) PCPP, the PCPP verifier must reject with high probability, which contradicts our assumption that $\mathbf{V}_{\mathsf{PCP}}$ (which runs the PCPP verifier) *accepts* with high probability.

Complexity measures. The above construction is an index-decodable PCP with polynomial-length proofs and where the verifier makes $O(1)$ queries to each indexer proof and makes $O(1)$ queries to the prover proof. Moreover, the prover proof has alphabet size 2^{k} since the prover bundles the PCPP consistency test proofs into k-bit symbols; this bundling is possible because the verifier shares randomness between all of the (oblivious) PCPPs in the consistency test. Since the PCPPs are oblivious to the index i, and they share randomness, they all must make the same queries to their oracles. The verifier uses $O(\log |\mathbb{x}|)$ bits of randomness: $O(\log |\mathbb{x}|)$ for the membership test, and $O(\log |\mathbb{x}|)$ for all k consistency tests combined.

Achieving constant query complexity over a binary alphabet We describe how to achieve an index-decodable PCP with constant query complexity per proof over the binary alphabet. The main tool is proof composition. In order to apply proof composition, we define and construct a robust variant of index-decodable PCPs.

Proof composition. Proof composition is a technique to lower the query complexity of PCPs [2] and IOPs [9]. In proof composition, an "inner" PCP is used to prove that a random execution of the "outer" PCP would have accepted. The inner PCP needs to be a PCPP, which is a PCP system where the verifier has oracle access to its input in addition to the prover's proof, and the soundness guarantee is that if the input is *far* from any input in the language, then the verifier accepts with small probability. To match this, the outer PCP must be *robust*, which means that the soundness guarantee ensures that when the instance is not in the language then not only is a random local view of the verifier rejecting but it is also far (in Hamming distance) from any accepting local view.

Typically the robust outer PCP has small proof length but large query complexity, while the inner PCPP has small query complexity but possibly a large proof length. Composition yields a PCP with small query complexity and small proof length.

We observe that proof composition *preserves decodability*: if the outer PCP in the composition is index-decodable, then the composed PCP is index-decodable. This is because the composition operation does not change the outer PCP proof and only adds a verification layer to show that the outer verifier accepts.

We thus apply proof composition as follows: the outer PCP is a robust variant of the index-decodable PCP from Section 2.5; and the inner PCP is a standard PCPP with polynomial proof length. This will complete the proof sketch of Theorem 4.

Defining robust index-decodable PCPs. Our goal is to perform proof composition where the outer PCP is index-decodable. As mentioned above, this requires the

PCP to be robust. Our starting point is the index-decodable PCP from Section 2.5. This PCP does have large query complexity over the binary alphabet ($O(\mathsf{k})$ queries to the prover proof). However, the fact that its queries to the prover proof are already bundled into a constant number of locations over an alphabet of size 2^{k} implies that we do not have to worry about a "generic" query bundling step and instead only have to perform a (tailored) robustification step prior to composition. Accordingly, the robustness definition below focuses on the prover proof, and so is the corresponding construction described after.

Definition 2. *A non-adaptive*[11] *index-decodable PCP* $(\mathbf{I}_{\mathsf{PCP}}, \mathbf{P}_{\mathsf{PCP}}, (\mathbf{V}^{\mathsf{qry}}_{\mathsf{PCP}}, \mathbf{V}^{\mathsf{dc}}_{\mathsf{PCP}}), \mathbf{iD}_{\mathsf{PCP}}, \mathbf{wD}_{\mathsf{PCP}})$ *for a multi-indexed relation* R *is* **prover-robustly index-decodable** *with decodability bound* κ_{PCP} *and robustness* σ_{PCP} *if for every* $\mathbb{x}$ *and proofs* $\tilde{\mathbf{\Pi}}_{\mathsf{i}} = (\tilde{\pi}_1, \ldots, \tilde{\pi}_{\mathsf{k}})$ *and* $\tilde{\Pi}$ *if*

$$\Pr_{\rho}\left[\exists\, A' \text{ s.t. } \mathbf{V}^{\mathsf{dc}}_{\mathsf{PCP}}(\mathbb{x}, \rho, \tilde{\mathbf{\Pi}}_{\mathsf{i}}[Q_{\mathsf{i}}], A') = 1 \ \wedge\ \Delta(A', A) \le \sigma_{\mathsf{PCP}}(|\mathbb{x}|) \,\middle|\, \begin{array}{l}(Q_{\mathsf{i}}, Q_*) \leftarrow \mathbf{V}^{\mathsf{qry}}_{\mathsf{PCP}}(\mathbb{x}, \rho) \\ A := \{\ \tilde{\Pi}[q] \mid q \in Q_*\ \}\end{array}\right] > \kappa_{\mathsf{PCP}}(|\mathbb{x}|)$$

then $\left(\mathbf{iD}_{\mathsf{PCP}}(\tilde{\pi}_1), \ldots, \mathbf{iD}_{\mathsf{PCP}}(\tilde{\pi}_{\mathsf{k}}), \mathbb{x}, \mathbf{wD}_{\mathsf{PCP}}(\tilde{\Pi})\right) \in R$. *Above* Q_{i} *and* Q_* *are the queries made to the indexer proofs the prover proof respectively and* $\Delta(A', A)$ *is the relative distance between* A' *and* A.

In other words, if $\left(\mathbf{iD}_{\mathsf{PCP}}(\tilde{\pi}_1), \ldots, \mathbf{iD}_{\mathsf{PCP}}(\tilde{\pi}_{\mathsf{k}}), \mathbb{x}, \mathbf{wD}_{\mathsf{PCP}}(\tilde{\Pi})\right) \notin R$ then with high probability not only will the verifier reject but also any set of answers *from the prover proof* that are close in Hamming distance to the real set of answers will also be rejecting.

Robustification. We outline how we transform the index-decodable PCP constructed in Section 2.5 into a *robust* index-decodable PCP. The techniques follow the robustification step in [15]. The transformation preserves the verifier's randomness complexity $O(\log |\mathbb{x}|)$, which facilitates using this modified PCP as the outer PCP in proof composition.

We apply an error-correcting code separately to each symbol of the prover proof. When the verifier wants to read a symbol from this proof, it reads the codeword encoding the symbol, decodes it, and then continues. It reads the indexer proofs as in the original PCP. This makes the PCP robust because if a few bits of the codeword representing a symbol are corrupted, then it will still be decoded to the same value. The robustness, however, degrades with the number of queries. If the relative distance of the error-correcting code is δ and the original verifier reads q symbols from the prover proof, then the resulting PCP will have robustness $O(\delta/q)$.

Indeed, let $c_1, \ldots, c_q$ be the codewords read by the new PCP verifier from the prover proof, and let $a_1, \ldots, a_q$ be such that a_i is the decoding of c_i. In order to change the decoding into some other set of strings $a'_1, \ldots, a'_q$ that, when received

[11] A PCP verifier is non-adaptive if it can be split into two algorithms: $\mathbf{V}^{\mathsf{qry}}_{\mathsf{PCP}}$ chooses which locations to query without accessing its oracles; and $\mathbf{V}^{\mathsf{dc}}_{\mathsf{PCP}}$ receives the results of the queries and decides whether to accept or reject.

by the verifier, may induce a different decision than $a_1, \ldots, a_q$, it suffices (in the worst case) to change a single codeword to decode to a different value. Since the relative distance of the code is δ, to do this, one must change at least a δ-fraction of the bits of a single codeword, c_i. A δ-fraction of a single codeword is a δ/q-fraction of the whole string of q codewords, $c_1, \ldots, c_q$.

In sum, to achieve constant robustness, *we need to begin with an index-decodable PCP with a small number of queries to the prover proof*, but possibly with a large alphabet. It is for this reason that we required this property in Section 2.5.

2.6 Commit-and prove SNARKs from index-decodable PCPs

We outline the proof of Theorem 3 by showing how to generically transform an index-decodable PCP into a commit-and-prove SNARK. First, we review the Micali transformation from PCPs to SNARGs. Then, we define commit-and-prove SNARKs and explain the challenges in constructing them. Finally, we outline how we overcome these challenges in our construction.

Review: the Micali construction. The SNARG prover uses the random oracle to Merkle hash the (long) PCP string into a (short) Merkle root that acts as a commitment; then, the SNARG prover uses the random oracle to derive randomness for the PCP verifier's queries; finally, the SNARG prover outputs an argument string that includes the Merkle root, answers to the PCP verifier's queries, and Merkle authentication paths for each of those answers (acting as local openings to the commitment). The SNARG verifier re-derives the PCP verifier's queries from the Merkle root and runs the PCP verifier with the provided answers, ensuring that each answer is authenticated.

The security analysis roughly works as follows. Fix a malicious prover that makes at most t queries to the random oracle and convinces the SNARG verifier to accept with probability δ. First, one argues that the malicious prover does not find any collisions or inversions for the random oracle except with probability $\mu := O(\frac{t^2}{2^\lambda})$. Next, one argues that there is an algorithm that, given the malicious prover, finds a PCP string that makes the PCP verifier accept with probability at least $\frac{1}{t} \cdot (\delta - \mu)$. This enables to establish soundness of the SNARG (if the PCP has soundness error β_{PCP} then for instances not in the language it must be that $\frac{1}{t} \cdot (\delta - \mu) \leq \beta_{\mathsf{PCP}}$ and thus that the SNARG has soundness error $t \cdot \beta_{\mathsf{PCP}} + \mu$) and also to establish knowledge soundness of the SNARG (if the PCP has knowledge error κ_{PCP} then the PCP extractor works provided that $\frac{1}{t} \cdot (\delta - \mu) \geq \kappa_{\mathsf{PCP}}$ and thus the SNARG is a SNARK with knowledge error $t \cdot \kappa_{\mathsf{PCP}} + \mu$).

The aforementioned algorithm that finds the PCP string is known as *Valiant's extractor* (it was used implicitly in [55] and formally defined and analyzed in [12]). Given the query/answer transcript of the malicious prover to the random oracle, Valiant's extractor finds the partial PCP string that the malicious prover "had in mind" when producing the SNARG: any location that the malicious prover could open is part of the partial PCP string (and has a unique value as we conditioned on the prover finding no collisions); conversely, any location that is

not part of the partial PCP string is one for which the malicious prover could not generate a valid local opening. Crucially, the malicious prover, in order to cause the SNARG verifier to accept, must generate randomness by applying the random oracle to the Merkle root, and answering the corresponding PCP queries with authenticated answers. Hence the partial PCP string output by Valiant's extractor causes the PCP verifier to accept with the same probability as the malicious prover, up to (i) the additive loss μ due to conditioning on no inversions and collisions, and (ii) the multiplicative loss of t due to the fact that the malicious prover can generate up to t different options of randomness for the PCP verifier and then choose among them which to use for the output SNARG. Overall, while Valiant's extractor *cannot* generate an entire PCP string, it finds "enough" of a PCP string to mimic the malicious prover, and so the PCP string's undefined locations can be set arbitrarily.

Commit-and-prove SNARK. A CaP-SNARK (in the ROM) for an indexed relation $R = \{(\mathbb{i}, \mathbb{x}, \mathbb{w})\}$ is a tuple $\mathsf{ARG} = (\mathbf{C}, \mathbf{P}, \mathbf{V})$ of deterministic polynomial-time oracle machines, where $\mathbf{C} = (\mathsf{Com}, \mathsf{Check})$ is a succinct commitment scheme,[12] that works as follows. The committer sends a short commitment $\mathsf{cm} := \mathbf{C}.\mathsf{Com}(\mathbb{i})$ to the verifier. Subsequently, the prover sends a short proof $\mathsf{pf} := \mathbf{P}(\mathbb{i}, \mathbb{x}, \mathbb{w})$ attesting that it knows a witness $\mathbb{w}$ such that $(\mathbb{i}, \mathbb{x}, \mathbb{w}) \in R$ and $\mathbb{i}$ is the index committed in cm. The verifier $\mathbf{V}$ receives $(\mathsf{cm}, \mathbb{x}, \mathsf{pf})$ and decides whether to accept the prover's claim. Completeness states that, if all parties act honestly, the verifier always accepts.

The security requirement of a CaP-SNARK is *(straight-line) knowledge soundness*. Informally, knowledge soundness says that if a query-bounded malicious prover convinces the verifier to accept the tuple $(\mathsf{cm}, \mathbb{x}, \mathsf{pf})$ with large enough probability, then the prover "knows" an index $\mathbb{i}$ opening cm and a witness $\mathbb{w}$ such that $(\mathbb{i}, \mathbb{x}, \mathbb{w}) \in R$. In more detail, we say that $\mathsf{ARG} = (\mathbf{C}, \mathbf{P}, \mathbf{V})$ has knowledge error ϵ if there exists a deterministic polynomial-time machine $\mathbf{E}$ (the extractor) such that for every $\lambda \in \mathbb{N}$, $n \in \mathbb{N}$, and deterministic t-query (malicious) prover $\tilde{\mathbf{P}}$,

$$\Pr\left[\begin{array}{c} \mathbf{V}^{\zeta}(\mathsf{cm}, \mathbb{x}, \mathsf{pf}) = 1 \ \wedge \ |\mathbb{x}| = n \ \wedge \\ \Big((\mathbb{i}, \mathbb{x}, \mathbb{w}) \notin R \ \vee \ \mathbf{C}.\mathsf{Check}^{\zeta}(\mathsf{cm}, \mathbb{i}, \mathsf{op}) = 0\Big) \end{array} \middle| \begin{array}{r} \zeta \leftarrow \mathcal{U}(\lambda) \\ (\mathsf{cm}, \mathbb{x}, \mathsf{pf}; \mathsf{tr}) := \tilde{\mathbf{P}}^{\zeta} \\ (\mathbb{i}, \mathsf{op}, \mathbb{w}) := \mathbf{E}(\mathsf{cm}, \mathbb{x}, \mathsf{pf}, \mathsf{tr}) \end{array}\right] \leq \epsilon(\lambda, n, t) \ ,$$

where $\mathcal{U}(\lambda)$ is the uniform distribution over functions $\zeta \colon \{0,1\}^* \to \{0,1\}^{\lambda}$ and $\mathsf{tr} := (j_1, a_1, \ldots, j_t, a_t)$ are the query/answer pairs made by $\tilde{\mathbf{P}}$ to its oracle.

First construction attempt. At first glance, constructing CaP-SNARKs using index-decodable PCPs seems like a straightforward variation of Micali's construction of SNARGs from PCPs.

[12] A pair of algorithms $\mathbf{C} = (\mathsf{Com}, \mathsf{Check})$ is a succinct commitment scheme if: (1) it is hard for every query-bounded adversary to find two different messages that pass verification for the same commitment string; and (2) the commitment of a message of length n with security parameter λ has length $\mathrm{poly}(\lambda, \log n)$.

- $\mathbf{C}.\mathsf{Com}$: Apply the ID-PCP indexer $\mathbf{I}_{\mathsf{PCP}}$ to the index $\mathbb{i}$ and output the Merkle root $\mathsf{rt}_{\mathbb{i}}$ of its output.
- $\mathbf{C}.\mathsf{Check}$: Given a Merkle root $\mathsf{rt}_{\mathbb{i}}$ and index $\mathbb{i}$, check that $\mathbf{C}.\mathsf{Com}(\mathbb{i}) = \mathsf{rt}_{\mathbb{i}}$.
- $\mathbf{P}$: Compute the ID-PCP prover proof and a corresponding Merkle root; then use the random oracle to derive randomness for the ID-PCP verifier's queries; finally, output an argument string pf that includes the Merkle root, answers to the verifier's queries, and authentication paths for each answer relative to the appropriate Merkle root (for the indexer proof or for the prover proof).
- $\mathbf{V}$: Re-derive the ID-PCP verifier's queries from the Merkle root and run the ID-PCP verifier with the provided answers, ensuring that each answer is authenticated.

The main issue with this strawman construction is that we need to handle malicious provers that have a *partial tree* in their query trace. Consider a malicious prover that, for some $(\mathbb{i}, \mathbb{x}, \mathbb{w}) \in R$, computes honestly the indexer proof for $\mathbb{i}$ as $\pi := \mathbf{I}_{\mathsf{PCP}}(\mathbb{i})$ and then generates as its "commitment" a Merkle tree root $\mathsf{rt}_{\mathbb{i}}$ obtained by computing a partial Merkle tree that ignores a small number of locations of π (i.e., for a small number of locations it begins deriving the tree from a level other than the leaves). While this malicious prover cannot open this small number of locations of π, it can still open all other locations of π. Next, the malicious prover generates honestly an argument string pf, opening the required locations of π from $\mathsf{rt}_{\mathbb{i}}$. This malicious prover makes the argument verifier accept (w.h.p.) since the ID-PCP verifier queries the small subset of locations that the prover cannot open with small probability.

However, the only way to find a string π' that (honestly) hashes to rt is to find inversions in the random oracle, which is infeasible. Thus, there is no efficient extractor that, given rt and all of the queries that the prover made, outputs $\mathbb{i}'$ whose indexer proof hashes to rt.

Solving the problem via proximity. We solve the above difficulty by modifying the commitment scheme $\mathbf{C} = (\mathsf{Com}, \mathsf{Check})$ and requiring more properties from the underlying index-decodable PCP.

- $\mathbf{C}.\mathsf{Com}$: Compute $\pi := \mathbf{I}_{\mathsf{PCP}}(\mathbb{i})$ (apply the ID-PCP indexer to the index $\mathbb{i}$) and output the commitment $\mathsf{cm} := \mathsf{rt}_{\mathbb{i}}$ that equals the Merkle hash of π and output the opening information op that consists of the list of authentication paths for each entry in π.
- $\mathbf{C}.\mathsf{Check}$: Given a commitment $\mathsf{cm} = \mathsf{rt}_{\mathbb{i}}$, index $\mathbb{i}$, and opening information op, check that op is a list of valid authentication paths for a number of entries that is above a certain threshold, and that the partial string specified by them decodes into $\mathbb{i}$ (when setting the unspecified values arbitrarily).

Now $\mathbf{C}.\mathsf{Check}$ allows *partial* specification of the string under the Merkle root, so to preserve the binding property of the commitment scheme we require that $(\mathbf{I}_{\mathsf{PCP}}, \mathbf{iD}_{\mathsf{PCP}})$ is an error correcting code. *The threshold of the number of authentication paths required is related to the distance of this code.*

In the security analysis, Valiant's extractor finds a partial PCP string that makes the ID-PCP verifier accept with probability related to the SNARG prover's convincing probability, as well as authentication paths for each entry of that partial PCP string. To ensure that the number of authenticated entries is large enough to pass the threshold in $\mathbf{C}.\mathsf{Check}$, we add another requirement: if $\tilde{\pi}$ and $\tilde{\Pi}$ make the ID-PCP verifier accept an instance $\mathbb{x}$ with probability larger than the decodability bound then $\tilde{\pi}$ is close to a codeword of the code $(\mathbf{I}_{\mathsf{PCP}}, \mathbf{iD}_{\mathsf{PCP}})$ (in addition to the fact that the decodings of $\tilde{\pi}$ and $\tilde{\Pi}$ put $\mathbb{x}$ in the relation as is the case in the definition considered so far).

Our construction of index-decodable PCP supports these new requirements.

From an index-decodable PCP to a CaP-SNARK. Let $(\mathbf{I}_{\mathsf{PCP}}, \mathbf{P}_{\mathsf{PCP}}, \mathbf{V}_{\mathsf{PCP}}, \mathbf{iD}_{\mathsf{PCP}}, \mathbf{wD}_{\mathsf{PCP}})$ be an index-decodable PCP system where $(\mathbf{I}_{\mathsf{PCP}}, \mathbf{iD}_{\mathsf{PCP}})$ is an error correcting code with relative distance δ and where indexer proofs are guaranteed to be $\delta/8$-close to valid codewords (when $\mathbf{V}_{\mathsf{PCP}}$ accepts above the decodability bound). We construct a CaP-SNARK $\mathsf{ARG} = (\mathbf{C}, \mathbf{P}, \mathbf{V})$ as follows.

- $\mathbf{C}.\mathsf{Com}$: Given as input an index $\mathbb{i}$, compute the indexer proof $\pi := \mathbf{I}_{\mathsf{PCP}}(\mathbb{i})$, compute the Merkle root $\mathsf{rt}_{\mathbb{i}}$ of a Merkle tree on π (using the random oracle as the hash function), and output the commitment $\mathsf{cm} := \mathsf{rt}_{\mathbb{i}}$ and the opening op containing all authentication paths.
- $\mathbf{C}.\mathsf{Check}$: Given as input a commitment $\mathsf{cm} := \mathsf{rt}_{\mathbb{i}}$, an index $\mathbb{i}$, and an opening op containing authentication paths, do the following:
  - check that op contains a list of authenticated entries relative to the Merkle root $\mathsf{rt}_{\mathbb{i}}$;
  - check that op represents at least a $(1 - \frac{\delta}{8})$-fraction of all possible entries for a string under $\mathsf{rt}_{\mathbb{i}}$;
  - let π be the string induced by the authenticated entries in op, setting arbitrarily other entries;
  - check that $\mathbf{I}_{\mathsf{PCP}}(\mathbb{i})$ is $\delta/4$-close to π.
- $\mathbf{P}$: Given as input $(\mathbb{i}, \mathbb{x}, \mathbb{w})$, do the following:
  - compute the commitment $\mathsf{rt}_{\mathbb{i}}$ to the index $\mathbb{i}$ as the committer does;
  - compute the PCP string $\Pi := \mathbf{P}_{\mathsf{PCP}}(\mathbb{i}, \mathbb{x}, \mathbb{w})$;
  - compute the Merkle root $\mathsf{rt}_{\mathbb{w}}$ of a Merkle tree on Π;
  - apply the random oracle to the string $(\mathsf{rt}_{\mathbb{i}} || \mathbb{x} || \mathsf{rt}_{\mathbb{w}})$ to derive randomness for the index-decodable PCP verifier $\mathbf{V}_{\mathsf{PCP}}$, and compute the answers to the verifier's queries to both π and Π;
  - collect authentication paths from the Merkle trees for each answer; and
  - output a proof pf containing $\mathsf{rt}_{\mathbb{w}}$, query answers for π and Π and their authentication paths.
- $\mathbf{V}$: Check the authentication paths, re-derive randomness, and run the index-decodable PCP verifier with this randomness and given these answers.

The tuple $\mathbf{C} = (\mathsf{Com}, \mathsf{Check})$ is a binding commitment scheme, as we now explain. Consider an attacker that outputs $\mathsf{cm} := \mathsf{rt}$, two distinct messages $\mathbb{m}, \mathbb{m}'$, and two openings $\mathsf{op} := S$ and $\mathsf{op}' := S'$ such that $\mathbf{C}.\mathsf{Check}^{\zeta}(\tilde{\mathsf{cm}}, \mathbb{m}, \mathsf{op}) = 1$ and $\mathbf{C}.\mathsf{Check}^{\zeta}(\tilde{\mathsf{cm}}, \mathbb{m}', \mathsf{op}') = 1$. Condition on the attacker not finding collisions or

inversions of the random oracle ζ (as this is true with high probability). Since S and S' each pass the checks in $\mathbf{C}.\mathsf{Check}$, each set covers at least $(1-\delta/8)$ of the possible openings for a string. Therefore, their intersection covers at least $(1-\delta/4)$ of the possible openings. Since there are no collisions or inversions, S and S' agree on all of these locations. Thus, letting π and π' be the strings defined using S and S' respectively (as computed by $\mathbf{C}.\mathsf{Check}$), we have that $\Delta(\pi,\pi') \leq \delta/4$. Additionally, we have that that $\Delta(\mathbf{I}_{\mathsf{PCP}}(\mathbb{m}),\pi) \leq \delta/4$ and $\Delta(\mathbf{I}_{\mathsf{PCP}}(\mathbb{m}'),\pi') \leq \delta/4$ since $\mathbf{C}.\mathsf{Check}$ accepts the commitments to $\mathbb{m}$ and $\mathbb{m}'$, and this is one of the checks it does. Putting all of this together, we have that $\Delta(\mathbf{I}_{\mathsf{PCP}}(\mathbb{m}),\mathbf{I}_{\mathsf{PCP}}(\mathbb{m}')) \leq \delta/2$ which implies that $\mathbb{m} = \mathbb{m}'$ since $\delta/2$ is the unique decoding distance.

Completeness of the CaP-SNARK is straightforward. Below we outline the extractor $\mathbf{E}$, which receives as input a commitment $\mathsf{cm} := \mathsf{rt}_{\mathbb{i}}$, an argument string pf (containing the commitment $\mathsf{rt}_{\mathbb{w}}$, query answers for π and Π, and corresponding authentication paths with respect to $\mathsf{rt}_{\mathbb{i}}$ and $\mathsf{rt}_{\mathbb{w}}$), and the list tr of query/answer pairs made by the malicious prover $\tilde{\mathbf{P}}$ to the random oracle.

> Use Valiant's extractor to compute the set $S_{\mathbb{i}}$ of all valid local openings of $\mathsf{rt}_{\mathbb{i}}$ that the prover could generate and similarly extract $S_{\mathbb{w}}$ from $\mathsf{rt}_{\mathbb{w}}$. Let $\tilde{\pi}$ be the string whose entries are defined by the local openings generated of $\mathsf{rt}_{\mathbb{i}}$ (and whose undefined entries are set arbitrarily to 0). Let $\tilde{\Pi}$ be defined similarly from the openings of $\mathsf{rt}_{\mathbb{w}}$. Compute the index $\mathbb{i} := \mathbf{iD}_{\mathsf{PCP}}(\tilde{\pi})$ and the witness $\mathbb{w} := \mathbf{wD}_{\mathsf{PCP}}(\tilde{\Pi})$, and set $\mathsf{op} := S_{\mathbb{i}}$. Output $(\mathbb{i},\mathsf{op},\mathbb{w})$.

We show the following lemma. See the full version of this paper for a proof.

Lemma 1. *Let κ_{PCP} be the decodability bound of the index-decodable PCP, $t \in \mathbb{N}$ be a bound on the number of queries made by a malicious prover $\tilde{\mathbf{P}}$, and $\lambda \in \mathbb{N}$ be a security parameter. Then the knowledge extractor $\mathbf{E}$ above has knowledge error $t \cdot \kappa_{\mathsf{PCP}} + O(\frac{t^2}{2^\lambda})$.*

2.7 Hardness of approximation

We outline our proof of Theorem 2 (it is AM[k]-complete to decide if an instance of k-SSAT has value 1 or at most $1 - \frac{1}{O(\mathsf{k})}$). See Section 1.1 for definitions.

First we explain how an AM[k] protocol can distinguish whether a k-SSAT instance has value 1 or value $1 - \frac{1}{O(\mathsf{k})}$. On input a k-SSAT instance ϕ, the prover and verifier take turns giving values to the variables: the verifier sends random bits $\rho_{1,1},\ldots,\rho_{1,\ell}$, the prover answers with $a_{1,1},\ldots,a_{1,\ell}$, the verifier sends $\rho_{2,1},\ldots,\rho_{2,\ell}$, and so on until all of the variables of ϕ are given values. The verifier then accepts if and only if all of the clauses of ϕ are satisfied. For completeness, if ϕ has value 1, then for any choice of verifier messages there exists some strategy for the prover that will make the verifier accept. For soundness, when the value of ϕ is at most $1 - \frac{1}{O(\mathsf{k})}$, no matter what strategy the prover uses, the probability that the verifier accepts is at most $1 - \frac{1}{O(\mathsf{k})}$ (which can be made constant using parallel repetition).

Next we show that, for every language $L \in \mathrm{AM}[\mathsf{k}]$, a given instance $\mathbb{x}$ can be reduced in deterministic polynomial time to a k-SSAT formula ϕ such that:

– if $\mathbb{x} \in L$ then the value of ϕ is 1;
– if $\mathbb{x} \notin L$ then the value of ϕ is at most $1 - \frac{1}{O(\mathsf{k})}$.

By Theorem 1, L has a k-round public-coin IOP with a non-adaptive verifier, polynomial proof length, and logarithmic decision randomness where the IOP verifier reads $\mathsf{q} = O(\mathsf{k})$ bits of its interaction with the IOP prover. We stress that in the following proof it is crucial that Theorem 1 achieves an IOP with both logarithmic decision randomness complexity and small query complexity to both the prover and verifier messages.

Let $\mathbf{V}_{\rho_{\mathsf{dc}}}$ be the circuit that computes the decision bit of the verifier given as input the q answers to the q queries made by the IOP verifier, for the instance $\mathbb{x}$ and decision randomness ρ_{dc}. By carefully following the proof of Theorem 1, we know that the IOP verifier's decision is the conjunction of $O(\mathsf{k})$ computations, each of which takes $O(1)$ bits as input. Therefore $\mathsf{d} := |\mathbf{V}_{\rho_{\mathsf{dc}}}| = O(\mathsf{k})$.

Via the Cook–Levin theorem we efficiently transform $\mathbf{V}_{\rho_{\mathsf{dc}}}$ into a 3CNF formula $\phi_{\rho_{\mathsf{dc}}} \colon \{0,1\}^{\mathsf{q}+O(\mathsf{d})} \to \{0,1\}$ of size $O(\mathsf{d})$ the satisfies the following for every $b_1, \ldots, b_{\mathsf{q}} \in \{0,1\}$:
– if $\mathbf{V}_{\rho_{\mathsf{dc}}}(b_1, \ldots, b_{\mathsf{q}}) = 1$ then $\exists\, \mathbb{z}_1, \ldots, \mathbb{z}_{O(\mathsf{d})} \in \{0,1\}$ $\phi_{\rho_{\mathsf{dc}}}(b_1, \ldots, b_{\mathsf{q}}, \mathbb{z}_1, \ldots, \mathbb{z}_{O(\mathsf{d})}) = 1$;
– if $\mathbf{V}_{\rho_{\mathsf{dc}}}(b_1, \ldots, b_{\mathsf{q}}) = 0$ then $\forall\, \mathbb{z}_1, \ldots, \mathbb{z}_{O(\mathsf{d})} \in \{0,1\}$ $\phi_{\rho_{\mathsf{dc}}}(b_1, \ldots, b_{\mathsf{q}}, \mathbb{z}_1, \ldots, \mathbb{z}_{O(\mathsf{d})}) = 0$.

Next we describe the k-SSAT instance ϕ. The variables of ϕ correspond to messages in the IOP as follows. For each $i \in [\mathsf{k}]$, the random variables $\rho_{i,1}, \ldots, \rho_{i,\mathsf{r}}$ represent the verifier's message in round i and the existential variables $a_{i,1}, \ldots, a_{i,\mathsf{l}}$ represent the prover's message in round i. To the final set of existential variables we add additional variables $\mathbb{z}_{\rho_{\mathsf{dc}},1} \ldots, \mathbb{z}_{\rho_{\mathsf{dc}},O(\mathsf{d})}$ for every $\rho_{\mathsf{dc}} \in \{0,1\}^{O(\log |\mathbb{x}|)}$, matching the additional variables added when reducing the boolean circuit $\mathbf{V}_{\rho_{\mathsf{dc}}}$ to the boolean formula $\phi_{\rho_{\mathsf{dc}}}$. The k-SSAT instance ϕ is the conjunction of the formulas $\phi_{\rho_{\mathsf{dc}}}$ for every $\rho_{\mathsf{dc}} \in \{0,1\}^{O(\log |\mathbb{x}|)}$ where each $\phi_{\rho_{\mathsf{dc}}}$ has as its variables the variables matching the locations in the IOP transcript that the IOP verifier queries given $\mathbb{x}$ and ρ_{dc}, and additionally the variables added by converting $\mathbf{V}_{\rho_{\mathsf{dc}}}$ into a formula, $\mathbb{z}_{\rho_{\mathsf{dc}},1} \ldots, \mathbb{z}_{\rho_{\mathsf{dc}},O(\mathsf{d})}$.

By perfect completeness of the IOP, if $\mathbb{x} \in L$ then there is a prover strategy such that, no matter what randomness is chosen by the verifier, every $\mathbf{V}_{\rho_{\mathsf{dc}}}$ is simultaneously satisfied, and hence so are the formulas $\phi_{\rho_{\mathsf{dc}}}$, implying that the value of ϕ is 1.

By soundness of the IOP, if $\mathbb{x} \notin L$ then (in expectation) at most a constant fraction of the circuits $\{\mathbf{V}_{\rho_{\mathsf{dc}}}\}_{\rho_{\mathsf{dc}} \in \{0,1\}^{O(\log |\mathbb{x}|)}}$ are simultaneously satisfiable, and thus this is also true for the formulas $\{\phi_{\rho_{\mathsf{dc}}}\}_{\rho_{\mathsf{dc}} \in \{0,1\}^{O(\log |\mathbb{x}|)}}$. Every formula $\phi_{\rho_{\mathsf{dc}}}$ that is not satisfied has at least one of its $O(\mathsf{d})$ clauses not satisfied. Thus, the value of ϕ is at most $1 - \frac{1}{O(\mathsf{d})} = 1 - \frac{1}{O(\mathsf{k})}$.

References

1. Arora, S., Lund, C., Motwani, R., Sudan, M., Szegedy, M.: Proof verification and the hardness of approximation problems. Journal of the ACM **45**(3), 501–555 (1998)

2. Arora, S., Safra, S.: Probabilistic checking of proofs: a new characterization of NP. Journal of the ACM **45**(1), 70–122 (1998), preliminary version in FOCS '92.
3. Babai, L.: Trading group theory for randomness. In: Proceedings of the 17th Annual ACM Symposium on Theory of Computing. pp. 421–429. STOC '85 (1985)
4. Babai, L., Fortnow, L., Levin, L.A., Szegedy, M.: Checking computations in polylogarithmic time. In: Proceedings of the 23rd Annual ACM Symposium on Theory of Computing. pp. 21–32. STOC '91 (1991)
5. Babai, L., Moran, S.: Arthur-merlin games: A randomized proof system, and a hierarchy of complexity classes. Journal of Computer and System Sciences **36**(2), 254–276 (1988)
6. Bellare, M., Goldreich, O., Goldwasser, S.: Randomness in interactive proofs. In: Proceedings of the 31st Annual Symposium on Foundations of Computer Science. pp. 563–572. FOCS '90 (1990)
7. Ben-Sasson, E., Bentov, I., Horesh, Y., Riabzev, M.: Fast Reed–Solomon interactive oracle proofs of proximity. In: Proceedings of the 45th International Colloquium on Automata, Languages and Programming. pp. 14:1–14:17. ICALP '18 (2018)
8. Ben-Sasson, E., Bentov, I., Horesh, Y., Riabzev, M.: Scalable zero knowledge with no trusted setup. In: Proceedings of the 39th Annual International Cryptology Conference. pp. 733–764. CRYPTO '19 (2019)
9. Ben-Sasson, E., Chiesa, A., Gabizon, A., Riabzev, M., Spooner, N.: Interactive oracle proofs with constant rate and query complexity. In: Proceedings of the 44th International Colloquium on Automata, Languages and Programming. pp. 40:1–40:15. ICALP '17 (2017)
10. Ben-Sasson, E., Chiesa, A., Goldberg, L., Gur, T., Riabzev, M., Spooner, N.: Linear-size constant-query IOPs for delegating computation. In: Proceedings of the 17th Theory of Cryptography Conference. pp. 494–521. TCC '19 (2019)
11. Ben-Sasson, E., Chiesa, A., Riabzev, M., Spooner, N., Virza, M., Ward, N.P.: Aurora: Transparent succinct arguments for R1CS. In: Proceedings of the 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 103–128. EUROCRYPT '19 (2019)
12. Ben-Sasson, E., Chiesa, A., Spooner, N.: Interactive oracle proofs. In: Proceedings of the 14th Theory of Cryptography Conference. pp. 31–60. TCC '16-B (2016)
13. Ben-Sasson, E., Goldberg, L., Kopparty, S., Saraf, S.: DEEP-FRI: sampling outside the box improves soundness. In: Proceedings of the 11th Innovations in Theoretical Computer Science Conference. pp. 5:1–5:32. ITCS '20 (2020)
14. Ben-Sasson, E., Goldreich, O., Harsha, P., Sudan, M., Vadhan, S.: Short PCPs verifiable in polylogarithmic time. In: Proceedings of the 20th Annual IEEE Conference on Computational Complexity. pp. 120–134. CCC '05 (2005)
15. Ben-Sasson, E., Goldreich, O., Harsha, P., Sudan, M., Vadhan, S.P.: Robust PCPs of proximity, shorter PCPs, and applications to coding. SIAM Journal on Computing **36**(4), 889–974 (2006)
16. Ben-Sasson, E., Sudan, M.: Short PCPs with polylog query complexity. SIAM Journal on Computing **38**(2), 551–607 (2008)
17. Benarroch, D., Campanelli, M., Fiore, D., Kim, J., Lee, J., Oh, H., Querol, A.: Proposal: Commit-and-prove zero-knowledge proof systems and extensions. https://docs.zkproof.org/pages/standards/accepted-workshop4/proposal-commit.pdf (2021)
18. Bootle, J., Cerulli, A., Ghadafi, E., Groth, J., Hajiabadi, M., Jakobsen, S.K.: Linear-time zero-knowledge proofs for arithmetic circuit satisfiability. In: Proceedings of the 23rd International Conference on the Theory and Applications of Cryptology and Information Security. pp. 336–365. ASIACRYPT '17 (2017)

19. Bootle, J., Chiesa, A., Groth, J.: Linear-time arguments with sublinear verification from tensor codes. In: Proceedings of the 18th Theory of Cryptography Conference. pp. 19–46. TCC '20 (2020)
20. Bootle, J., Chiesa, A., Liu, S.: Zero-knowledge IOPs with linear-time prover and polylogarithmic-time verifier. Cryptology ePrint Archive, Report 2020/1527 (2020)
21. Bordage, S., Nardi, J.: Interactive oracle proofs of proximity to algebraic geometry codes. ArXiv cs/2011.04295 (2021)
22. Campanelli, M., Fiore, D., Querol, A.: LegoSNARK: Modular design and composition of succinct zero-knowledge proofs. In: Proceedings of the 26th Conference on Computer and Communications Security. pp. 2075–2092. CCS '19 (2019)
23. Canetti, R., Chen, Y., Holmgren, J., Lombardi, A., Rothblum, G.N., Rothblum, R.D.: Fiat–Shamir from simpler assumptions. Cryptology ePrint Archive, Report 2018/1004 (2018)
24. Chiesa, A., Hu, Y., Maller, M., Mishra, P., Vesely, N., Ward, N.: Marlin: Preprocessing zkSNARKs with universal and updatable SRS. In: Proceedings of the 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques. EUROCRYPT '20 (2020)
25. Chiesa, A., Manohar, P., Spooner, N.: Succinct arguments in the quantum random oracle model. In: Proceedings of the 17th Theory of Cryptography Conference. pp. 1–29. TCC '19 (2019)
26. Chiesa, A., Ojha, D., Spooner, N.: Fractal: Post-quantum and transparent recursive proofs from holography. In: Proceedings of the 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 769–793. EUROCRYPT '20 (2020)
27. Condon, A., Feigenbaum, J., Lund, C., Shor, P.W.: Probabilistically checkable debate systems and nonapproximability of PSPACE-hard functions. Chicago Journal of Theoretical Computer Science **1995** (1995)
28. Condon, A., Feigenbaum, J., Lund, C., Shor, P.W.: Random debaters and the hardness of approximating stochastic functions. SIAM Journal on Computing **26**(2), 369–400 (1997)
29. Costello, C., Fournet, C., Howell, J., Kohlweiss, M., Kreuter, B., Naehrig, M., Parno, B., Zahur, S.: Geppetto: Versatile verifiable computation. In: Proceedings of the 36th IEEE Symposium on Security and Privacy. pp. 250–273. S&P '15 (2015)
30. Dinur, I.: The PCP theorem by gap amplification. Journal of the ACM **54**(3), 12 (2007)
31. Dinur, I., Reingold, O.: Assignment testers: Towards a combinatorial proof of the PCP theorem. In: Proceedings of the 45th Annual IEEE Symposium on Foundations of Computer Science. pp. 155–164. FOCS '04 (2004)
32. Drucker, A.: Efficient probabilistically checkable debates. In: Proceedings of the 15th International Workshop on Approximation, Randomization, and Combinatorial Optimization. pp. 519–529. RANDOM '11 (2011)
33. Drucker, A.: A PCP characterization of AM. In: International Colloquium on Automata, Languages and Programming. pp. 581–592 (2011)
34. Drucker, A.: An improved exponential-time approximation algorithm for fully-alternating games against nature. In: Proceedings of the 61st Annual IEEE Symposium on Foundations of Computer Science. pp. 1081–1090. FOCS '20 (2020)
35. Escala, A., Groth, J.: Fine-tuning Groth–Sahai proofs. In: Proceedings of the 17th International Conference on Practice and Theory in Public Key Cryptography. pp. 630–649. PKC '14 (2014)

36. Feige, U., Goldwasser, S., Lovász, L., Safra, S., Szegedy, M.: Interactive proofs and the hardness of approximating cliques. Journal of the ACM **43**(2), 268–292 (1996), preliminary version in FOCS '91.
37. Fürer, M., Goldreich, O., Mansour, Y., Sipser, M., Zachos, S.: On completeness and soundness in interactive proof systems. Advances in Computing Research **5**, 429–442 (1989)
38. Goldreich, O., Micali, S., Wigderson, A.: Proofs that yield nothing but their validity or all languages in NP have zero-knowledge proof systems. Journal of the ACM **38**(3), 691–729 (1991), preliminary version appeared in FOCS '86.
39. Goldreich, O., Vadhan, S., Wigderson, A.: On interactive proofs with a laconic prover. Computational Complexity **11**(1/2), 1–53 (2002)
40. Goldwasser, S., Micali, S., Rackoff, C.: The knowledge complexity of interactive proof systems. SIAM Journal on Computing **18**(1), 186–208 (1989), preliminary version appeared in STOC '85.
41. Goldwasser, S., Sipser, M.: Private coins versus public coins in interactive proof systems. In: Proceedings of the 18th Annual ACM Symposium on Theory of Computing. pp. 59–68. STOC '86 (1986)
42. Guruswami, V., Umans, C., Vadhan, S.P.: Unbalanced expanders and randomness extractors from parvaresh–vardy codes. Journal of the ACM **56**(4), 20:1–20:34 (2009)
43. Haviv, I., Regev, O., Ta-Shma, A.: On the hardness of satisfiability with bounded occurrences in the polynomial-time hierarchy. Theory of Computing **3**(1), 45–60 (2007)
44. Ishai, Y., Weiss, M.: Probabilistically checkable proofs of proximity with zero-knowledge. In: Proceedings of the 11th Theory of Cryptography Conference. pp. 121–145. TCC '14 (2014)
45. Lee, N., Wang, Y., Jiang, J.R.: Solving stochastic boolean satisfiability under random-exist quantification. In: Proceedings of the 26th International Joint Conference on Artificial Intelligence. pp. 688–694. IJCAI 17 (2017)
46. Lipmaa, H.: Prover-efficient commit-and-prove zero-knowledge SNARKs. International Journal of Applied Cryptography **3**(4), 344–362 (2017)
47. Littman, M.L., Majercik, S.M., Pitassi, T.: Stochastic boolean satisfiability. Journal of Automated Reasoning **27**(3), 251–296 (2001)
48. Lund, C., Fortnow, L., Karloff, H.J., Nisan, N.: Algebraic methods for interactive proof systems. Journal of the ACM **39**(4), 859–868 (1992)
49. Majercik, S.M.: APPSSAT: Approximate probabilistic planning using stochastic satisfiability. International Journal of Approximate Reasoning **45**(2), 402–419 (2007)
50. Micali, S.: Computationally sound proofs. SIAM Journal on Computing **30**(4), 1253–1298 (2000), preliminary version appeared in FOCS '94.
51. Papadimitriou, C.H.: Games against nature (extended abstract). In: 24th Annual ACM Symposium on Theory of Computing. pp. 446–450. STOC '83 (1983)
52. Reingold, O., Rothblum, R., Rothblum, G.: Constant-round interactive proofs for delegating computation. In: Proceedings of the 48th ACM Symposium on the Theory of Computing. pp. 49–62. STOC '16 (2016)
53. Ron-Zewi, N., Rothblum, R.: Local proofs approaching the witness length. In: Proceedings of the 61st Annual IEEE Symposium on Foundations of Computer Science. pp. 846–857. FOCS '20 (2020)
54. Shamir, A.: IP = PSPACE. Journal of the ACM **39**(4), 869–877 (1992)
55. Valiant, P.: Incrementally verifiable computation or proofs of knowledge imply time/space efficiency. In: Theory of Cryptography Conference. pp. 1–18 (2008)