Locally Testable Codes and PCPs of Almost-Linear Length

Oded Goldreich*
Department of Computer Science
Weizmann Institute of Science
Rehovot, ISRAEL.
oded@wisdom.weizmann.ac.il

Madhu Sudan†
Laboratory for Computer Science
Massachusetts Institute of Technology
Cambridge, MA 02139.
madhu@mit.edu

August 5, 2002

Abstract

Locally testable codes are error-correcting codes that admit very efficient codeword tests. Specifically, using a constant number of (random) queries, non-codewords are rejected with probability proportional to their distance from the code.

Locally testable codes are believed to be the combinatorial core of PCPs. However, the relation is less immediate than commonly believed. Nevertheless, we show that certain PCP systems can be modified to yield locally testable codes. On the other hand, we adapt techniques we develop for the construction of the latter to yield new PCPs. Our main results are locally testable codes and PCPs of almost-linear length. Specifically, we present:

- Locally testable (linear) codes in which k information bits are encoded by a codeword of length approximately $k \cdot \exp(\sqrt{\log k})$. This improves over previous results that either yield codewords of exponential length or obtained almost quadratic length codewords for sufficiently large non-binary alphabet.
- PCP systems of almost-linear length for SAT. The length of the proof is approximately $n \cdot \exp(\sqrt{\log n})$ and verification in performed by a constant number (i.e., 19) of queries, as opposed to previous results that used proof length $n^{1+O(1/q)}$ for verification by q queries.

The novel techniques in use include a random projection of certain codewords and PCP-oracles, an adaptation of PCP constructions to obtain "linear PCP-oracles" for proving conjunctions of linear conditions, and a direct construction of locally testable (linear) codes of sub-exponential length.

Keywords: Error-Correcting Codes, PCP, randomized reductions, low-degree tests, codeword tests,

*Supported by the MINERVA Foundation, Germany.
†Supported in part by NSF Awards CCR 9875511, CCR 9912342, and MIT-NTT Award MIT 2001-04.

Contents

1 Introduction **2**
- 1.1 Relation to PCP . . . 2
- 1.2 Relation to Locally Decodable Codes . . . 3

2 Formal Setting **3**
- 2.1 Codes . . . 3
- 2.2 PCP . . . 4

3 Direct Constructions of Codes **5**
- 3.1 The Basic Code (FS-Code) . . . 5
- 3.2 Random Truncation of the FS-Code . . . 6
- 3.3 Decreasing the alphabet size . . . 8
- 3.4 A Binary Code . . . 10

4 Nearly linear-sized PCPs **11**
- 4.1 MIP verifiers and random sampling . . . 12
- 4.2 Improved 3-Prover Proof System for NP . . . 13
  - 4.2.1 Abtracting the verifier of Corollary A.4 . . . 14
  - 4.2.2 The 3-prover MIP: Stage I . . . 15
  - 4.2.3 The 3-prover MIP: Stage II . . . 15
  - 4.2.4 The 3-prover MIP: Stage III . . . 17
- 4.3 Nearly linear PCPs . . . 18

5 Nearly-linear-sized codes from PCPs **19**
- 5.1 Problems with using a PCP directly . . . 19
- 5.2 Inner verifiers for linear systems: Definition and composition . . . 20
- 5.3 Linear inner verifiers: Two constructions . . . 24
- 5.4 Combining all the constructions . . . 29
- 5.5 Additional remarks . . . 30

6 Conclusions and Open Problems **32**

Bibliography **33**

Appendices **35**

A The Gap Polynomial-Constraint-Satisfaction Problem **35**

1 Introduction

We study the existence of (good) error-correcting codes that admit very efficient codeword tests. Specifically, we require the testing procedure to use only a constant number of (random) queries, and reject non-codewords with probability proportional to their distance from the code. Such codes may be thought of as a combinatorial counterparts of the complexity theoretic notion of probabilistically checkable proofs (PCPs). They were formally introduced by Friedl and Sudan [11]. Here we initiate a systematic study of this notion.

Some examples: Codeword testing is meaningful only for good codes. In particular, it is easy to test trivial codes (e.g., for codes containing all possible strings of certain length or, on the other extreme, for codes containing a single codewords (or very few codewords)). One non-trivial code allowing efficient testing is the Hadamard code: the codewords are linear functions, and so codeword testing amounts to linearity testing [7].

The drawback of the Hadamard code is that k bits of information are encoded using a codeword of length 2^k. (The k information bits represent the k coefficients of a linear function $\{0,1\}^k \to \{0,1\}$, and bits in the codeword correspond to all possible evaluation points.)

The question addressed in this work is whether one can hope for a better relation between the number of information bits, k, and the length of the codeword, denoted n. Specifically, *can n be polynomial or even linear in k?* For (sufficiently large) *non-binary* alphabet, Friedl and Sudan [11] showed that n can be almost quadratic in k. We show that n may be *almost-linear* in k (i.e., $n = k^{1+o(1)}$), *even for the binary alphabet*.

1.1 Relation to PCP

It is a common belief, among PCP enthusiasts, that the PCP Theorem [1, 2] already provides codes as we desire. Consider the mapping of standard witnesses for, say SAT, to PCP-oracles. When applied to an instance of SAT that is a tautology, the map typically induces a good error-correcting code mapping k information bits to codewords of length $\text{poly}(k)$ (or almost linear in k, when using [17]). The common belief is that the PCP verifier also yields a codeword test. However, this is not quite true: It is only guaranteed that each passing oracle can be "decoded" to a corresponding NP-witness, but this does not mean that a passing oracle is (close to) a valid codeword (because the "decoding" procedure is actually stronger than is standard in coding theory), or that only codewords pass the test with probability one. For example, part of the PCP oracle is supposed to encode an m-variate polynomial of *individual degree* d, yet the PCP verifier will also accept the encoding of any m-variate polynomial of *total degree* $m \cdot d$ (and the "decoding" procedure will work in this case too).

Still, we show that many known PCP constructions can be modified to yield good codes with efficient codeword tests. We stress that these modifications are non-trivial and furthermore are unnatural in the context of PCP. Yet, they yield coding results of the type we seek (e.g., see Theorem 2.1).

On the other hand, a technique that emerges naturally in the context of our study of efficient codeword tests yields improved results on the length of efficient PCP proofs. Specifically, we obtain constant-query PCP systems that utilize oracles that are shorter than known before (see Theorem 2.3).

1.2 Relation to Locally Decodable Codes

The problem of designing efficient codeword tests *seems* easier than the question of designing efficient decoding procedures that allow to recover any desired information bit by reading only a constant number of bits in the codeword. Our results confirm this intuition:

- We show the existence of almost-linear (i.e., $n = k^{1+o(1)}$) length (binary) codes supporting codeword tests with a constant number of queries. In contrast, it was shown that locally decodable codes cannot have almost-linear length [16].[1]
- For large alphabet, we show almost-linear length coordinate-linear codes in which testing requires only *two* queries. In contrast, it was shown that coordinate-linear codes with *two* query recovery require exponential length [13].

2 Formal Setting

Throughout this work, all oracle machines (i.e., codeword testers and PCP verifiers) are non-adaptive; that is, they determine their queries based solely on their input and random choices. This is in contrast to adaptive orcale machines that may determine their queries based on answers obtained to prior queries. Since our focus is on positive results, this makes our results only stronger.

2.1 Codes

We consider codes mapping a sequence of k input symbols into a sequence of $n \geq k$ symbols over the same alphabet, denoted Σ, which may but need not be the binary alphabet. Such a generic code is denoted by $\mathcal{C} : \Sigma^k \to \Sigma^n$. Throughout this paper, *the integers k and n are to be thought of as parameters*, and Σ may depend on them. Thus, we actually discuss infinite families of codes (which are associated with infinite sets of possible k's), and whenever we say that some quantity of the code is a constant we mean that this quantity is constant for the entire family (of codes). Typically, we seek to have Σ as small as possible, desire that $|\Sigma|$ be a constant (i.e., does not depend on k), and are most content when $\Sigma = \{0,1\}$ (i.e., a binary code).

Distance between n-symbol sequences over Σ is defined in the natural manner; that is, for $u, v \in \Sigma^n$, the distance $\Delta(u,v)$ is defined as the number of locations on which u and v differ (i.e., $\Delta(u,v) \stackrel{\text{def}}{=} |\{i : u_i \neq v_i\}|$, where $u = u_1 \cdots u_n \in \Sigma^n$ and $v = v_1 \cdots v_n \in \Sigma^n$). The distance of a code $\mathcal{C} : \Sigma^k \to \Sigma^n$ is the minimum distance between its codewords; that is, $\min_{a \neq b}\{\Delta(\mathcal{C}(a), \mathcal{C}(b))\}$. *Throughout this work, we focus on codes of "large distance"*; specifically, codes $\mathcal{C} : \Sigma^k \to \Sigma^n$ of distance $\Omega(n)$.

The distance of $w \in \Sigma^n$ from a code $\mathcal{C} : \Sigma^k \to \Sigma^n$ is the minimum distance between w and the codewords; that is, $\min_a\{\Delta(w, \mathcal{C}(a))\}$. An interesting case is of non-codewords that are "relatively far from the code", which may mean that their distance from the code is greater than (say) half the distance of the code.

By a codeword test (for the code $\mathcal{C} : \Sigma^k \to \Sigma^n$) we mean a randomized (non-adaptive) oracle machine (called tester) that given oracle access to $w \in \Sigma^n$ (viewed as a function $w : \{1, ..., n\} \to \Sigma$) satisfies the following two conditions:[2]

[1] If q queries are used for recovery then $n = \Omega(k^{1+(1/(q-1))})$.

[2] Both the following conditions may be meaningfully relaxed. For example, the tester may be allowed to err with small probability in case it is given oracle access to a codeword, and the rejection condition may be restricted to non-codewords that are relatively far from the code. Since our results are positive, it make sense for us to use the stronger definition provided below.

- *Accepting codewords*: For any $a \in \Sigma^k$, given oracle access to $w = \mathcal{C}(a)$, the tester accepts with probability 1.

- *Rejection of non-codeword*: For every $w \in \Sigma^n$ that is at distance ϵn from C, given oracle access to $w = \mathcal{C}(a)$, the tester rejects with probability $\Omega(\epsilon) - o(1)$. (The $o(1)$ term can be avoided if we consider only non-codewords that are at distance more than $\epsilon_0 n$ from the code, for some constant $\epsilon_0 > 0$.)[3]

We say that the code $\mathcal{C} : \Sigma^k \to \Sigma^n$ is locally testable if it has a codeword test that makes a *constant number of queries*. Our main result regarding codes is

Theorem 2.1 *For every $c > 0.5$ and infinitely many k's, there exist locally testable codes with binary alphabet such that $n = \exp((\log k)^c) \cdot k = k^{1+o(1)}$. Furthermore, these codes are linear and have distance $\Omega(n)$.*

Theorem 2.1 (as well as Part 2 of Theorem 2.2) vastly improves over the Hadamard code (in which $n = 2^k$), which is the only locally testable *binary* code previously known. Theorem 2.1 is proven by combining Part 1 of the following Theorem 2.2 with non-standard modifications of standard PCP constructions.

Theorem 2.2 (proven by direct/self-contained constructions):

1. *For every $c > 0.5$ and infinitely many k's, there exist locally testable codes with non-binary alphabet Σ such that $n = \exp((\log k)^c) \cdot k = k^{1+o(1)}$ and $\log |\Sigma| = \exp((\log k)^c) = k^{o(1)}$. Furthermore, the tester makes two queries.*

2. *For every $c > 1$ and infinitely many k's, there exist locally testable codes binary alphabet such that $n < k^c$.*

In both cases, the codes are linear in a suitable sense and have distance $\Omega(n)$.

Part 1 improves over the work of Friedl and Sudan [11], which only yields $n = k^{2+o(1)}$. We comment that (good) *binary* codes cannot be tested using two queries (cf. [6]). The set of k's for which such codes exist is reasonable dense; in both cases, if k is in the set then the next integer in the set is smaller than $k^{1+o(1)}$. Specifically, in Part 1 (resp., Part 2), if k is in the set then the next integer in the set is smaller than $\exp((\log k)^{0.51}) \cdot k$ (resp., $O(\text{poly}(\log k) \cdot k)$).

2.2 PCP

A probabilistic checkable proof (PCP) system for a set L is a probabilistic polynomial-time (non-adaptive) oracle machine (called verifier), denoted V, satisfying

- *Completeness*: For every $x \in L$ there exists an oracle π_x so that V, on input x and access to oracle π_x, always accepts x.

- *Soundness*: For every $x \notin L$ and every oracle π, machine V, on input x and access to oracle π, rejects x with probability at least $\frac{1}{2}$.

[3] Following this alternative (i.e., of considering only non-codewords that are very far from the code), we may use an alternative formulation (which is more standard in the "property testing" literature; cf. [19, 12]). Specifically, we may require that every non-codeword that is at least $\epsilon_0 n$-far from the code be rejected with probability at least 1/2.

As usual, we focus on PCP systems with *logarithmic randomness complexity* and *constant query complexity*. This means that, without loss of generality, the length of the oracle is polynomial in the length of the input. However, we aim at PCP systems that utilize oracles that are of almost-linear length. Our main result regarding such PCP systems is

Theorem 2.3 *For every $c > 0.5$, there exists an almost-linear time randomized reduction of SAT to a promise problem that has a 19-query PCP system that utilizes oracles of length $\exp((\log n)^c) \cdot n = n^{1+o(1)}$, where n is the length of the input. Furthermore, the reduction maps k-bit inputs to n-bit inputs such that $n = \exp((\log k)^c) \cdot k = k^{1+o(1)}$.*

This should be compared to the PCP system for SAT of Polishchuk and Spielman [17] that when utilizing oracles of length $n^{1+\epsilon}$ makes $O(1/\epsilon)$ queries. In contrast, our PCP system utilizing oracles of length $n^{1+o(1)}$ while making 19 queries.

3 Direct Constructions of Codes

In this section, we prove Theorem 2.2. Although we do not use any variant of the PCP Theorem, our constructions are somewhat related to known PCP constructions in the sense that we use codes (and analysis) that appear (at least implicitly) in the latter. Specifically, we will use results regarding low-degree tests that were proven for deriving the PCP Theorem [1, 2]. We stress that we neither use the (complex) parallelization procedure (of [1, 2]) nor the full power of the proof composition paradigm (of [2], which is more complex than the classical notion of concatenated codes [10] used below).

3.1 The Basic Code (FS-Code)

Our starting point is a code proposed by Friedl and Sudan [11] based on a low-degree test due to Rubinfeld and Sudan [19].

Let F be a finite field, and m, d be integer parameters such that (typically) $m \leq d < |F|$. Denote by $P_{m,d}$ the set of m-variate polynomials of total degree d over F. We represent each $p \in P_{m,d}$ by the list of its $\binom{m+d}{d}$ coefficients; that is, $|P_{m,d}| = |F|^{\binom{m+d}{d}}$. (For $m \leq d$, we use $|P_{m,d}| < |F|^{(2d/m)^m}$.)

Denote by L_m the set of lines over F^m, where each line is define by two points $a, b \in F^m$; that is, for $a = (a_1, ..., a_m)$ and $b = (b_1, ..., b_m)$, the line $\ell_{a,b}$ consists of the set of $|F|$ points $\{\ell_{a,b}(t) \stackrel{\text{def}}{=} ((a_1 + tb_1), ..., (a_m + tb_m)) : t \in F\}$.

We consider the code $\mathcal{C} : P_{m,d} \to \Sigma^{|L_m|}$, where $\Sigma = F^{d+1}$; that is, $\mathcal{C}$ assigns each $p \in P_{m,d}$ a ($|L_m|$-long) sequence of Σ-values, where each Σ-value corresponds to a different element of L_m. The element associated with $\ell \in L_m$ in the ($|L_m|$-long) sequence $\mathcal{C}(p)$, denoted $\mathcal{C}(p)_\ell$, is the univariate polynomial that represents the values of the polynomial $p : F^m \to F$ on the line ℓ; that is, for $\ell_{a,b} \in L_m$, the univariate polynomial $\mathcal{C}(p)_{\ell_{a,b}}$ can be formally written as $q_{a,b}(z) \stackrel{\text{def}}{=} p(\ell_{a,b}(z)) = p((a_1 + b_1 z), ..., (a_m + b_m z))$. Since the polynomial p has total degree d, so does the univariate polynomial $q_{a,b}$.

To evaluate the basic parameters of the code $\mathcal{C}$, let use consider it as mapping $\Sigma^k \to \Sigma^n$, where indeed $n = |L_m| = |F|^{2m}$ and $k = \log |P_{m,d}| / \log |\Sigma|$. Note that

$$k = \frac{\log |P_{m,d}|}{\log |\Sigma|} = \frac{\binom{m+d}{d} \log |F|}{(d+1) \log |F|} = \frac{\binom{m+d}{m}}{d+1} \tag{1}$$

which, for $m \ll d$, is approximated by $(d/m)^m/d \approx (d/m)^m$. Using $|F| = \text{poly}(d)$, we have $n = |F|^{2m} = \text{poly}(d^m)$, and so k is polynomially related to n (provided, say, $m < \sqrt{d}$). Note that the code has large distance (since the different $\mathcal{C}(p)$'s tend to disagree on most lines).

The Codeword Test: The test consists of selecting two random lines that share a random point, and checking that the univariate polynomials associated with these lines yield the same value for the shared point. That is, to check whether $w \in \Sigma^{|L_m|}$ is a codeword, we select a random point $r \in F^m$, and two random lines ℓ', ℓ'' going through r (i.e., $\ell'(t') = r$ and $\ell''(t'') = r$ for some $t', t'' \in F$), obtain the answer polynomials q' and q'' (i.e., $q' = w_{\ell'}$ and $q'' = w_{\ell''}$) and check whether they agree on the shared point (i.e., whether $q'(t') = q''(t'')$). This test is essentially the one analyzed in [1], where it is shown that (for $|F| = \text{poly}(d)$) if the oracle is ϵ-far from the code then this is detected with probability $\Omega(\epsilon)$.

3.2 Random Truncation of the FS-Code

Our aim is to tighten the relation between k and n. Recall that the gap between them is due to two sources; firstly, the analysis in [1] required a field F that is polynomially bigger than the degree d. This problem can be eliminated using the better analysis of [17], which only requires $|F| = \Omega(d)$ (see [11]). The second problem is that n is quadratic in $|F|^m$, whereas $k = o(d^m) = o(|F|^m)$. Thus, to obtain n almost-linear in k, we must use a different code.

We will use a random projection (or "truncation") of the FS-code on approximately $|F|^m$ of the coordinates. Let $R_m \subset L_m$ be a random subset of $O(|F|^m \log |F|)$ lines. We consider the code $\mathcal{C}^{R_m} : P_{m,d} \to \Sigma^{|R_m|}$, where the element associated with $\ell_{a,b} \in R_m \subset L_m$ in the sequence $\mathcal{C}^{R_m}(p)$ is the univariate polynomial that represents the values of the polynomial $p : F^m \to F$ on the line $\ell_{a,b}$. When R_m is (unimportant or) understood from the context, we shorthand $\mathcal{C}^{R_m}$ by $\mathcal{C}$.

To evaluate the basic parameters of the code $\mathcal{C}$, let use consider it as mapping $\Sigma^k \to \Sigma^n$, where $n = |R_m| = O(|F|^m \log |F|)$ (and as before $k = \log |P_{m,d}|/\log |\Sigma|$). Thus, for $m \ll d$, we have $k \approx d^{m-1}/m^m$ and, for $|F| = O(d)$, we have $n = O(|F|^m \log |F|) = O(d)^m$. We highlight two possible settings of the parameters:

1. Using $d = m^m$, we get $k \approx m^{m^2-2m}$ and $n = m^{m^2+o(m)}$, which yields $n \approx \exp(\sqrt{\log k}) \cdot k$ and $\log |\Sigma| = \log |F|^{d+1} \approx d \log d \approx \exp(\sqrt{\log k})$.

2. Letting $d = m^e$ for constant $e > 1$, we get $k \approx m^{(e-1)m}$ and $n \approx m^{em}$, which yields $n \approx k^{e/(e-1)}$ and $\log |\Sigma| \approx d \log d \approx (\log k)^e$.

The Codeword Test: The original codeword test can be extended to the current setting. Specifically, the new test consists of selecting two random lines in R_m that share a random point, and checking that the univariate polynomials associated with these lines yield the same value for the shared point. (We stress that we first select uniformly a point $r \in F^m$, and next select two lines in R_m that pass through r.) We prove that this codeword test for the randomly-truncated code $\mathcal{C}^{R_m}$ works as well as the codeword test for the basic FS-code.

Lemma 3.1 *Let $|F| = \Omega(d)$ and $|F| < \exp(m^m)$. Then, for $1 - o(1)$ fraction of the possible choices of R_m of size n, the following holds for every $w \in \Sigma^n$: if the distance of w from the code $\mathcal{C}^{R_m}$ is ϵn then the probability that the above codeword test rejects is $\Omega(\epsilon) - o(1)$.*

Proof sketch: First we reduce the analysis of the above codeword test (which compares the value given to two intersecting lines) to an analysis of a point-vs-line test that compares the value of

a *suitable* function $f : F^m \to F$ on a random point with the value induced by (the polynomial associated with) a random line passing through this point. Fixing any R_m and any $w \in \Sigma^n$, we construct a random function $f : F^m \to F$ by selecting uniformly, for each $r \in F^m$, a line ℓ in R_m that passes through r and setting $f(r)$ accordingly (i.e., $f(r) = w_\ell(t)$ where $r = \ell(t)$). We note that the probability that the original intersecting-lines test accepts w equals the probability that the point-vs-line test accepts w *along with the resulting random* f, because the (random) value $f(r)$ (obtained from f) may be viewed as obtained from a (second) random line that passes through r. Thus, it suffices to analyze the point-vs-line test as applied to w and the corresponding random f. This will be done in two stages: In the first stage we relate the distance of w from the code $\mathcal{C} = \mathcal{C}^{R_m}$ to the distance of f from the set $P_{m,d}$, and in the second stage we relate the rejection probability of w and f to the distance of f from $P_{m,d}$.

First stage: We will show that for every $p \in P_{m,d}$, the (fractional) distance of f from p approximates the (fractional) distance of w from $\mathcal{C}(p)$. For simplicity, we first assume that R_m covers all points uniformly (i.e., each point in F^m resides appears in exactly $|F| \cdot |R_m|/|F^m|$ lines of R_m). Let $p \in P_{m,d}$ and denote by εn the distance of w from $\mathcal{C}(p)$; that is, $w_\ell \neq \mathcal{C}(p)$ $(= p(\ell))$ on an ε fraction of the ℓ's in R_m. For each $\ell \in R_m$ for which $w_\ell \neq \mathcal{C}(p)$ it is the case that w_ℓ disagrees with p on almost all (i.e., all but d) points that reside on the line ℓ (because both $w_\ell(\cdot)$ and $p(\ell(\cdot))$ are low-degree polynomials that determine the corresponding values). Since $f(r)$ is defined according to a random line $\ell \in R_m$ that passes through r, it holds that the expected (fractional) disagreement of a random f with p is at least $(1 - (d/|F|)) \cdot \varepsilon$.[4] Furthermore, since f is define independently on each point of F^m, with probability at least $1 - \exp(\varepsilon |F|^m)$, a random f disagrees with p on at least a $\varepsilon/2$ fraction of the points. Using the union bound (for all $p \in P_{m,d}$) and $|P_{m,d}| < |F|^{(2d/m)m} \ll 2^{\varepsilon |F|^m}$ (for $\varepsilon > 2^{-m}$), with very high probability, the distance of a random f from every $p \in P_{m,d}$ (i.e., f's distance from $P_{m,d}$) approximates (up-to an additive term of $(\epsilon/2) - o(1)$) the distance of w from the corresponding $\mathcal{C}(p)$. We conclude that the expected distance of a random f from the set $P_{m,d}$ approximates the distance of w from the code $\mathcal{C}$.

Recall that, in the above analysis, we have assumed that R_m covers all points uniformly (i.e., each point resides on the same number of lines in R_m). In general, this is not the case. Yet, with very high probability, a random set R_m cover almost all points in an almost uniform manner. This "almost uniformity" suffices for extending the above analysis.[5] Thus, for almost all R_m's, the distance of w from the code $\mathcal{C}^{R_m}$ is well-approximated by the distance of a corresponding random

[4] Envision a table with row corresponding to lines in R_m, columns corresponding to points, and entries corresponding to pairs such that the pair (r, ℓ) is marked if r resides on ℓ, where the marking equals the value of the point r as determined by the polynomial assigned to the line ℓ. Indeed, exactly $|F|$ entries in each row are marked. By the uniformity condition, exactly $|F| \cdot |R_m|/|F^m|$ entries in each column are marked. By the hypothesis that ε fraction of the lines don't fit p, it follows that for an ε fraction of the rows, at most d of the entries are marked in agreement with p (and the rest are marked in disagreement with p). It follows that at least $\varepsilon' \stackrel{\text{def}}{=} (1 - (d/|F|)) \cdot \varepsilon$ fraction of all marked entries are marked in disagreement with p. In other words, ε' equals the expected fraction of disagreement among the marked entries in a random column. But the fraction of disagreements among the marked entries in column r equals the probability that $f(r) \neq p(r)$, where f is a random function constructed as above (because $f(r)$ is assigned at random a value according to a uniformly selected marked entry in column r). Thus, the expected distance of f from p equals ε'.

[5] Alternatively, for almost all R_m's, there exists a set $S_m \subset L_m$ that covers all points uniformly such that $|S_m| = n = |R_m|$ and $|R_m \cap S_m| = n - o(n)$. In order to analyze the construction of f based on w with respect to the code $\mathcal{C}^{R_m}$, we consider the construction of g based on u with respect to the code $\mathcal{C}^{S_m}$, where $u_\ell = w_\ell$ for every $\ell \in R_m \cap S_m$. By the above analysis, the expected distance of g from $P_{m,d}$ approximates the distance of u from $\mathcal{C}^{S_m}$, which in turn approximates the distance of w from $\mathcal{C}^{R_m}$ (because u and w as well as $\mathcal{C}^{S_m}$ and $\mathcal{C}^{R_m}$ agree on all $\ell \in R_m \cap S_m$). Finally, observe that the expected fractional distance between g and f is $o(1)$, because for all but an $o(1)$ fraction of the points r all but an $o(1)$ fraction of the lines of R_m that pass through r are also in S_m and vice versa.

function f from the set $P_{m,d}$.

Second stage: We turn to analyze the performance of the point-vs-line test applied to any $w \in \Sigma^n$ and a corresponding random $f : F^m \to F$ (constructed as above). Following [19, 1, 2, 17], we observe that for each possible function $f : F^m \to F$ there exists an optimal strategy of answering all possible line-queries (such that the acceptance probability of the line-vs-point test is maximized). Specifically, for a fixed function f, and each line ℓ, the optimal way to answer the line-query ℓ is given by the degree d univariate polynomial that agrees with the value of f on the maximum number of points of ℓ. Thus, the optimal acceptance probability of the line-vs-point test on f depends only on f (and not on w, which may not be optimal for f). Furthermore, this probability is the average of quantities (i.e., the agreement of f with the best univariate polynomial) that f associates with each of the possible lines. Let us denote by $D_\ell(f)$ the fractional disagreement of f restricted to ℓ with the best univariate polynomial. Then, by the relevant results in [1, 2, 17], the average of $D_\ell(f)$ *taken over all lines* (i.e., over L_m) is linearly related to the distance of f from $P_{m,d}$. Clearly, the rejection probability of our test (i.e., the line-vs-point test for lines uniformly selected in R_m, when applied to w and f as above) is lower-bounded by the average of the $D_\ell(f)$'s over the lines in R_m (rather than over the set of all lines, L_m). Now, *for each fixed* f, with probability $1 - \exp(-|R_m|)$, the average of the $D_\ell(f)$'s (taken over all lines) is approximated (up-to some constant) by the average taken over a random set R_m. Taking the union bound over all $|F|^{|F|^m}$ functions f's we conclude that, for almost all R_m, the point-vs-line test rejects each f with probability that is linearly related to the distance of f from $P_{m,d}$ (because $\exp(-|R_m|) \cdot |F|^{|F|^m} = o(1)$).

By the first stage, for almost all R_m's, the distance of each w is related to the expected distance of a corresponding random f from $P_{m,d}$, whereas by the second stage (for almost all R_m's) each w and the corresponding random f is rejected by the point-vs-line with probability that is linearly related to the distance of f from $P_{m,d}$. Combining these two facts, the lemma follows. ∎

F-linearity: The (modified as well as the original) code $\mathcal{C}$ is F-linear; that is, the individual F-elements in the codeword sequence are linear combinations (over F) of the F-elements in the information being encoded. Equivalently, for every $\alpha', \alpha'' \in F$ and every $p', p'' \in P_{m,d}$, it holds that $\mathcal{C}(\alpha' p' + \alpha'' p'')_\ell = \alpha' \mathcal{C}(p')_\ell + \alpha'' \mathcal{C}(p'')_\ell$, for every line ($\Sigma$-coordinate) ℓ. This is the case because $\mathcal{C}(\alpha' p' + \alpha'' p'')_\ell$ equals the univariate polynomial (in z) given by $(\alpha' p' + \alpha'' p'')(\ell(z)) = \alpha' p'(\ell(z)) + \alpha'' p''(\ell(z))$, which in turn equals $\alpha' \mathcal{C}(p')_\ell + \alpha'' \mathcal{C}(p'')_\ell$.

Conclusion: Using the first parameter-setting (i.e., $d = m^m$), we establish Part 1 of Theorem 2.2.

An alternative construction: To simplify the analysis of the codeword test, we may construct an alternative code in which $\mathcal{C}$ is augmented by an evaluation of the polynomial p on all possible points (i.e., F^m). Furthermore, the augmentation is repeated enough times (i.e., $\Omega((d+1) \cdot \log |F|)$ times) such that this portion dominates the length of the code (as well as the distance to it). Using the alternative construction allows to directly apply the analysis of [1, 2, 17] (while confining ourselves to analyzing the effect of taking a sample R_m of the quantities assigned by f to all possible lines). On the other hand, using the alternative code will slightly complicate the next subsection.

3.3 Decreasing the alphabet size

The above construction uses quite a big alphabet (i.e., $\Sigma = F^{d+1}$). Our aim in this subsection is to maintain the above performance while using a smaller alphabet (i.e., F rather than F^{d+1}). This is achieved by concatenating the above code (which encodes information by a sequence of n degree d

univariate polynomials over F) with the following inner-code that maps F^{d+1} to $F^{n'}$, where n' is sub-exponential in $k' \stackrel{\text{def}}{=} d+1$.

For a (suitable) constant d', let $k' = h^{d'}$ and $[h] = \{1, ..., h\}$. As a warm-up, consider the special case of $d' = 2$. In this case, the code $\mathcal{C}'$ maps bilinear forms in x_i's and y_i's (with coefficients $(c_{i,j})_{i,j\in[h]}$) to the values of these forms under all possible assignments. That is, $\mathcal{C}' : F^{h^2} \to F^{|F|^{2h}}$ maps the sequence of coefficients $(c_{i,j})_{i,j\in[h]}$ to the sequence of values $(v_{a_1,...,a_h,b_1,...,b_h})_{a_1,...,a_h,b_1,...,b_h\in F}$ where $v_{a_1,...,a_h,b_1,...,b_h} = \sum_{i,j\in[h]} c_{i,j} \cdot a_i b_j$. In general (i.e., arbitrary $d' \geq 1$), the inner-code $\mathcal{C}' : F^{k'} \to F^{n'}$ maps d'-linear forms in the variables sets $\{z_i^{(1)} : i \in [h]\}, ..., \{z_i^{(d')} : i \in [h]\}$ to the values of these d'-linear forms under all possible assignments to these $d'h$ variables. That is, $\mathcal{C}'$ maps the sequence of coefficients $(c_{i_1,...,i_{d'}})_{i_1,...,i_{d'}\in[h]}$ to the sequence of values $(v_{a_1^{(1)},...,a_h^{(1)},...,a_1^{(d')},...,a_h^{(d')}})_{a_1^{(1)},...,a_h^{(1)},...,a_1^{(d')},...,a_h^{(d')}\in F}$ where $v_{a_1^{(1)},...,a_h^{(1)},...,a_1^{(d')},...,a_h^{(d')}} = \sum_{i_1,...,i_{d'}\in[h]} c_{i_1,...,i_{d'}} \cdot \prod_{j=1}^{d'} a_{i_j}^{(j)}$. Thus, ($k' = h^{d'}$ and) $n' = |F|^{d'h} = \exp(d' \cdot (k')^{1/d'} \log |F|)$.

Testing the inner-code: A valid codeword is a multi-linear function (in the variable sets $\{z_i^{(1)} : i \in [h]\}, ..., \{z_i^{(d')} : i \in [h]\}$); that is, for each j, a valid codeword is linear in the variables $z_i^{(j)}$'s. Thus, testing whether a sequence belongs to the inner-code amounts to d' linearity checks. Specifically, for each j, we randomly select $\overline{r} = (r_1^{(1)}, ..., r_h^{(1)}, ..., r_1^{(d')}, ..., r_h^{(d')})$ and $s_1^{(j)}, ..., s_h^{(j)}$, and compare $v_{\overline{r}} + v_{0,...,0,s_1^{(j)},...,s_h^{(j)},0,...,0}$ to $v_{(t_1^{(1)},...,t_h^{(1)},...,t_1^{(d')},...,t_h^{(d')})}$, where $t_i^{(j)} = r_i^{(j)} + s_i^{(j)}$ and $t_i^{(j')} = r_i^{(j')}$ for $j' \neq j$. To simplify the analysis, we also let the test employ a total low-degree test (to verify that the codeword is a multi-variate polynomial of total-degree d').[6] (The total-degree test uses $d'+2$ queries, and so our codeword test uses $3d' + d' + 2$ queries.)

Lemma 3.2 *If the distance of $w' \in F^{n'}$ from $\mathcal{C}'$ is $\epsilon n'$ then the probability that the codeword test for $\mathcal{C}'$ rejects is $\Omega(\epsilon)$.*

Proof sketch: If $w' \in F^{|F^{d'h}|}$ (viewed as a function $F^{d'h} \to F$) is at fractional distance at least $\min(\epsilon, 0.5)$ from the set of $d'h$-variate polynomials of total degree d' then it is rejected with probability $\Omega(\epsilon)$ by the total-degree test. Otherwise, w' is at distance less than $\min(\epsilon, 0.5) \cdot n'$ from such a polynomial, denoted p', which is unique. By the hypothesis (regarding the distance of w' from $\mathcal{C}'$), this p' must be non-linear in some block of variables (i.e., in the $z_i^{(j)}$'s). With probability $1 - (d'/|F|) > 0.9$ this non-linearity is preserved when assigning random values to the variables of *all the other blocks*. On the other hand, the expected fractional distance between the residual w' and p' under such a random assignment is less than 0.5. Thus, under such random assignment, the expected fractional distance of the residual w' from the set of linear functions in the $z_i^{(j)}$'s is at least $0.9 - 0.5 = 0.4$. It follows that w' is rejected with constant probability by the j^{th} linearity test (because, with probability at least 0.2, the residual w' is at least 0.2-far from being linear in the $z_i^{(j)}$'s). ■

Testing the concatenated-code: In order to test the concatenated code, we first test (random instances of) the inner-code and next use self-correction on the latter to emulate the testing of the outer-code. Specifically, the tester for the concatenated code selects at random two intersecting lines ℓ' and ℓ'', and first applies the inner-code tester to the inner-encoding of the polynomials

[6] We conjecture that the codeword test operates well also without employing the total-degree test, but the augmented codeword test is certainly easier to analyze.

associated by the outer code to these lines. To emulate the outer-code test, the current tester needs to obtain the value of these polynomials at some elements of F (which are determined by the outer test). Suppose that we need the value of q' (a univariate polynomial of degree $d = h^{d'} - 1$ over F) at $t \in F$, and that q' is encoded by the inner-code. However, the value $q'(t) = \sum_{i_1,...,i_{d'} \in [h]} q'_{i_1,...,i_{d'}} t^{(i_1-1)+(i_2-1)h+\cdots+(i_{d'}-1)h^{d'-1}}$ equals the entry of $\mathcal{C}'(q')$ that is associated with the sequence $(t^0, ..., t^{h-1}, t^0, ..., t^{(h-1)h}, ..., t^0, ..., t^{(h-1)h^{d'-1}})$.[7] Self-correction of the desired entry is performed via polynomial interpolation, and requires only $d' + 1$ queries. Thus, the concatenated code can be tested by making $O(d')$ queries.

Notes: Observe that the inner-code is linear (over F), and thus so is also the concatenated code. Furthermore, the codeword test is a conjunction of $(O(d'))$ linear tests. Alternatively, we may perform one of these linear tests, selected at random (with equal probability). Regarding the parameters of the concatenated code, suppose that in the outer-code we use the setting $d = m^e$ (for constant $e > 1$), and in the inner-code we use $d' = 2e$. Then, we obtain a code that maps $F^{kk'}$ to $F^{nn'}$, where $n \approx k^{e/(e-1)}$ and $n' \approx \exp(d^{1/d'}) \approx \exp((\log k)^{e/d'}) = \exp(\sqrt{\log k}) = k^{o(1)}$ (using $d \approx (\log k)^e$). Thus, $nn' \approx (kk')^{e/(e-1)}$ and $|F| = O(d) \approx (\log k)^e$ (as before).

3.4 A Binary Code

The last step is to derive a binary code. This is done by concatenating the above code with a Hadamard code, while assuming that $F = GF(2^{k''})$. The Hadamard code is used to code elements of F by binary sequences of length $n'' \stackrel{\text{def}}{=} 2^{k''}$.

To test the newly concatenated code, we combine the obvious testing procedure for the Hadamard code with the fact that all that we need to check for the current outer-code are (a constant number of) linear (in F) conditions involving a constant number of F-entries. Instead of checking such a linear condition over F, we check that the corresponding equality holds for a random sum of the bits in the representation of the elements of F (using the hypothesis that $F = GF(2^{k''})$). Specifically, suppose that we need to check whether $\sum_i \alpha_i a_i = 0$ (in F), for some known $\alpha_i \in F$ and oracle answers denoted by a_i's. Then, we uniformly select $r \in GF(2^{k''})$, and check whether $\langle r, \sum_i \alpha_i a_i \rangle \equiv 0 \bmod 2$ holds, where $\langle u, v \rangle$ denotes the inner-product modulo 2 of (the $GF(2^{k''})$ elements) u and v (viewed as k''-bit long vectors). The latter check is performed by relying on the following two facts:

1. $\langle r, \sum_i \alpha_i a_i \rangle \equiv \sum_i \langle r, \alpha_i a_i \rangle \bmod 2$.
2. Each $\langle r, \alpha_i a_i \rangle$ can be obtained by making a single query (which is determined by r and α_i) to the Hadamard coding of a_i, because $\langle r, \alpha_i a_i \rangle$ is merely a linear combination with coefficients depending only on α_i and r of the bits of a_i. (Each bit of $\alpha_i a_i \in GF(2^{k''})$ is a linear combination with coefficients depending only on α_i of the bits of a_i, and $\langle r, v \rangle$ is a linear combination with coefficients depending only on r of the bits of v.)

Thus, the emulation of the outer-code test is performed by accessing a constant number of entries in the inner-code. It follows that the final concatenated code is locally testable. The final code

[7] That is, we consider the entry of $\mathcal{C}'(q')$ that is associated with the sequence $(a_1^{(1)}, ..., a_h^{(1)}, ..., a_1^{(d')}, ..., a_h^{(d')})$ that satisfies $a_i^{(j)} = t^{(i-1)h^{j-1}}$. The value of this entry equals $\sum_{i_1,...,i_{d'} \in [h]} q'_{i_1,...,i_{d'}} \cdot \prod_{j=1}^{d'} a_{i_j}^{(j)}$, which equals $\sum_{i_1,...,i_{d'} \in [h]} q'_{i_1,...,i_{d'}} \cdot \prod_{j=1}^{d'} t^{(i_j-1)h^{j-1}} = \sum_{i_1,...,i_{d'} \in [h]} q'_{i_1,...,i_{d'}} \cdot t^{\sum_{j=1}^{d'} (i_j-1)h^{j-1}}$.

maps $\{0,1\}^{kk'k''}$ to $\{0,1\}^{nn'n''}$, where $nn' \approx (kk')^{e/(e-1)}$ and $n'' = 2^{k''} = |F| = \text{poly}(\log k) = k^{o(1)}$. Thus, $nn'n'' \approx (kk'k'')^{e/(e-1)}$. This establishes Part 2 of Theorem 2.2.

Note: Fixing any integer $e > 1$, the above code can be constructed for any integer h, while determining $k' = h^e$, $k'' = \log O(k')$ and $k \approx (m^{e-1})^m$, where $m = (h^e - 1)^{1/e} \approx h$. Thus, $K \stackrel{\text{def}}{=} kk'k'' \approx h^{(e-1)h} \cdot h^e \cdot \log h^e \approx h^{(e-1)h}$. The ratio between consecutive values of K is given by $\frac{(h+1)^{(e-1)(h+1)}}{h^{(e-1)h}} = O(h)^{e-1} < (\log K)^{e-1}$, and so the successor of K is smaller than $(\log K)^{e-1} \cdot K$.

4 Nearly linear-sized PCPs

In this section we give a probabilistic construction of nearly-linear sized PCPs for SAT. More formally, we reduce SAT probabilistically to a promise problem recognized by a PCP verifier tossing $(1+o(1))\log n$ random bits (on inputs of length n) and queries a proof oracle in a constant number of bits and has perfect completeness and soundness arbitrarily close to $\frac{1}{2}$. We stress that the constant number of bits is explicit and small. Specifically, if the $o(1)$ function in the randomness is allowed to be as large as $1/\text{poly}\log\log n$, then the number of queries can be reduced to 16 bits. The little $o(1)$ function can be reduced to $O(\sqrt{\log\log n/\log n})$ for a small cost in the number of queries, which now goes up to 19 bits. These improvements are obtained by using/improving results of Harsha and Sudan [15].

We get our improvements by applying the "random truncation" method (introduced in Section 3) to certain *constant-prover one-round proof systems*, which are crucial ingredients in the constructions of PCPs. Typically, these proof systems use provers of very different sizes, and by applying the "random truncation" method we obtain an equivalent system in which all provers have size roughly equal to the size of the smallest prover in the original scheme. At this point, we reduce the randomness complexity to be logarithmic in the size of the provers (i.e., and thus logarithmic in the size of the smallest original prover).

Recall that typical PCP constructions are obtained by the technique of proof composition introduced by Arora and Safra [2]. In this technique, an "outer verifier", typically a verifier for a constant prover one round proof system, is composed with an "inner verifier" to get a new PCP verifier. The new verifier essentially inherits the randomness complexity of the outer verifier and the query complexity of the inner verifier. Since our goal is to reduce the randomness complexity of the composed verifier, we achieve this objective by reducing the randomness complexity of the outer verifier.

As stated above, the key step is to reduce the sizes of the provers. As a warm-up, we first show that the random truncation method can be applied to any 2-prover one-round proof system, where the size of one prover is much larger than the size of the second prover, to reduce the size of the larger prover to roughly the size of the smaller prover.

We then show how to apply the random truncation to the verifier of a specific 3-prover one-round proof system used by Harsha and Sudan [15]. Their verifier is a variant of the one constructed by Raz and Safra [18] (see also, Arora and Sudan [3]), which are, in turn, variants of a verifier constructed by Arora et al. [1]. All these verifiers share the common property of working with provers of "imbalanced" sizes. We manage to reduce the size of the provers to the size of the smallest one, and consequently reduce the randomness of the verifier to $(1+o(1))\log n$ (i.e., logarithmic in the prover size). We stress that this part is not generic but relies on properties of the proof of soundness in, say, [15], which are abstracted below. Applying the composition lemmas used/developed in [15] to this new verifier gives us our efficient PCP constructions.

4.1 MIP verifiers and random sampling

We start by defining a 2-prover 1-round proof system as a combinatorial game between a verifier and two provers. Below, Ω denotes the space of verifier's coins, q_i denotes its strategy of forming queries to the ith prover, and P_i denote strategies for answering these queries (where all refer to the residual strategies for a fixed common input).

Definition 4.1 *For finite sets Q_1, Q_2, Ω, and A, a (Q_1, Q_2, Ω, A)-2IP verifier V is given by functions $q_1 : \Omega \to Q_1$ and $q_2 : \Omega \to Q_2$ and* Verdict $: \Omega \times A \times A \to \{0,1\}$. *The value of V, denote $w(V)$, is the maximum, over all functions $P_1 : Q_1 \to A$ and $P_2 : Q_2 \to A$ of the quantity $\mathbf{E}_{r \leftarrow \Omega}[\text{Verdict}(r, P_1(q_1(r)), P_2(q_2(r)))]$. A 2IP verifier V is said to be* uniform *if for each $i \in \{1,2\}$, the distributions $\{q_i(r)\}_{r \leftarrow \Omega}$ are uniform over Q_i.*

Focusing on the case $|Q_2| \gg |Q_1|$, we define a "sampled" 2IP verifier:

Definition 4.2 *Given a (Q_1, Q_2, Ω, A)-2IP verifier V and set $S \subseteq Q_2$, let $\Omega_S = \{r \in \Omega \mid q_2(r) \in Q_1\}$. For $T \subseteq \Omega_S$, the (S,T)-*sampled 2IP verifier *$V|_{S,T}$ is a (Q_1, S, T, A)-2IP verifier given by functions $q_1' : T \to Q_1$, $q_2' : T \to S$, and* Verdict$' : T \times A \times A \to \{0,1\}$ *obtained by restricting q_1, q_2 and* Verdict *to T.*

In the following lemma we show that a sufficiently large randomly sampled set S from Q_2 is very likely to preserve the value of a verifier approximately. Furthermore, the value continues to be preserved approximately if we pick T to be a sufficiently large random subset of Ω_S.

Lemma 4.3 *There exist absolute constants c_1, c_2 such that the following holds for every Q_1, Q_2, Ω, A, ϵ and $\gamma > 0$. Let V be an (Q_1, Q_2, Ω, A)-uniform 2IP verifier.*

Completeness: *Any (S,T)-sampled verifier preserves the perfect completeness of V. That is, if $\omega(V) = 1$ then, for every $S \subseteq Q_2$ and $T \subseteq \Omega_S$, it holds that $\omega(V|_{S,T}) = 1$.*

Soundness: *For sufficiently large S and T, a random (S,T)-sampled verifier preserves the soundness of V up-to a constant factor. Specifically, let $N_1 = \frac{c_1}{\epsilon}\left(|Q_1| \log |A| + \log \frac{1}{\gamma}\right)$ and $N_2 = \frac{c_2}{\epsilon}\left(N_1 \log |A| + \log \frac{1}{\gamma}\right)$, and suppose that S is a uniformly selected multiset of size N_1 of Q_2, and T is a uniformly selected multiset of size N_2 of Ω_S. Then, for $\omega(V) \leq \epsilon$, with probability at least $1 - \gamma$ it holds that $\omega(V|_{S,T}) \leq 2\epsilon$.*

Note that the reduction in randomness complexity (i.e., obtaining $N_2 = \tilde{O}(|Q_1|)$) relies on the shrinking of the second prover to size $N_1 = \tilde{O}(|Q_1|)$. Without shrinking the second prover, we would obtain $N_2 = \tilde{O}(|Q_2|)$, which is typically useless (becuase, typically, $|\Omega| = \tilde{O}(|Q_2|)$).

Proof: Assuming that $\omega(V) \leq \epsilon$, we focus on the soundness condition. The proof is partitioned into two parts. First we show that a random choice of S is unlikely to increase the value of the game to above $3/2\epsilon$. Next, assuming the first part was ok, we show that a random choice of T is unlikely to increase the value of the game above 2ϵ. The second part of the proof is really a standard argument which has been observed before in the context of PCPs. We thus focus on the first part, which abstracts the idea of the random truncation from Section 3.

Our aim is to bound the value $\omega(V|_{S,\Omega_S})$, for a randomly chosen S. Fix any prover strategy $P_1 : Q_1 \to A$ for the first prover. Now note that an optimal function, denoted P_2^*, for the second prover answer each question $q_2 \in Q_2$ by an answer that maximizes the acceptance probability with respect to the fixed P_1 (i.e., an optimal answer is a string a_2 that

maximizes $\mathbf{E}_{r\in\Omega|q_2(r)=q_2}[\mathrm{Verdict}(r,P_1(q_1(r)),a_2)]$). We stress that this assertion holds both for the original 2IP verifier V as well as for any (S,Ω_S)-sampled verifier.[8] For every question $q_2 \in Q_2$, let ϵ_{q_2} denote the acceptance probability of the verifier V given that the second question is q_2 (i.e., $\epsilon_{q_2} = \mathbf{E}_{r\in\Omega|q_2(r)=q_2}[\mathrm{Verdict}(r,P_1(q_1(r)),P_2^*(q_2))]$). By definition (and uniformity) $E_{q_2\in Q_2}[\epsilon_{q_2}] = E_{r\in\Omega}[\epsilon_{q_2(r)}] \leq \epsilon$. The quantity of interest to us is $E_{r\in\Omega_S}[\epsilon_{q_2(r)}] = \mathbf{E}_{q_2\in S}[\epsilon_{q_2}]$. A straightforward application of Chernoff bounds shows that the probability that this quantity exceeds $(3/2)\epsilon$ is $\exp(-\epsilon N_1)$. Taking the union bound over all possible P_1's, we infer that the probability that there exists a P_1, P_2 such that $\mathbf{E}_{r\leftarrow\Omega_S}[\mathrm{Verdict}(r,P_1(q_1(r)),P_2(q_2(r)))] > (3/2)\epsilon$ is at most $\exp(-\epsilon N_1)\cdot|A|^{|Q_1|}$. Thus, using $N_1 = \frac{c_1}{\epsilon}\left(|Q_1|\log|A| + \log\frac{1}{\gamma}\right)$ (for some absolute constant c_1), it follows that $\omega(V|_{S,\Omega_S}) \leq (3/2)\epsilon$ with probability at least $1-\frac{\gamma}{2}$ (over the choices of S). The lemma follows.[9] ∎

4.2 Improved 3-Prover Proof System for NP

We now define the more general notion of a constant-prover one-round interactive proof system (MIP).

Definition 4.4 *For positive reals c, s, integer p and functions $r, a : \mathsf{Z}^+ \to \mathsf{Z}^+$, we say that a language $L \in \mathrm{MIP}_{c,s}[p,r,a]$* (or, *$L$ has a p-prover one-round proof system with answer length a) if there exists a probabilistic polynomial-time verifier V interacting with p provers $P_1,\ldots,P_p$ such that*

Operation: *On input x of length n, the verifier tosses $r(n)$ coins, generates queries $q_1,\ldots,q_p$ to provers $P_1,...,P_p$, obtain the corresponding answers $a_1,\ldots,a_p \in \{0,1\}^{a(n)}$, and outputs a Boolean verdict that is a function of x, its randomness and the answers $a_1,\ldots,a_p$.*

Completeness: *If $x \in L$ then there exist strategies $P_1,\ldots,P_p$ such that V accepts their response with probability at least c.*

Soundness: *If $x \notin L$ then for every sequence of prover strategies $P_1,\ldots,P_p$, machine V accepts their response with probability at most s.*

Harsha and Sudan [15] presented a randomness efficient 3-prover one-round proof system with answer length $\mathrm{poly}(\log n)$ and randomness complexity $(3+\epsilon)\log_2 n$, where $\epsilon > 0$ is an arbitrary constant and n denotes the length of the input. Here we reduce the randomness required by their verifier to $(1+o(1))\log n$.

Before going on we introduce a notion that will be useful in this section — namely, the notion of a *length preserving reduction*. For a function $\ell : \mathsf{Z}^+ \to \mathsf{Z}^+$, a reduction is $\ell(n)$-length preserving if it maps instances of length n to instances of length at most $\ell(n)$.

Lemma 4.5 *For every $\epsilon > 0$ and functions $m(n)$, $\ell(n)$ satisfying $\ell(n) = \Omega(m(n)^{\Omega(m(n))}n^{1+\Omega(1/m(n))})$, SAT reduces in probabilistic polynomial time, under $\ell(n)$-length preserving reductions to a promise problem $\Pi \in \mathrm{MIP}_{1,\epsilon}[3,(1+1/m(n))\log n + O(m(n)\log m(n)), m(n)^{O(1)}n^{O(1/m(n))}]$.*

[8] But, the assertion does not hold for most (S,T)-sampled verifiers.

[9] Indeed, we have ignored the effect of sampling Ω_S; that is, the relation of $\omega(V|_{S,\Omega_S})$ and $\omega(V|_{S,T})$, for a random $T \subseteq \Omega_S$ of size N_2. Here, we fix any choice of $P_1 : Q_1 \to A$ and $P_2 : S \to A$. Again, applying Chernoff bounds, we see that the probability that the restrictions of Ω_S to T lead to acceptance with probability more than $\omega(V|_{S,\Omega_S}) + (\epsilon/2)$ is $\exp(-\epsilon N_2)$. Taking the union bound over all choices of P_1 and P_2, we infer that $\omega(V|_{S,T}) > \omega(V|_{S,\Omega_S}) + (\epsilon/2)$ with probability at most $\exp(-\epsilon N_2)\cdot|A|^{|Q_1|+|S|}$. Thus, using $N_2 = \frac{c_2}{\epsilon}(|S|\log|A| + \log(1/\gamma))$, we conclude that $\omega(V|_{S,T}) \leq \omega(V|_{S,\Omega_S}) + (\epsilon/2)$ with probability at least $1-\frac{\gamma}{2}$ (over the choices of T).

Before proving this lemma, let us see some special cases obtained by setting $m(n) = \text{poly}(\log\log n)$ and $m(n) = \sqrt{\log n}$, respectively in the above lemma.

Corollary 4.6 *For every $\mu > 0$ and every polynomial p, there exists a promise problem $\Pi \in \text{MIP}_{1,\mu}[3, (1+1/p(\log\log n))\cdot\log n, 2^{\text{poly}(\log\log n)}]$ such that SAT reduces probabilistically to Π under $n^{1+(1/p(\log\log n))}$-length preserving reductions.*

Corollary 4.7 *For every $\mu > 0$, there exists a promise problem $\Pi \in \text{MIP}_{1,\mu}[3, (1 + O((\log\log n)/\sqrt{\log n}))\cdot\log n, 2^{O(\sqrt{\log n}\log\log n)}]$, such SAT reduces probabilistically to Π under $n^{1+O((\log\log n)/\sqrt{\log n})}$-length preserving reductions.*

We defer the proof of Lemma 4.5 to Section 4.2.4. Here we give an overview of the proof steps. We modify the proof of [15] improving it in two steps. The proof of [15] first reduces SAT to a parametrized problem they call GapPCS under $\ell'(n)$-length preserving reductions for $\ell'(n) = n^{1+\gamma}$ for any $\gamma > 0$. Then they give a 3-prover MIP proof system for the reduced instance of GapPCS where the verifier tosses $(3+\gamma)\log\ell'(n)$ random coins.

Our first improvement shows that the reduction of [15] actually yields a stronger reduction than stated there, in two ways. First we note that their proof allows for smaller values of $\ell(n)$ than stated there, allowing in particular for the parameters we need. Furthermore, we notice that their result gives rise to instances from a restricted class, for which slightly more efficient protocols can be designed. In particular, we can reduce the size of the smallest prover in their MIP protocol to roughly $\ell(n)$ (as opposed to their result which gives a prover of size $\ell(n)^{1+\gamma}$ for arbitrarily small γ). These improvements are stated formally in Lemmas A.2 and A.3 and Corollary A.4.

The second improvement is more critical to our purposes. Here we improve the randomness complexity of the MIP verifier of [15], by applying a random truncation. To get this improvement we need to abstract the verifier of [15] (or the one obtained from Corollary A.4). This is done in Section 4.2.1. We then show how to transform such a verifier into one with $(1+o(1))\log n$ randomness. This transformation comes in three stages, described in Sections 4.2.2-4.2.4.

4.2.1 Abtracting the verifier of Corollary A.4

The verifier of Corollary A.4 interacts with three provers which we'll denote P, P_1, and P_2. We will let Q, Q_1, and Q_2 denote the question space of the provers respectively; and we'll let A, A_1, and A_2 denote the space of answers of the provers respectively. Denote by $V_x(r, a, a_1, a_2)$, the acceptance predicate of the verifier on input x, where r denotes the verifier's coins, and a (resp., a_1, a_2) the answer of prover $f = P$ (resp., P_1, P_2). (Note: The value of V_x is 1 if the verifier accepts.) We'll usually drop the subscript x unless needed. Let us denote by $q(r)$, (resp. $q_1(r)$, $q_2(r)$) the verifier's query to P (resp., P_1, P_2) on random string r. We note that the following properties hold for the 3-prover proof system given by Corollary A.4.

1. The acceptance-predicate decomposes: $V(r, a, a_1, a_2) = V_1(r, a, a_1) \wedge V_2(r, a, a_2)$, where V_1 and V_2 are predicates.

2. Sampleability: The verifier only tosses $O(\log n)$ coins. Thus, it is feasible to sample from various specified subsets of the space of all possible coin outcomes. For example, given $S_1 \subseteq Q_1$, we can uniformly generate in poly(n)-time a sequence of coins r such that $q_1(r) \in S_1$.

3. Uniformity: The verifier's queries to prover P (resp. P_1, P_2) are uniformly distributed over Q (resp. Q_1, Q_2).

4. If x is a NO-instance, then for $V = V_x$, for small ϵ and every possible P strategy, there exists a subset $Q' = Q'_P \subseteq Q$ such that for every P_1, P_2 the following two conditions holds

$$\Pr_r[q(r) \in Q' \wedge V_1(r, f(Q(r)), P_1(Q_1(r)))] < \frac{\epsilon}{2} \quad (2)$$

$$\Pr_r[q(r) \notin Q' \wedge V_2(r, f(Q(r)), P_2(Q_2(r)))] < \frac{\epsilon}{2} \quad (3)$$

4.2.2 The 3-prover MIP: Stage I

We start by modifying the verifier of Corollary A.4 so that its questions to provers P_1 and P_2 are "independent" (given the question to the prover P). That is, we define a new verifier, denoted W, that behaves as follows

- On input x, let $V = V_x$ be the verifier's predicate and let V_1 and V_2 be as given in Property (1).
- Pick $q \in Q$ uniformly and pick coins r_1 and r_2 uniformly and independently from the set $\{r \in \Omega | q(r) = q\}$. [Here we use sampleability with respect to a specific set of r's.]
- Make queries q (which indeed equals $q(r_1) = q(r_2)$), $q_1 = q_1(r_1)$ and $q_2 = q_2(r_2)$, to P, P_1 and P_2, receiving answers $a = P(q)$, $a_1 = P_1(q_1)$ and $a_2 = P_2(q_2)$.
- Accept if and only if $V_1(r_1, a, a_1) \wedge V_2(r_2, a, a_2)$.

Claim 4.8 *W has perfect completeness and soundness at most ϵ.*

Proof: The completeness is obvious, and so we focus on the soundness. Fix a NO-instance x and any set of provers P, P_1 and P_2. Let $Q' = Q'_P$ be the subset of Q as given by Property (4) of the MIP. The probability that W accepts is given by

$$\Pr_{q,r_1,r_2}[EV_1(r_1) \wedge EV_2(r_2)] \quad (4)$$

where $EV_1(r_1) = V_1(r_1, P(q), P_1(q_1(r_1)))$ and $EV_2(r_2) = V_2(r_2, P(q), P_2(q_2(r_2)))$. Note that $q = q(r_1) = q(r_2)$, where (q and) r_1, r_2 are selected as above. Thus, EV_i only depends on r_i, and the shorthand above is legitimate. Note that the process of selecting r_1 and r_2 in (4) is equivalent to selecting each of them uniformly (though not independently). We thus upper bound (4) by

$$\Pr_{r_1}[q(r) \in Q' \wedge EV_1(r_1)] + \Pr_{r_2}[q(r) \notin Q' \wedge EV_2(r_2)].$$

Using Property (4), each term above is bounded by $\epsilon/2$ and thus the sum above is upper-bounded by ϵ. ■

4.2.3 The 3-prover MIP: Stage II

In the next stage, the crucial one in our construction, we reduce the size of the provers P_1 and P_2 by a random truncation. For sets $S_1 \subseteq Q_1$ and $S_2 \subseteq Q_2$, we define the (S_1, S_2) restricted verifier W_{S_1,S_2} as follows:

- On input x, let $V = V_x$ be the verifier's predicate and let V_1 and V_2 be as given in Property (1).
- Pick $q \in Q$ uniformly and for $i \in \{1, 2\}$ pick coins r_i's uniformly and independently from the sets $\{r \in \Omega | q(r) = q \wedge q_i(r) \in S_i\}$. If either of the sets is empty, then the verifier simply accepts. [Here, again, we use sampleability of subsets of the verifier coins.]

- Make queries $q = q(r_1) = q(r_2)$, $q_1 = q_1(r_1)$ and $q_2 = q_2(r_2)$, to P, P_1 and P_2, receiving answers $a = P(q)$, $a_1 = P_1(q_1)$ and $a_2 = P_2(q_2)$.
- Accept if and only if $V_1(r_1, a, a_1) \wedge V_2(r_2, a, a_2)$.

As usual it is clear that the verifier W_{S_1,S_2} has perfect completeness (for every S_1 and S_2). We bound the soundness of this verifier, for most choices of sufficiently large sets S_1 and S_2:

Lemma 4.9 *For randomly chosen sets S_1, S_2 of size $O(|Q| \max\{\log |A|, \log |Q|\}$, with probability at least $4/5$, the soundness of the verifier W_{S_1,S_2} is at most 6ϵ.*

Proof: We start with some notation: Let Ω denote the space of random strings of the verifier V (of Section 4.2.1). For $i \in \{1, 2\}$ and a fixed set S_i, let A_i denote the distribution on Ω induced by picking a random string $r \in \Omega$ uniformly, conditioned on $q_i(r) \in S_i$ (i.e., uniform in $\{r \in \Omega | q_i(r) \in S_i\}$). Similarly, let B_i denote the distribution on Ω induced by picking a query $q \in Q$ uniformly and then picking r_i uniformly at random from the set $\{r \in \Omega | q(r) = q \wedge q_i(r) \in S_i\}$. We use the notation $r_i \leftarrow D$ to denote that r_i is picked according to distribution D. Note that the verifier W_{S_1,S_2} picks $r_1 \leftarrow B_1$ and $r_2 \leftarrow B_2$ (depending on the same random $q \in Q$). In our analysis, we will show that, for a random S_i, the distributions A_i and B_i are statistically close, where as usual the statistical difference between A_i and B_i is defined as $\max_{S \subseteq \Omega} \{\Pr_{r_i \leftarrow A_i}[r_i \in S] - \Pr_{r_i \leftarrow B_i}[r_i \in S]\}$. We will then show that the verifier has low soundness error if it works with the distributions A_1 and A_2. This informal description is made rigorous below by considering the following "bad" events (over the probability space defined by the random choices of S_1, S_2):

BE1: The statistical difference between A_1 and B_1 is more than ϵ.

BE2: The statistical difference between A_2 and B_2 is more than ϵ.

BE3: There exist P, P_1 such that for $Q' = Q'_P$ (as in Property (4) of Section 4.2.1)

$$\Pr_{r_1 \leftarrow A_1}[(q(r_1) \in Q') \wedge V_1(r_1, P(q(r_1)), P_1(q_1(r_1)))] > 2\epsilon.$$

BE4: There exist P, P_2 such that for $Q' = Q'_P$ (as in Property (4) of Section 4.2.1)

$$\Pr_{r_2 \leftarrow A_2}[(q(r_2) \notin Q') \wedge V_2(r_2, P(q(r_2)), P_2(q_2(r_2)))] > 2\epsilon.$$

Below we will bound the probability of these bad events, when S_1, S_2 are chosen at random. But first we show that if none of the bad events occur, then the verifier W_{S_1,S_2} has small soundness. Let $(r_1, r_2) \leftarrow W_{S_1,S_2}$ denote a random choice of the pair (r_1, r_2) as chosen by the verifier W_{S_1,S_2}. Fix proofs P, P_1, P_2 and let Q' be as in Property (4). Then,

$$\begin{aligned}
&\Pr_{(r_1,r_2) \leftarrow W_{S_1,S_2}}[V_1(r_1, P(q(r_1)), P_1(q_1(r_1))) \wedge V_2(r_2, P(q(r_2)), P_2(q_2(r_2)))] \\
&\quad \leq \Pr_{r_1 \leftarrow B_1}[(q(r_1) \in Q') \wedge V_1(r_1, P(q(r_1)), P_1(q_1(r_1)))] \\
&\qquad + \Pr_{r_2 \leftarrow B_2}[(q(r_2) \notin Q') \wedge V_2(r_2, P(q(r_2)), P_2(q_2(r_2)))] \\
&\quad \leq \Pr_{r_1 \leftarrow A_1}[(q(r_1) \in Q') \wedge V_1(r_1, P(q(r_1)), P_1(q_1(r_1)))] + \epsilon \\
&\qquad + \Pr_{r_2 \leftarrow A_2}[(q(r_2) \notin Q') \wedge V_2(r_2, P(q(r_2)), P_2(q_2(r_2)))] + \epsilon \qquad [\neg\text{BE1 and } \neg\text{BE2}] \\
&\quad \leq 6\epsilon \qquad [\neg\text{BE3 and } \neg\text{BE4}]
\end{aligned}$$

Claim 4.10 *The probability of event BE1 (resp., BE2) is at most $1/20$.*

Proof: To estimate the statistical difference between A_i and B_i, note that sampling r_i according to A_i is equivalent to the following process: select $r'_i \leftarrow A_i$ (i.e., r'_i is selected uniformly in $\{r|q_i(r) \in S_i\}$), set $q = q(r_i)$, and pick r_i uniformly from the set $\{r|(q(r)=q) \wedge (q_i(r) \in S_i)\}$. Thus, the statistical difference between A_i and B_i equals $\frac{1}{2} \cdot \sum_{q \in Q} |\Pr_{r_i \leftarrow A_i}[q(r_i) = q] - \Pr_{r_i \leftarrow B_i}[q(r_i) = q]|$, which in turn equals $\frac{1}{2} \cdot \sum_{q \in Q} \left|\Pr_{r_i \leftarrow A_i}[q(r_i) = q] - \frac{1}{|Q|}\right|$. To bound this sum, we bound the contribution of each of its terms (for a random S_i). Fixing an arbitrary $q \in Q$, we consider the random variable

$$\Pr_{r \leftarrow A_i}[q(r) = q] = \frac{|\{r|(q(r)=q) \wedge (q_i(r) \in S_i)\}|}{|\{r|q_i(r) \in S_i\}|}$$

(as a function of the random choice of S_i). The expectation of this quantity is $\frac{1}{|Q|}$. A simple application of Chernoff bounds shows that, with probability at least $\exp(-\epsilon|S_i|/|Q|)$, this random variable is in $(1 \pm \epsilon)\frac{1}{|Q|}$. Thus, for $|S_i| = c \cdot |Q| \log |Q|$ (where $c = O(1/\epsilon)$), the probability that $\Pr_{r \leftarrow A_i}[q(r) = q]$ is not in $[(1 \pm \epsilon)\frac{1}{|Q|}]$ is at most $\frac{1}{20|Q|}$. By the union bound, the probability that such a q exists is at most $\frac{1}{20}$, and if no such q exists then the statistical difference is bounded by at most ϵ. ∎

Claim 4.11 *The probability of event BE3 (resp., BE4) is at most* 1/20.

Proof: We will bound the probability of the event BE3. The analysis for BE4 is identical. Both proofs are similar to the proof of Lemma 4.3.

Fix P and let Q' be the set as given by Propery (4) of Section 4.2.1. We will show that

$$\Pr_{S_1}\left[\exists P_1 \text{ s.t. } \Pr_{r_1 \in A_1}[(q(r_1) \in Q') \wedge V_1(r_1, P(q(r_1)), P_1(q_1(r_1)))] \geq 2\epsilon\right] \leq \frac{1}{20}|A|^{-|Q|} \qquad (5)$$

The claim will follow by a union bound over the $|A|^{|Q|}$ possible choices of P. For fixed P (and thus fixed Q'), note that there is an optimal prover $P_1 = P_1^*$ that maximizes the quantity $\epsilon_{q_1} \stackrel{\text{def}}{=} \Pr_{r|q_1(r)=q_1}[(q(r) \in Q') \wedge V_1(r, P(q(r)), P_1(q_1))]$ for every $q_1 \in Q_1$. Furthermore $\mathbf{E}_{q_1 \in Q_1}[\epsilon_{q_1}] = \epsilon$. Applying Chernoff bounds, we get that the probability that when we pick $|S_1|$ elements from Q_1 uniformly and independently, their average is more than twice the expectation is at most $\exp(-|S_1|)$. Thus if $|S_1| \geq c \cdot |Q| \log |A|$ for some large enough constant c, then this probability is at most $\frac{1}{20}|A|^{-|Q|}$ as claimed in Equation (5). The claim follows. ∎

Lemma 4.9 follows now since we have that some bad event (i.e., one of the four BEi's) occurs with probability at most 4/20, and otherwise the soundness is indeed as claimed. ∎

4.2.4 The 3-prover MIP: Stage III

Having reduced the sizes of the three prover oracles, it is straightforward to reduce the amount of randomness used by the three provers. Below we describe a reduced randomness verifier $W_{S_1,S_2,T}$ where $S_i \subseteq Q_i$ and $T \subseteq \{(r_1, r_2)|(q(r_1) = q(r_1)) \wedge (q_i(r_i) \in S_i, \forall i \in \{1, 2\})\}$.

- On input x, let $V = V_x$ be the verifier's predicate and let V_1 and V_2 be as given in Property (1).
- Pick $(r_1, r_2) \in T$ uniformly at random. [This uses the sampleability property.]
- Compute $q = q(r_1) = q(r_2)$, and make queries q, $q_1 = q_1(r_1)$ and $q_2 = q_2(r_2)$, to P, P_1 and P_2, receiving answers $a = P(q)$, $a_1 = P_1(q_1)$ and $a_2 = P_2(q_2)$.

- Accept if and only if $V_1(r_1, a, a_1) \wedge V_2(r_2, a, a_2)$.

It is obvious that the verifier uses $\log_2 |T|$ random bits. It is also easy to see (as in the second part of the proof of Lemma 4.3) that if T is chosen randomly of sufficiently large size then its soundness remains low. We skip this proof, stating the resulting lemma.

Lemma 4.12 *If S_1, S_2 are chosen randomly of size $O(|Q| \max\{\log |A|, \log |Q|\}$ and T is chosen randomly of size $O(|Q| \log |A| + |S_1| \log |A_1| + |S_2| \log |A_2|)$, then with probability at least $\frac{2}{3}$ that $W_{S_1,S_2,T}$ has soundness at most 7ϵ.*

Using Lemma 4.12, we now prove Lemma 4.5.

Proof [of Lemma 4.5]: Fix $\epsilon' = \epsilon/7$. Let V be the 3-prover verifier for SAT as obtained from Corollary A.4. In particular, V has perfect completeness and soundness ϵ'. The size of the smallest prover is $\ell'(n) = m(n)^{O(m(n))} \cdot n^{1+O(1/m(n))}$, the answer length is bounded by $m(n)^{O(1)} \cdot n^{O(1/m(n))}$, and V satisfies the properties listed in Section 4.2.1. For sets S_1, S_2, T, let $W_{S_1,S_2,T}$ be the verifier obtained by modifying V as described in the current section. Consider the promise problem Π whose instances are tuples (ϕ, S_1, S_2, T) where an instance is a YES-instance if $W_{S_1,S_2,T}$ accepts ϕ with probability one, and the instance is a NO-instance if $W_{S_1,S_2,T}$ accepts with probability at most ϵ. We note that an instance of Π of size N has a 3-prover proof system using at most $\log_2 N$ random coins, perfect completeness and soundness error $7\epsilon' = \epsilon$ (since $W_{S_1,S_2,T}$ is such a verifier). Now, consider the reduction that maps an instance ϕ of SAT of length n to the instance (ϕ, S_1, S_2, T), where S_1, S_2 are random subsets of queries of V of size $O(\ell'(n) \cdot n^{O(1/m(n))})$ and T is a random subset of size $O(\ell'(n) \cdot n^{O(1/m(n))}) \cdot n^{O(1/m(n))} = \ell(n)$ of the random strings used by the verifier W_{S_1,S_2}. This reduction always maps satisfiable instances of SAT to YES-instances of Π and, by Lemma 4.12, with probability at least $\frac{2}{3}$, it maps unsatisfiable instances of SAT to NO-instances of Π. ∎

4.3 Nearly linear PCPs

Applying state-of-the-art composition lemmas to the MIP constructed in the previous subsection gives our final results quite easily. In particular, we use the following lemmas.

Lemma 4.13 (cf. [3] or [5, 18]) *For every $\mu_1 > 0$ and $p < \infty$, there exists $\mu > 0$ and constants c_1, c_2, c_3 such that for every $r, a : \mathsf{Z}^+ \to \mathsf{Z}^+$,*

$$\mathrm{MIP}_{1,\mu}[p, r, a] \subseteq \mathrm{MIP}_{1,\mu_1}[p + 3, r + c_1 \log a, c_2 (\log a)^{c_3}].$$

We apply the lemma above repeatedly till the answer lengths become $\mathrm{poly} \log \log \log n$. Then to terminate the recursion, we use the following result of [15].

Lemma 4.14 (Lem. 2.6 in [15]) *For every $\epsilon > 0$ and $p < \infty$, there exists a $\gamma > 0$ such that for every $r, a : \mathsf{Z}^+ \to \mathsf{Z}^+$,*

$$\mathrm{MIP}_{1,\gamma}[p, r, a] \subseteq \mathrm{PCP}_{1,\frac{1}{2}+\epsilon}[r + O(2^{pa}), p + 7].$$

Combining the above lemmas with the nearly-linear 3-IP obtained in the previous subsection, we obtain:

Theorem 4.15 (Our main PCP result):

1. *For every $\epsilon > 0$, SAT reduces probabilistically, under $n^{1+O(1/\log\log n)}$-length preserving reductions to a promise problem $\Pi \in \mathrm{PCP}_{1,\frac{1}{2}+\epsilon}[(1+O(1/\log\log n))\cdot\log n, 16]$.*

2. *For every $\epsilon > 0$, SAT reduces probabilistically, under $n^{1+O(\sqrt{\log n}\log\log n)}$-length preserving reductions to a promise problem $\Pi \in \mathrm{PCP}_{1,\frac{1}{2}+\epsilon}[(1+O(\log\log n/\sqrt{\log n}))\cdot\log n, 19]$.*

Part 2 implies Theorem 2.3.

Proof: The first part is obtained by starting with Corollary 4.6, and applying Lemma 4.13 twice to get a 9-prover MIP system with answer lengths $\mathrm{poly}(\log\log\log n)$. Applying Lemma 4.14 to this 9-prover proof system, gives the desired 16-bit PCP. For the second part, we start with Corollary 4.7 and apply Lemma 4.13 thrice, obtaining a 12-prover MIP system with answer lengths $\mathrm{poly}(\log\log\log n)$. Applying Lemma 4.14 gives the 19-bit PCP. ■

5 Nearly-linear-sized codes from PCPs

Here we augment the results of Section 3 by constructing nearly-linear-sized locally-testable codes. We do so by starting with the randomly truncation of the FS-code from Section 3.2, and applying PCP techniques to reduce the alphabet size (rather than following the paradigm of concatenated codes as done in the rest of Section 3). Specifically, in addition to encoding individual alphabet symbols via codewords of smaller alphabet, we also augment the new codewords with small PCPs that allow to emulate the local-tests of the original code.

5.1 Problems with using a PCP directly

Before turning to the actual constructions, we explain why merely plugging-in a standard (inner-verifier) PCP will not work. We start with the most severe problem, and then turn to additional ones.

Non-uniqueness of the encoding: As discussed in the Introduction, the soundness property of standard PCPs does not guarantee unique encodings of witnesses, but rather that PCP oracles accepted with high probability can be decoded into some witnesses. Indeed, current PCPs tend to do exactly this, due to a gap between the canonical oracles (used in the completeness condition) that encodes information as polynomials of some given individual degree, and the soundness condition that refers to the total degree of the polynomial.[10]

Linearity: We wish the resulting code to be linear, and it is not clear whether this property holds when composing a linear code with a standard inner-verifier. Since we start with a linear code (and a linear codeword test), there is hope that the proof oracle added to the concatenated code will also be linear. Indeed, with small modifications of standard constructions, this is the case.

[10] In basic constructions of codes, this is not a real problem since we can define the code to be the collection of all polynomials of a given total degree as opposed to polynomials of specified individual degree bound. However, when using such a code as the inner code in composition, we cannot adopt the latter solution because we only know how to construct adequate inner-verifiers for inputs encoded as polynomials of individually-bouded degree (rather than bounded total degree).

Other technical problems: Other problems arise in translating some of the standard "complexity-theoretic tricks" that are used in all PCP constructions. For example, PCP constructions are typically described in terms of a dense collection of input lengths (e.g., the input length must fit $|H|^m$ for some suitable sizes of $|H|$ and m (i.e., $m = \Theta(|H|/\log|H|)$), and are extended to arbitrary lengths by padding (of the input). In our context, such padding, depending on how it is done, either permits multiple encodings (of the same information), or forces us to check for additional conditions on the input (e.g., that certain bits of the input are zeroes). Other complications arise when one attempts to deal with "auxiliary variables" that are introduced in a process analogous to the standard reduction of verification of an arbitrary computation to the satisfiability of a 3CNF expression.

This forces us to rework the entire PCP theory, while focusing on unique encodings and on obtaining "linear PCP oracles" when asked to verify homogenous linear conditions on the input. For the purposes of constructing short locally testable codes, it suffices to construct verifiers verifying systems of homogenous linear equations and this is all we'll do (though we could verify affine equations equally easily). In what follows, whenever we refer to a linear system, it will be implied that the constraints are homogenous.

5.2 Inner verifiers for linear systems: Definition and composition

We use PCP techniques to transform linear locally testable codes over large alphabet into ones over smaller alphabet. Specifically, we adapt the construction of inner-verifiers suct that using it to test linear conditions on the input-oracles will result in testing linear conditions on the proof oracle.

The basic ingredient of our proofs is the notion of an inner verifier for linear codes. A $(p,\ell) \to (p',\ell')$ inner verifier is designed to transform an F-linear code over an alphabet $\Sigma = F^\ell$ *that is testable by p queries*, into an F-linear code (of a typically longer size) over an alphabet $\Sigma' = F^{\ell'}$ *that is testable by p' queries*, where typically $\ell' \ll \ell$ (but $p' > p$). Informally, the inner-verifier allows to emulate a local test in the given code over Σ, by providing an encoding (over Σ') of each symbol in the original codeword as well as auxiliary proofs (of homogenous linear conditions) that can be verified based on a constant number of queries.

Verifying that a vector satisfies a conjunction of (homogenous) linear conditions is equivalent to verifying that it lies in some linear subspace (i.e., the space of vectors that satisfy these conditions). For integer m and field F, we let $\mathcal{L}_{F,m}$ denote the set of all linear subspaces of F^m. We'll assume that such a subspace $L \in \mathcal{L}_{F,m}$ is specified by a matrix $M \in F^{m\times m}$ such that $L = \{x \in F^m | Mx = \vec{0}\}$. According to convenience, we will sometimes say that *a vector lies in L* and sometimes say that *it satisfies the conditions L*.

Definition 5.1 *For a field F, and positive integers p, ℓ, p', ℓ', and positive reals δ and γ, a* $(F, (p,\ell) \to (p',\ell'), \delta, \gamma)$-linear inner verifier *consists of a triple* $(E, P, \text{Verdict})$ *such that*

- $E : F^\ell \to (F^{\ell'})^n$ *is an* F-linear code of minimum distance at least δn *over the alphabet* $F^{\ell'}$.
- $P : \mathcal{L}_{F,p\ell} \times (F^\ell)^p \to (F^{\ell'})^N$, *is* a proving function *that satisfies the completeness condition below.*
- Verdict *is an* oracle machine *that gets as input* $L \in \mathcal{L}_{F,p\ell}$ *and* (coins) $R \in \{0,1\}^r$ *and has oracle access to* $p+1$ *vectors, denoted* $X_1, \ldots, X_p \in (F^{\ell'})^n$ *and* $X_{p+1} \in (F^{\ell'})^N$, *such that each oracle call is answered by one* $F^{\ell'}$*-coordinate of the corresponding oracle vector.*[11] *Machine* Verdict *satisfies the following properties:*

[11] That is, query $j \in [n]$ (resp., $j \in [N]$) to oracle $i \in [p]$ (resp., $i = p+1$) is answered by the j^{th} element of X_i.

Queries and Linearity: *For every choice of* $L \in \mathcal{L}_{F,p\ell}$ *and* $R \in \{0,1\}^r$*, machine* Verdict *makes at most* p' *oracle calls to the oracles* $X_1, \ldots, X_{p+1}$*. Furthermore, for every* R *and* L*, the acceptance condition of* Verdict *is a conjunction of* F*-linear constraints on the responses to the queries.*

Completeness: *If the* p *first oracles encode a* p*-tuple of vectors over* F^ℓ *that satisfies* L *and if* X_{p+1} *is selected adequately then* Verdict *always accepts.*

That is, for every $x_1, \ldots, x_p \in F^\ell$ *and* $L \in \mathcal{L}_{F,p\ell}$ *such that* $(x_1, \ldots, x_p) \in L$*, and for every* $R \in \{0,1\}^r$*, it holds that* $\text{Verdict}(L, R, E(x_1), \ldots, E(x_p), P(L, x_1, \ldots, x_p)) = 1$.

Augmented Soundness: *If the* p *first oracles are far from encoding a* p*-tuple of vectors over* F^ℓ *that satisfies* L *then* Verdict *rejects for most choices of* R*, no matter which* X_{p+1} *is used. Furthermore, if the* p *first oracles encode a* p*-tuple that satisfies* L *but* X_{p+1} *is far from the unique proof determined by* P *then* Verdict *rejects for most choices of* R*.*

Formally, for $X_1, \ldots, X_p \in (F^{\ell'})^n$*,* $X_{p+1} \in (F^{\ell'})^N$*,* $L \in \mathcal{L}_{F,p\ell}$*, and* $(x_1, \ldots, x_p) \in L$*, let* $\epsilon(X_1, \ldots, X_{p+1}, L, x_1, \ldots, x_p)$ *denote the maximum distance of* X_i *from the corresponding adequate encoding (i.e.,* $E(x_i)$ *if* $i \leq p$ *and* $P(L, x_1, \ldots, x_p)$ *otherwise). That is,*

$$\epsilon(X_1, \ldots, X_{p+1}, L, x_1, \ldots, x_p) = \max\left\{ \max_{i \in [p]} \left\{ \frac{\Delta(X_i, E(x_i))\}}{n} \right\} ; \frac{\Delta(X_{p+1}, P(L, x_1, \ldots, x_p))}{N} \right\}$$

Then, for every $X_1, \ldots, X_p \in (F^{\ell'})^n$ *and* $\Pi \in (F^{\ell'})^N$*,*

$$\Pr_R[\text{Verdict}(L, R, X_1, \ldots, X_p, \Pi) = 0] \geq \gamma \cdot \min\left\{ \delta/2, \min_{(x_1, \ldots, x_p) \in L} \{\epsilon(X_1, \ldots, X_{p+1}, L, x_1, \ldots, x_p)\} \right\}$$

Such a verifier is said to use r *coins, encodings of length* n *and proofs of length* N*.*

Typically, we aim at having N, n and 2^r be small functions of ℓ (i.e., polynomial or even almost-linear in ℓ). Definition 5.1 is designed to suit our applications. Firstly, the augmented notion of soundness that refers also to "non-canonical" proofs of valid statements fits our aim of obtaining a code that is locally checkable (because it guarantees rejection of sequences that are not obtained by the unique coding transformation). Indeed, this augmentation of soundness is non-standard (and arguablly unnatural) in the context of PCP. Secondly, Definition 5.1 only handles the verification of linear conditions, and does so while only utilizing linear tests. Indeed, this fits our aim of transforming linear codes over large alphabet (i.e., the alphabet F^ℓ) to linear codes over smaller alphabet (i.e., $F^{\ell'}$).

The utility of linear inner verifiers in constructing locally-testable codes is demonstrated by the following two propositions, which follow immediately from Definition 5.1. The first proposition merely serves as a warm-up towards the second one.

Proposition 5.2 *A* $(F, (1, \ell) \to (p', \ell'), \delta, \gamma)$*-linear inner verifier implies the existence of a linear locally-testable code of relatoive distance at most* $\delta/2$ *over the alphabet* $\Sigma = F^{\ell'}$ *mapping* $F^\ell = \Sigma^{\ell/\ell'}$ *to* Σ^m *for* $m = O(p' \cdot (n + N))$*, where* n *and* N *are the corresponding lengths of the encoding and the proof used by the verifier. Specifically, the code is testable with* p' *queries, with the rejection probability of a word at distance* ϵ *from any codeword being at least* $\Omega(\gamma \cdot \epsilon)$*.*

Proof: Let $V = (E, P, \text{Verdict})$ be the $(F, (1, \ell) \to (p', \ell'), \delta, \gamma)$-linear inner verifier, where $E : F^\ell \to (F^{\ell'})^n$ and $P : \mathcal{L}_{F,\ell} \times F^\ell \to F^N$. Below we assume that $n < N$ (which is typically the

case).[12] The locally testable encoding E' of a string $x \in \Sigma^{\ell/\ell'} \cong F^\ell$ equals the $(\lceil N/n \rceil + 1)$-long sequence $(E(x), \ldots, E(x), P(L, x))$, where $L = F^\ell$ (i.e., L is satisfied by every vector) and $E(x)$ is replicated $\lceil N/n \rceil$ times. The relative distance of the code given by this encoding is at least $\delta/2$. To test a potential codeword $(X_1, \ldots, X_{\lceil N/n \rceil}, Y)$, we perform at random one out of two kinds of tests: With probability $\frac{1}{2}$ we test that the N/n strings X_j's are replications. We do so by picking a random index $i \in [n]$, and two distinct indices $j_1, j_2 \in [N/n]$, and testing that $(X_{j_1})_i = (X_{j_2})_i$. With the remaining probability we pick a random test as per the verifier V, replacing calls to the first oracle with corresponding probes to one of the first N/n oracles (i.e., one of the X_i's), selected at random. (Oracle calls to the proof oracle of V are replaced by corresponding probes to Y.) It can be verified that words at distance ϵ from codewords are rejected with probability $\Omega(\gamma\epsilon)$. ■

The following proposition will be used to compose locally testable codes over large alphabets with suitable linear inner verifiers to obtain locally testable codes over smaller alphabets. Specifically, given a q-query testable F-linear code over the alphabet $\Sigma = F^b$, we wish to use an adequate encoding (over $\Sigma' = F^a$) and an inner-verifier in order to emulate the local conditions checked by the test. The latter conditions are subspaces of $F^{q \cdot b}$, and so we need a $(F, (q, b) \to (p, a), \delta, \gamma)$-linear inner verifier in order to verify them.

Proposition 5.3 (composing an outer code with an inner-verifier):

- *Let C be a locally testable F-linear code over the alphabet $\Sigma = F^b$ mapping Σ^K to Σ^N, and suppose that the codeword test uses R coins and q queries.*
- *Let $V = (E, P, \text{Verdict})$ be a $(F, (q, b) \to (p, a), \delta, \gamma)$-linear inner verifier, where $E : F^b \to (F^a)^n$ and $P : \mathcal{L}_{F, q \cdot b} \times (F^b)^q \to (F^a)^m$.*

Then, there exists a locally testable code over the alphabet $\Sigma' = F^a$ mapping $\Sigma^K \equiv \Sigma'^{b \cdot K/a}$ to Σ'^M for $M = O(Nn + 2^R m)$. Furthermore, the resulting code has distance at least $\delta n D$, where D is the distance of C.

Proof: The new code consists of two parts (which are properly balanaced). The first part is obtained by encoding each Σ-symbol of the codeword of C by the code E, whereas the second part is obtained by providing proofs (for the inner-verifier) for the validity of each of the 2^R possible checks that may be performed by the codeword test. Specifically, $x \in \Sigma^K$ is encoded by the sequence

$$\Big(E(y_1), \ldots, E(y_N); P(L_{0^R}, y_{i_{0^R,1}}, \ldots, y_{i_{0^R,q}}), \ldots, P(L_{1^R}, y_{i_{1^R,1}}, \ldots, y_{i_{1^R,q}})\Big)$$

where $y_1 \cdots y_N = C(x)$, and for every $\omega \in \{0,1\}^R$, on coins ω, the codeword test (for C) probes locations $i_{\omega,1}, ..., i_{\omega,q}$ and verifies the linear condition L_ω. Indeed, as in the proof of Proposition 5.3, the above should be modified such that the two parts of the new codeword (i.e., the E-part and the P-part) have about the same length.[13]

Testing the new code is done by emulating the codeword test of C. That is, to test a potential codeword $(X_1, ..., X_N; Y_{0^R}, ..., Y_{1^R})$, we select uniformly $\omega \in \{0,1\}^R$, determine the corresponding condition $(i_{\omega,1}, ..., i_{\omega,q}, L_\omega)$ checked by the original codeword test, and invoke the inner-verifier V on input L_ω while providing V with (coins and) oracle access to $X_{i_{\omega,1}}, ..., X_{i_{\omega,q}}$ and Y_ω. ■

[12] Otherwise, one can augment $P(L, x)$ with $E(x)$ and maintain the soundness by testing consistency between the two copies of $E(x)$ (as done below).

[13] As before, the modification is via replication, and the new codeword test should check that the replication is proper.

Whereas Proposition 5.3 refers to the composition of an outer *code* with an inner-verifier yielding a new *code*, the following lemma refers to composing two inner-verifiers yielding a new inner-verifier. Indeed, we could have worked only with Proposition 5.3 (or alternatively only with Lemma 5.4 and Proposition 5.2), but it seems more convenient to (have and) work with both.[14]

Lemma 5.4 (composition of linear inner-verifiers): *Let $\gamma_1, \gamma_2 \leq 1$. Given a $(F, (p, \ell) \to (p', \ell'), \delta_1, \gamma_1)$-linear inner verifier and a $(F, (p', \ell') \to (p'', \ell''), \delta_2, \gamma_2)$-linear inner verifier, it is possible to construct a $(F, (p, \ell) \to (p'', \ell''), \delta_1\delta_2, \gamma_1\gamma_2\delta_2/6)$-linear inner verifier. Furthermore, if the i^{th} given verifier uses r_i coins, encoding length n_i and proof length N_i, then the resulting inner verifier uses $r_1 + r_2$ coins, encoding length $n_1 \cdot n_2$ and proof length $N_1 \cdot n_2 + 2^{r_1} \cdot N_2$.*

Proof: We start with the construction. Given a $(F, (p, \ell) \to (p', \ell'), r_1, \delta_1, \gamma_1)$ inner verifier $V_1 = (E_1, P_1, \text{Verdict}_1)$ and a $(F, (p', \ell') \to (p'', \ell''), r_2, \delta_2, \gamma_2)$ inner verifier $V_2 = (E_2, P_2, \text{Verdict}_2)$, we define their composition $V_1 \otimes V_2 = (E, P, \text{Verdict})$ as follows

- $E : F^\ell \to (F^{\ell''})^{n_1 \cdot n_2}$ is the concatenation of the encoding functions $E_1 : F^\ell \to (F^{\ell'})^{n_1}$ and $E_2 : F^{\ell'} \to (F^{\ell''})^{n_2}$. That is, $E(x_1, \ldots, x_\ell) = (E_2(y_1), \ldots, E_2(y_{n_1}))$, where $(y_1, \ldots, y_{n_1}) \stackrel{\text{def}}{=} E_1(x_1, \ldots, x_\ell)$.

- $P = (P^{(1)}, P^{(2)})$ is obtained as follows: Given $L, x_1, \ldots, x_p$, the first part of the proof (i.e., $P^{(1)}(L, x_1, \ldots, x_p)$) is the symbol-by-symbol encoding under E_2 of $P_1(L, x_1, \ldots, x_p)$. That is, $P^{(1)}(L, x_1, \ldots, x_p) = (E_2(y_1), \ldots, E_2(y_{N_1}))$, where $(y_1, \ldots, y_{N_1}) \stackrel{\text{def}}{=} P_1(L, x_1, \ldots, x_p)$. The second part of the proof (i.e., $P^{(2)}(L, x_1, \ldots, x_p)$) consists of 2^{r_1} blocks corresponding to each of the 2^{r_1} possible checks of Verdict_1. For each $R_1 \in \{0,1\}^{r_1}$, the block corresponding to R_1 in $P^{(2)}(L, x_1, \ldots, x_p)$ is the value $P_2(L_{R_1}, z_1, \ldots, z_{p'})$, where $z_1, \ldots, z_{p'}$ denote the p' coordinates of $E_1(x_1), \ldots, E_p(x_p)$ and $P_1(L, x_1, \ldots, x_p)$ that are inspected by $\text{Verdict}_1(L, R_1, \ldots)$ and L_{R_1} is the linear conjunction of F-linear conditions checked by Verdict_1.

  Note that the proof length is $N_1 \cdot n_2 + 2^{r_1} \cdot N_2$, where the first (resp., second) term corresponds to $P^{(1)}$ (resp., $(P^{(2)})$).

- $\text{Verdict}(L, (R_1, R_2), X_1, \ldots, X_p, \Pi)$ is computed as follows: Let $q_1, \ldots, q_{p'}$ be the queries that the function $\text{Verdict}_1(L, R_1, \ldots)$ makes into its oracles $X'_1, \ldots, X'_p, \Pi'$ on randomness R_1, and let L' denote the conjunction of linear equations it needs to verify on its responses. Then Verdict now applies the function $\text{Verdict}_2(L', R_2, \ldots)$ on input L' to the sub-oracles corresponding the E_2-encodings of the p' queries determined by Verdict_1. That is, if the jth query (i.e., q_j) of Verdict_1 is to X'_i then Verdict identifies the jth oracle of Verdict_2 (to be denoted X''_j) with block q_j of X_i (which supposedly encodes the corresponding symbol of X'_i). Otherwise (i.e., the jth query of Verdict_1 is to Π'), Verdict identifies the jth oracle of Verdict_2 (i.e., X''_j) with block q_j of the first part of $\Pi = (\Pi^{(1)}, \Pi^{(2)})$ (which supposedly encodes the corresponding symbol of Π'). Finally, Verdict identifies the proving oracle of Verdict_2 (to be denoted Π'') with the block of $\Pi^{(2)}$ that corresponds to R_1, invokes $\text{Verdict}_2(L', R_2, X''_1, ..., X''_{p'}, \Pi'')$, and Verdict accepts if and only Verdict_2 accepts.

We now argue that the composition satisfies the required properties. The main issue is the (augmented) soundness requirement. Suppose that $X_1, \ldots, X_p$ and $X_{p+1} = (\Pi^{(1)}, \Pi^{(2)})$ are $p+1$ oracles that are rejected by $V_1 \otimes V_2(L, \cdot, \ldots)$ with probability $(\gamma_1\gamma_2\delta_2/6) \cdot \epsilon$, where $\epsilon \leq \delta_1\delta_2$.

[14] An analogous comment may apply to the design of PCP system.

We need to show that there exist vectors $(x_1, \ldots, x_p) \in L$ such that $\Delta(E(x_i), X_i) \leq \epsilon n_1 n_2$ and $\Delta(P(L, x_1, \ldots, x_p), X_{p+1}) \leq \epsilon N$, where $N \stackrel{\text{def}}{=} N_1 \cdot n_2 + 2^{r_1} \cdot N_2$.

Let D_2 denote a unique decoding function for the inner encoding function E_2 (i.e., $D_2(X) = x$ if $\Delta(E_2(x), X) \leq (\delta_2/2) \cdot n_2$ and arbitrary otherwise). Applying this function to each of the n_1 blocks of $X_i \in (F^{\ell''})^{n_2 \cdot n_1}$, for $i \in [p]$, we obtain corresponding $Y_i \in (F^{\ell'})^{n_1}$. Similarly, applying this function to each of the N_1 blocks of $\Pi^{(1)} \in (F^{\ell''})^{n_2 \cdot N_1}$, we obtain $Y_{p+1} \in (F^{\ell'})^{N_1}$.

For each R_1, let use denote by $p_2(R_1)$ the probability that on coins $(R_1, \cdot)$ verifier V rejects the above oracles, where the probability is taken over V_2's actions. Suppuse that $p_2(R_1) < \gamma_2\delta_2/2$, and consider the p' input oracles and the proof oracle (i.e., part of $\Pi^{(2)}$) determined by R_1. Then, by the (basic) soundness of V_2, these p' sub-oracles (which are blocks in $X_1, ..., X_p, \Pi^{(1)}$) are at relative distance at most $p_2(R_1)/\gamma_2$ from the E_2-encoding of the corresponding blocks in $Y_1, ..., Y_p, Y_{p+1}$. Furthermore, by the augmented soundness (of V_2), the corresponding part of $\Pi^{(2)}$, denoted Z_{R_1}, is at relative distance at most $p_2(R_1)/\gamma_2$ from the value obtained by applying P_2 to these p' blocks (of the Y_i's).

Next, let $p_1 \stackrel{\text{def}}{=} \Pr_{R_1}[p_2(R_1) \geq \gamma_2\delta_2/2]$ Since $\mathbf{E}_{R_1}[p_2(R_1)] = (\gamma_1\gamma_2\delta_2/6)\epsilon$, it follows that $p_1 \leq \gamma_1\epsilon/3$. Now, since $\epsilon \leq \delta_1\delta_2$, the Y_i's are p_1/γ_1-close to a valid encoding of a p-tuple, denoted $(x_1, ..., x_p)$, and a corresponding P_1-proof (i.e., $P_1(L, x_1, ..., x_p)$). We conclude that the Y_i's are at relative distance at most p_1/γ_1 from the corresponding $E_1(x_i)$'s (resp., $P_1(L, x_1, ..., x_p)$). Defining $\epsilon_2(R_1) \stackrel{\text{def}}{=} p_2(R_1)/\gamma_2$ if $p_2(R_1) < \gamma_2\delta_2/2$ and $\epsilon_2(R_1) \stackrel{\text{def}}{=} 1$ otherwise, recall that the Y_i's are at relative distance at most $\mathbf{E}_{R_1}[\epsilon_2(R_1)]$ from the corresponding blocks of the X_i's (resp., $\Pi^{(1)}$). Recall that, except for a p_1 fraction of the R_1's, it holds that $p_2(R_1) < \gamma_2\delta_2/2$, we obtain

$$\begin{aligned}
\frac{\Delta(E(x_i), X_i)}{n_1 n_2} &\leq \frac{\Delta_1(E_1(x_i), Y_i)}{n_1 \ell'} + \frac{\Delta_2(E_2(Y_i), X_i)}{n_1 n_2} \\
&\leq \frac{p_1}{\gamma_1} + \mathop{\mathbf{E}}_{R_1}[\epsilon_2(R_1)] \\
&\leq \frac{\epsilon}{3} + \left(\mathop{\mathbf{E}}_{R_1}[p_2(R_1)/\gamma_2] + p_1 \right) \;<\; \epsilon
\end{aligned}$$

using $\gamma_1, \gamma_2 \leq 1$. The same holds with respect to the distance of $\Pi^{(1)}$ from $P^{(1)}(L, x_1, ..., x_p)$. Finally, recall that for all but at most an p_1 fraction of the R_1's, the relative distance between Z_{R_1} (i.e., the corresponding block of $\Pi^{(2)}$) and the value obtained by applying P_2 to the relevant blocks of the Y_i's is at most $\epsilon_2(R_1)$. It follows that the relative distance between $\Pi^{(2)}$ from $P^{(2)}(L, x_1, ..., x_p)$ is at most $p_1 + \mathbf{E}_{R_1}[\epsilon_2(R_1)]$, which is bounded by ϵ (as shown above). ■

5.3 Linear inner verifiers: Two constructions

Throughout the rest of this section, $F_2 \stackrel{\text{def}}{=} GF(2)$. We start by presenting a linear inner verifier that corresponds to the inner-most verifier of Arora et al. [1]. Things are simpler in our context, since we only need to prove/verify linear conditions. Here these (linear) conditions refer to p elements of F_2^k, and are verified by a (random) linear test that depends on $p+1$ bits (at random locations).

Lemma 5.5 *There exists a $\gamma > 0$ such that for every pair of integers p, ℓ, there exists a $(F_2, (p, \ell) \to (p+1, 1), \frac{1}{2}, \gamma)$-linear inner verifier. Furthermore, the length of the encoding is 2^ℓ, the length of the proof is $2^{p\ell}$, and the randomness in use equals $2p\ell$.*

Proof: The encoding E is just the Hadamard encoding; and the proving function $P(L, x_1, \ldots, x_p)$ is also Hadamard encoding, this time of the vector $(x_1, \ldots, x_p)$. To check whether $X_1, \ldots, X_p \in F_2^{2^\ell}$

encodes a vector in the linear subspace L given by a matrix $M \in F_2^{p\ell \times p\ell}$, the verdict function uniformly selects $q_1, \ldots, q_p \in F_2^\ell$ and a (random) vector v orthogonal to the subspace L (i.e., a random linear combination of the rows of M), and verifies that $(X_1)_{q_1} \oplus \cdots (X_p)_{q_p} = (X_{p+1})_{(q_1,\ldots,q_p)\oplus v}$. The now standard analysis implies the soundness of this verifier. ■

The main result in this subsection is an adaptation of the intermediate inner-verifier of Arora et al. [1, Section 7]. Recall that the latter uses significantly shorter encoding and proofs (and less randomness) than the simpler Hadamard-based verifier, but verification is based on (a constant number of) non-boolean answers.

Lemma 5.6 *There exists a $\gamma > 0$ such that for every pair of integers p, ℓ, there exists a $(F_2, (p,\ell) \to (p+3, \mathrm{poly}(\log p\ell)), \frac{1}{2}, \gamma)$-linear inner verifier. Furthermore, the lengths of the encoding and the proofs are $\mathrm{poly}(p\ell)$, and the randomness in use equals $O(p \log \ell)$.*

Our construction is a modification of an inner verifier given by Arora et al. [1] (Proof of Theorem 2.1.9, Section 7.5). We thus start by providing an overview of their proof and discuss the main issues that need to be addressed in adapting their to a proof of Lemma 5.6.

Overview of the proof of [1, Thm. 2.1.9]. We use the formalism of [15] to interpret the main steps in the proof of [1]. (In particular, whenever we refer to a step as "standard", such a step is performed explicitly in [15].) As a first step in their proof, Arora et al. [1] reduce SAT to a GapPCS problem (see Appendix for definition). Then, using a low-total-degree test, they give a 3-prover 1-round proof system for NP languages. Finally they observe that the proof system with slight modifications also works as proofs of properties of concatenated strings. Since the gap problem that is target of the reduction is critical, let us review the completeness and soundness condition of the reduction. Recall that an instance of GapPCS consists of a sequence of algebraic constraints on the values of a function $g : F^m \to F$. Each constraint is dependent on the value of g at (roughly) only polylogarithmically many inputs. The goal is to find a low-degree polynomial g that satisfies all or most constraints. In greater detail, the reduction consists of a pair of algorithms A and B, where A reduces instances of SAT to instances of GapPCS, and B takes as input an instance ϕ of SAT and an assignment a satisfying ϕ and produces a polynomial g that satisfies all constraints of $A(\phi)$. The properties of the reduction are as follows:

Completeness: If a is an assignment satisfying ϕ then $g = B(\phi, a)$ is a degree d bounded polynomial g that satisfies all constraints of $A(\phi)$.

Soundness: If ϕ is not satisfiable, then no total degree d bounded polynomial g satisfies even an ϵ fraction of the constraints of $A(\phi)$.

Since the soundness condition only focusses on degree d polynomials (and not arbitrary functions), constructing such a reduction turns out to be easier than constructing a full PCP. On the other hand, by combining this with a low-degree test it is easy to extend the soundness to all functions.

One would hope to use the above reduction directly to get a locally testable code by setting ϕ to be some formula enforcing the linear conditions L. But as noted earlier, several problems come up: First, B is not a linear map, but this is fixed easily. The more serious issue is that the soundness condition permits the existence of low-degree functions that satisfy all constraints that are not of the form $B(a)$ for any a. Indeed, in standard reductions the only functions of the form $g = B(a)$ have a bound of d/m in the degree of each variable, but this is not something that the low-degree test can test. Thus to apply the low-degree test and protocol of [1], we effectively augment the reduction from SAT to GapPCS so as to get the following soundness condition.

Modified Soundness: If g is a degree d polynomial that is not of the form $g = B(a)$ for some a satisfying ϕ, then g does not satisfy an ϵ fraction of the constraints of $A(\phi)$.

To obtain the modified soundness condition, we need to delve further into the reduction of [1] and the transformation B implied there. Say that their reduction produces a GapPCS instance on m variate polynomials.

1. The m-variant polynomial $g = B(a)$ in their transformation has the form $g(i, \vec{x}) = g_i(\vec{x})$, for $i \in [k]$, where the g_i's are polynomials (of varying degrees) in $m-1$ variables. Furthermore, g is a polynomial of degree $k-1$ in the first variable.

2. There exists a sequence of integers $\langle m_i \rangle_{i \in [k]}$ such that the polynomial g_i only depends on the first $m_i \leq m-1$ variables.

3. For every $i \in [k]$ there exists a sequence of integers $\langle d_{i,j} \rangle_{j \in [m-1]}$ such that $g_i(\vec{x})$ has a degree bound of $d_{i,j} \leq d$ in its jth variable.

4. The polynomial g must evaluate to zero on some subset of the points (due to some padding on input variables).

5. Finally, over some subset of the points g evaluates to either 0 or 1. (Note that this condition is not trivial since we will not be working with F_2 but some extension field K of F_2. In fact over the extension field, these constraints are not even linear. However since K is an extension of F_2, these conditions turn out to be F_2-linear.)

In what follows we will, in effect, be augmenting the reduction from SAT to GapPCS so as to include all constraints of the above form. This will force the GapPCS problem to only have satisfying assignments of the form $g = B(a)$ and thus salvage the reduction. (In actuality, we will be considering satisfying assignments that are presented as a concatenation of several pieces that are individually encoded and the constraints of the system we build will be verifying that the "concatenation" of the various pieces is a satisfying assignment. Furthermore, we will only by looking at systems of linear equations and not general satisfiability.)

The actual construction (i.e., proof of Lemma 5.6): Recall that we need to describe the three ingredients in the inner verifier: the encoding function $E : F_2^{\ell} \to (F_2^{\ell'})^n$, the proving function $P : F_2^{p\ell} \to (F_2^{\ell'})^N$, and the oracle machine Verdict. We start by developing the machinery for the encoding function and the proving function. We do so by transforming the question of satisfaction of a system of linear equations into a seqeunce of consistency relationships among polynomials and using this sequence to describe the encoding and proving function. Fix a linear space $L \in \mathcal{L}_{F_2, p\ell}$ and vectors $x_1, \ldots, x_p$ such that $(x_1, \ldots, x_p) \in L$.

Transforming the linear system. Our first step is to convert L into a conjunction of width-3 linear constraints (i.e., constraints that apply to at most 3 variables at a time). So we introduce a vector of auxiliary variables x_{p+1} on at most $n = p^2\ell^2$ variables and transform L into a linear space L' of width 3-constraints such that $(x_1, \ldots, x_p) \in L$ if and only if there exists x_{p+1} such that $(x_1, \ldots, x_{p+1}) \in L'$. (Note that $L' \in \mathcal{L}_{F_2, p\ell+n}$ and $|x_i| = \ell$ if $i \leq p$ whereas $|x_{p+1}| = n \gg \ell$. We'll take care of this discrepency in the next step.)

Low-degree extensions and dealing with padding. The low-degree extension is standard, but we need to deal with the padding it creates (and with the padding already done above). That is, we

have to augment the linear system to verify that the padded parts of the input are indeed all-zero.

We pick a field $K = \{\zeta_1 = 0, \zeta_2 = 1, \ldots, \zeta_{|K|}\}$, that extends F_2, of sufficiently large size (to be specified later), and a subset $H = \{\zeta_1, \ldots, \zeta_h\}$ of size $h = \lceil \log n \rceil$ and let $m = \lceil \log n / \log\log n \rceil$ so that $h^m \geq n$. Next, we let $x'_i = x_i 0^{h^m - n}$ (i.e., we pad x_i with enough zeroes so that its length is exactly h^m). Now, we let L'' be the F_2-linear constraints indicating that the padded parts of x'_i are zero, and $(x'_1, \ldots, x'_{p+1})$ correspond to the padding of $(x_1, \ldots, x_{p+1}) \in L'$.

Finally, as usual, we view x'_i as a function from $H^m \to \{0,1\}$ and let $f_1, \ldots, f_{p+1} : K^m \to K$ be m-variate polynomials of degree $h-1$ in each of the m variables that extend the functions described by $x'_1, \ldots, x'_{p+1}$. (We note for future reference that the encoding E function for x_i will essentially be the function f_i.)

Concatenating the p pieces (standard): Now let $f : K^{m+1} \to K$ be the function given by $f(\zeta_i, \cdots) = f_i(\cdots)$ if $i \in \{1, \ldots, p+1\}$ that is a polynomial of degree p in its first variable.

Low-degree extension of L'' (standard): Note that L'' imposes linear constraints of the form $\alpha_1 f(z_1) + \alpha_2 f(z_2) + \alpha_3 f(z_3)$ for $\alpha_1, \alpha_2, \alpha_3 \in \{0,1\}$ and $z_1, z_2, z_3 \in \{\zeta_1, \ldots, \zeta_{p+1}\} \times H^m$ on f. We extend L'' as a function $\hat{L''} : K^{3(m+1)+3} \to K$, by letting $L''(\alpha_1, \alpha_2, \alpha_3, z_1, z_2, z_3) = 1$, for $\alpha_1, \alpha_2, \alpha_3 \in H$ and $z_1, z_2, z_3 \in H^{m+1}$ if the constraint $\alpha_1 f(z_1) + \alpha_2 f(z_2) + \alpha_3 f(z_3)$ is imposed by L'', by letting $L''(\cdots) = 0$ for other inputs from H^{3m+6}, and letting L'' be a polynomial of degree $h-1$ in all other variables.

We comment that the current step does not rely on L'' being a linear function. The linearity of L'' (or rather of the condition $\alpha_1 f(z_1) + \alpha_2 f(z_2) + \alpha_3 f(z_3)$) will be used in the next step.

Verifying satisfiability of L'' via sequence of polynomials. This part is standard except for rule $(\mathcal{R}_0)$ below which includes an extra check that some elements being considered are 0/1. In fact, this part corresponds to the "sum check" in [1] (which is one of the two procedures in the original innner-verifier, the other being a low-degree test).

Let $m' = 4m + 8$. We define a sequence of polynomials $g_0, \ldots, g_{m'+1} : K^{m'} \to K$, where g_0 is essentially f; g_1 is related to g_0 by an F_2-linear relationship, and g_i is related to g_{i-1} by a K-linear relationship. The motivation behind these polynomials is the following: g_1 is defined so that the condition $(x_1, \ldots, x_p) \in L$ is equivalent to the condition $g_1(\vec{u}) = 0$ for every $\vec{u} \in H^{m'}$. The polynomials g_i relax this condition gradually, giving "$g_{i+1}(\vec{u}) = 0$ for every $\vec{u} \in F^i \times H^{m-i}$" if and only if "$g_i(\vec{u}) = 0$ for every $\vec{u} \in F^{i-1} \times H^{m-i+1}$". Thus finally we have $g_{m'+1} \equiv 0$ if and only if $(x_1, \ldots, x_p) \in L$. We now define these polynomials explicitly. For α_i's and u_i's from K and z_i's from K^{m+1}, let we define:

$$
\begin{aligned}
& g_0(z_1, \ldots, z_4, \alpha_1, \ldots, \alpha_4) \stackrel{\text{def}}{=} f(z_1) \\
(\mathcal{R}_0): \quad & g_1(z_1, \ldots, z_4, \alpha_1, \ldots, \alpha_4) = \left(\sum_{i=1}^{3} \alpha_i \cdot g_0(z_i \vec{0}) \right) \cdot \hat{L''}(\alpha_1, \alpha_2, \alpha_3, z_1, z_2, z_3) \\
& \qquad\qquad + \alpha_4 \cdot (g_0(z_4\vec{0})^2 - g_0(z_4\vec{0})).
\end{aligned}
$$

The terms involving $g_0(z_4\vec{0})$ are meant to verify that $g_0(z_4\vec{0})$ are always 0/1. These are "optional" in standard PCPs, in that they are not needed to get soundness, but are occasionally thrown in since they don't involve much extra work. In contrast, in our case these are necessary to enforce the augmented soundness condition. Note that while this condition is a

quadratic constraint (regarding g_0) over K, the map $\beta \mapsto \beta^2$ is an F_2-linear map over fields of characteristic two, and so the identity above is indeed F_2-linear, despite the quadratic term.

For $i = 1$ to m', let

$$(\mathcal{R}_i): \; g_{i+1}(u_1, \ldots, u_{i-1}, u_i, u_{i+1}, \ldots, u_{4m+8}) \; = \; \sum_{j=0}^{h-1} u_i^j \cdot g_i(u_1, \ldots, u_{i-1}, \zeta_j, u_{i+1}, \ldots, u_{4m+8}).$$

Merging the different polynomials into a single polynomial g (standard): Now, let $g : K^{m'+1} \to K$ be the function given by $g(i, z) = g_i(z)$ if $i \in \{0, \ldots, m'+1\}$ that is a degree $m'+1$ polynomial in the first variable i. Assuming $h \geq m' \geq p$, we have that g is a polynomial of individual degree at most $2h$ and thus has total degree at most $d = 2m'h$.

Lines and curves over g (standard): Let $g|_{\text{lines}} : K^{2m'+1} \to K^d$ be the function describing g restricted to lines. Let $w = 2(m'+1)h$, $\ell'' = wd$ and let $g|_{\text{curves}} : \mathcal{C} \to K^{\ell''}$ be the restriction of g to some subset $\mathcal{C}$ of degree w curves, where $\mathcal{C}$ are all the curves that arise in the verdict function's computations below.

The encoding and proving functions (standard): Finally, we get to define the encoding and proving functions. The encoding function $E(x_i)$ is the table of values of the function $f_i' : K^m \to K^{\ell''}$ where $f_i'(x) = (f_i(x), 0^{\ell''-1})$ (i.e., elements of K are being written as vectors from $K^{\ell''}$. The proving function $P(L, x_1, \ldots, x_p)$ consists of the triple of functions $(g', (g|_{\text{lines}})', g|_{\text{curves}})$, where $g' : K^{m'+1} \to K^{\ell''}$ and $(g|_{\text{lines}})' : K^{2(m'+1)} \to K^{\ell''}$ are the functions g and $g|_{\text{lines}}$ with their range being mapped, by padding, into $K^{\ell''}$.

We now describe the verdict function. To motivate this, recall that the verdict function, which essentially has access to oracles for g, $g|_{\text{lines}}$, $g|_{\text{curves}}$ and $f_1, \ldots, f_p$, needs to verify the following items:

1. g is a polynomial of degree at most d, $g|_{\text{lines}}$ is the restriction of g to lines, and $g|_{\text{curves}}$ is the restriction of g to curves.
2. The degree of g in its first variable is at most $m'+1$.
3. For $i \in \{1, \ldots, m'+1\}$, then function g_i given by $g_i(u) = g(i, u)$ is computed correctly from g_{i-1} by an application of the rule $(\mathcal{R}_{i-1})$.
4. Verify that $g_{m'+1}$ is identically zero.
5. Verify that g_0 is a polynomial of degree 0 in all but its first $m+1$ variables.
6. Verify that the function $f : K^{m+1} \to K$ given by $f(x) = g_0(x, 0 \cdots 0)$ is a polynomial of degree at most p in its first variable and a polynomial of degree at most $h-1$ in the remaining m variables.
7. Verify that $f(i, \cdots) = f_i(\cdots)$ for every $i \in \{1, \ldots, p\}$.

(Working one's way upwards, one can see that $P(L, x_1, \ldots, x_p)$ is the only function to satisfy all the above constraints.)

We are now ready to describe the verifier's actions (or to be formal, the Verdict function). The aim is to emulate a large number of checks (i.e., random verification of all the above conditions) by

using only $p+3$ oracle calls, and still incur only a constant error probability. Specifically, ignoring condition (1) for a moment, a random test of condition (2) requires $m'+2$ points in the domain of g, condition (3) involves $m'+1$ equalities (which refer to $m'+1$ different parts of g), condition (5) involves $m'-m$ equalities (one per each suitable variable in g_0), and condition (7) involves p equalities, each referring to a different function f_i. Following [1], all these different conditions will be checked by retreiving the corresponding (random) g-values from a suitable curve in $g|_{\text{curves}}$, and obtaining the f_i-values from the corresponding oracles. Finally, Condition (1) will be tested via an adequate low-degree test that makes only 2 additional queries. Details follow.

The verifier first picks one random test (to be emulated) per each of the equalities corresponding to the conditions (2)–(7) above. Specifically, in order to emulate the testing of conditions (2), (5) & (6), it picks random axis parallel lines (one per each of the relevant variables) and picks $O(h)$ points on these $K^{m'+1}$-lines with the intent of inspecting the value of g' at these points. (We stress that the verifier does not query g' at these points, but rather only determines these points at this stage.) Similarly, in order to emulate the testing of conditions (3), (4) & (7), it picks random points from the domain of the corresponding g_i's and f. Having chosen these points, it picks one totally random point in $K^{m'}$. All in all this amounts to determining $w = O(mh)$ points in the domain of g'. The verifier then determines a degree w curve, denoted C, (over $K^{m'+1}$) that passes through these m points. Next, it picks a random point α on this curve and a random line l through the point α.

We finally get to the actual queries of the verifier. The verdict function queries $g'(\alpha)$, $(g|_{\text{lines}})'(l)$ and $g|_{\text{curves}}(C)$. It verifies that $g'(\alpha)$ is actually in K and $(g|_{\text{lines}})'(l)$ is in K^d (as opposed to K^ℓ). It then verifies that the three responses agree at α. Finally, it verifies the values of g' on the test points for tests (2)-(7), as claimed by $g|_{\text{curves}}(C)$, are consistent with the conditions (2)-(7). In particular, verifying condition (7) requires one probe each into the oracles $X_1, \ldots, X_p$. (Once again the responses to these probes are elements of K^ℓ and the verdict verifies that the responses are in K padded with 0's.) Thus, in total, we made only $3+p$ queries.

This concludes the description of the verifier. We stress that all the "0-padding verifications" are only intended to guarantee the modified notion of soundness (and are not needed for the standard notion of soundness). The same holds with respect to the various tests of individual degrees (which test a degree lower than the (curve-to-line) low degree test). Omitting all these extra test, would get us back to [1].

The modified soundness of the above verifier is established as usual assuming $|K| \geq \text{poly}(\ell''/\epsilon)$. In particular, if the function $g : K^{m'+1} \to K$ obtained by ignoring the last $\ell''-1$ coordinates of the function g' is not, say .01-close to some polynomial $\hat{g}$ of total degree d then the low-degree test will reject with constant probability. If the response of the query to $g|_{\text{curves}}$ is not consistent with $\hat{g}$ on all the queried points, then the curve to g' consistency test will detect this with constant probability. Finally if any of the conditions (2)-(7) is violated, then the final check above detects with constant probability.

Recall that the oracle machine Verdict makes $p+3$ queries in all. The answers it receives are from $K^{\ell''}$ and thus ℓ', the answer length, equals $\ell'' \log_2 |K|$ which is $\text{poly}\log(p\ell)$ as required. The soundness error, γ, is some constant bounded away from 0. Finally, note that all checks by the verifier are actually K-linear, except for the satisfaction of rule $(\mathcal{R}_0)$, which is only F_2-linear. ■

5.4 Combining all the constructions

We are now ready to prove the main theorem of this section.

Theorem 5.7 (Theorem 2.1, restated): *For infinitely many k, there exists a linear locally-testable binary code mapping k bits to $n \stackrel{\text{def}}{=} k^{O(\sqrt{\log k}\log\log k)}$ bits. Furthermore, the codes has distance $\Omega(n)$.*

Proof: Composing (as per Lemma 5.4) the $(F_2,(p,\ell)\to(p+3,\mathrm{poly}(\log p\ell)),1/2,\gamma)$-linear inner verifier of Lemma 5.6 with the $(F_2,(p+3,\mathrm{poly}(\log p\ell))\to(p+4,1),1/2,\gamma)$-linear inner verifier of Lemma 5.5, we obtain that there exist constants $\delta_1,\gamma_1>0$ such that for every constant p' and for every ℓ', there exists an $(F_2,(p',\ell')\to(p'+4,1),\delta_1,\gamma_1)$-linear inner verifier V_1. Furthermore, V_1 uses $r_1=O(\log p'\ell')+2(p'+3)\cdot\mathrm{poly}(\log p'\ell')=\mathrm{poly}(\log\ell')$ coins, encoding of length $n_1=\mathrm{poly}(\ell')\cdot\exp(\mathrm{poly}(\log p'\ell'))$, and proofs of length $m_1=\mathrm{poly}(n_1)$.

Similarly, for any constant p, composing the verifier of Lemma 5.6 with V_1 (while setting $p'=p+3$ and $\ell'=\mathrm{poly}(\log\ell)$),[15] we get a $(F_2,(p,\ell)\to(p+7,1),\delta_2,\gamma_2)$-linear inner verifier V_2 for some $\delta_2,\gamma_2>0$. Furthermore, V_2 uses $r_2=O(\log p\ell)+r_1=O(\log\ell)+\mathrm{poly}(\log\log\ell)$ coins, encoding of length $n_2=\mathrm{poly}(\ell)\cdot n_1=\mathrm{poly}(\ell)$, and proofs of length $m_2=\mathrm{poly}(n_2)$.

Our final step will be to compose (as per Proposition 5.3) the truncated version of the FS-code (from Section 3.2) with the linear inner verifier V_2. Recall that, for any constant $c>1/2$, the truncated version of the FS-code maps $(F^{d+1})^K$ to $(F^{d+1})^N$, where $N=\exp(\log^c K)\cdot K$ and $|F|=\Theta(d)<\exp(\log^c K)$. The corresponding codeword test uses $R\stackrel{\text{def}}{=}\log_2 N+2\log\log|F|$ random bits and makes $q=O(1)$ queries. Using $F=F_2^{O(1)+\log_2 d}$ and $\Sigma=F^{d+1}\equiv F_2^{O(d\log d)}$, we apply Proposition 5.3 to this code (and codeword test) and V_2 above, while setting $p=q$, $\ell=O(d\log d)$ and $b=O(d\log d)$, where $d<\exp(\log^c K)$ (for any constant $c>1/2$). We obtain a binary linear locally-testable code mapping $(F_2^b)^K$ to F_2^M, where $M=O(N\cdot n_2+2^R\cdot m_2)$. Using $R=\log_2 N+O(\log\log d)$ and $m_2=\mathrm{poly}(n_2)=\mathrm{poly}(\ell)=\mathrm{poly}(d)$, we get $M=N\cdot\exp(O(\log^c K))=K\cdot\exp(O(\log^c K))$. The theorem follows. ■

5.5 Additional remarks

In this section we show that locally testable codes over small alphabets can be modified such that the tester only uses randomness that is logarithmic in the codeword and only makes three queries. We stress that the stated modification increases the length of the codewords by a constant factor. We start with reducing the randomness complexity of the tester.

Proposition 5.8 *Let $C:\Sigma^k\to\Sigma^n$ be a code. Then every codeword tester for C can be modified into one that maintains the same acceptance probabilities up-to an additive term of ϵ, while preserving the number of queries and using randomness complexity at most $O(\log(1/\epsilon))+\log_2 n+\log_2\log_2|\Sigma|$.*

Proof: The proof follows the standard/easy part of the proof of Lemma 3.1 (and analogous results in Section 4). Specifically, using the probabilistic method, there exists a set of $O(\epsilon^{-2}\log_2|\Sigma^n|)$ possible random-tapes for the original tester so that if the tester restricts its choices to this set then its acceptance probability on every potential sequence is preserved up to an additive term of ϵ. (Observe that, with probability $1-\exp(-\epsilon^2 t)$, a random set of t random-tapes approximates the acceptance probability for a fixed sequence up to ϵ, and that the number of possible sequences is $|\Sigma^n|$.) The proposition follows. ■

Using Proposition 5.8, we show that *our main result regarding locally testable codes* (i.e., Theorem 2.1) *holds also with tester that make only three queries.* The latter assertion is an immediate corollary of the following proposition.

[15] Thus, $r_1=\mathrm{poly}(\log\mathrm{poly}(\log\ell))=\mathrm{poly}(\log\log\ell)=o(\log\ell)$ and $n_1=\exp(\mathrm{poly}(\log\mathrm{poly}(\log\ell)))=\exp(\mathrm{poly}(\log\log\ell))=\ell^{o(1)}$.

Proposition 5.9 *Let $C : \Sigma^k \to \Sigma^n$ be a locally-testable linear code of distance $d = \Omega(n)$. Then, there exists a linear code $C' : \Sigma^k \to \Sigma^{O(n \log |\Sigma|)}$ of distance at least d that is testable with three queries.*

Proof: By Proposition 5.8, the code C is locally-testable by a tester, denoted T, having randomness complexity $\rho \stackrel{\text{def}}{=} \log_2 n + O(1) + \log_2 \log_2 |\Sigma|$. By a slight modification of T (which only increases ρ by an additive constant),[16] we may assume that T checks a single linear combination (determined by its random-tape) of the oracle answers. We construct a code C', by augmenting C with a suitable encoding of each of the answer tuples obtained by T when using a fixed random-tape. Specifically, for each possible $r \in \{0,1\}^\rho$, we consider the $q = O(1)$ queries, denoted $(i_1, ..., i_q)$, made by T and the linear combination $(c_1, ..., c_q) \in \Sigma^q$ of the answers checked by the tester. For $x \in \Sigma^n$ and $r \in \{0,1\}^\rho$, we augment $C(x)$ by a block of length $q-1$ such that the ℓth symbol in the block equals $\sum_{j=1}^{\ell+1} c_j x_{i_j}$. Thus, we obtain a code C' of length $n + 2^\rho \cdot (q-1) = n + O(n \log(|\Sigma|)$ over Σ. The corresponding tester for C', performs at random (with equal probability) one of the following two tests:

1. A consistency test:[17] The test selects at random a random-tape r for T and a query (out of the q queries) that T makes on random-tape r. It checks whether the answer obtained from the n-symbol prefix of C' matches the value obtain from the block corresponding to r. Specifically, suppose that on coins r the tester T makes the queries $i_1, ..., i_q \in [n]$ and checks the linear combination $(c_1, ..., c_q) \in \Sigma^q$, and that we decided to check the jth query (where $\ell \in [q]$). For $j > 1$, we compare c_j times the answer obtained from the prefix of C' (i.e., the i_jth bit of the alledged codeword) to the difference between the jth and $j-1$st entries in the block corresponding to r. For $j = 1$ we compare the first entry in the block corresponding to r to the weighted sum of the answers obtained to queries i_1 and i_2.

2. Emulating T: The test selects at random a random-tape r for T and checks the corresponding linear condition by obtaining the desired linear combination of the answer bits from the last entry of the block corresponding to r.

The proposition follows. ■

Perspective. Proposition 5.9 indicates that *three* queries suffice for a meaningful definition of locally-testable linear codes. This result is analogous to the three-query PCPs available for NP-sets. In both cases, the constant error probability remains unspecified, and a second level project aimed at minimizing the error of three-query test arises. Another worthy project refers to the trade-off between the number of queries and the error probability, which in the context of PCP is captured by the notion of amortized query complexity. The definition of an analogous notion for locally-testable codes is less staightforward because one needs to specify which strings (i.e., at what distance from the code) should be rejected with the stated error probability. One natural choice is to consider the error probability of strings that are at distance $d/2$ from the code, where d is the distance of the code itself.

[16] In addition, the detection probability is reduced by a constant factor.

[17] Indeed, this consistency test is quite weak (but it suffices for our purposes). This consistency test reduces the rejection probability by a factor of q. Stronger consistency test seem to require more redundent encodings (e.g., one may use the Hadamard code). But since our focus is on the total length of C', our choice of a trivial code (which corresponds to using auxiliary variables) seems best.

6 Conclusions and Open Problems

Our code constructions are randomized, and so we do not obtain fully-explicit codes. The randomization amounts to selecting a random subspace of random-tapes for certain low-degree tests, and the probabilistic analysis shows that almost all choices of the subspace will do. A natural (de-randomization) goal is to provide an explicit construction of a good subspace. For example, in case of the low-degree test, the goal is to provide an explicit set of $\tilde{O}(|F|^m)$ lines that can be used (as R_m in the construction of Section 3.2).

As a seemingly easier goal, consider the linearity test of Blum, Luby and Rubinfeld [7]: To test whether $f : G \to H$ is linear, one uniformly selects $(x, y) \in G \times G$ and accepts if and only if $f(x) + f(y) = f(x + y)$. Now, by the probabilistic method, there exists a set $R \subset G \times G$ of size $O(|G| \log |H|)$ such that the test works well when (x, y) is uniformly selected in R (rather than in $G \times G$).[18] The goal is to present an explicit construction of such a set R. Recent progress on this special case (i.e., derandomization of the BLR test) is reported in [14].

Another natural question that arises in this work refers to obtaining locally-testable codes for coding $k' < k$ information symbols out of codes that apply to k information symbols. The straightforward idea of converting k'-symbol messages into k-symbol messages (via padding) and encoding the latter by the original code, preserves many properties of the code but does not necessarily preserve local-testability.[19]

We have presented locally testable codes and PCP schemes of almost-linear length, where $\ell : \mathsf{N} \to \mathsf{N}$ is called almost-linear if $\ell(n) = n^{1+o(1)}$. For PCP, this improved over a previous result where for each $\epsilon > 0$ a scheme of length $n^{1+\epsilon}$ was presented (with query complexity $O(1/\epsilon)$). Recall that our schemes have length $\ell(n) = \exp(\log n)^c) \cdot n$, for any $c > 0.5$. We wonder whether length $\ell(n) = \mathrm{poly}(\log n) \cdot n$ (or even linear length) can be achieved. Similarly, the number of queries in our proof system is really small, say 16, while simultaneously achieving nearly linear-sized proofs. Further reduction of this query complexity is very much feasible and it is unclear what the final limit may be. Is it possible to achieve nearly-linear (or even linear?) proofs with 3 query bits and soundness nearly 1/2?

[18] For every $f : G \to H$, with probability $1 - \exp(-|R|)$ a random set R will be good for testing whether f is linear, and the claim follows using the union bound for all $|H|^{|G|}$ possible functions $f : G \to H$.

[19] Indeed, this difficulty (as well as other difficulties regarding the gap between PCPs and codes) disappears if one allows probabilistic coding. That is, define a code $\mathcal{C} : \Sigma^k \to \Sigma^n$ as a randomized algorithm (rather than a mapping), and state all code properties with respect to randomized codewords $\mathcal{C}(a)$'s.

References

[1] S. Arora, C. Lund, R. Motwani, M. Sudan and M. Szegedy. Proof Verification and Intractability of Approximation Problems. *JACM*, Vol. 45, pages 501–555, 1998. Preliminary version in *33rd FOCS*, 1992.

[2] S. Arora and S. Safra. Probabilistic Checkable Proofs: A New Characterization of NP. *JACM*, Vol. 45, pages 70–122, 1998. Preliminary version in *33rd FOCS*, 1992.

[3] S. Arora and M. Sudan. Improved low degree testing and its applications. In *29th STOC*, pages 485–495, 1997.

[4] L. Babai, L. Fortnow, L. Levin, and M. Szegedy. Checking Computations in Polylogarithmic Time. In *23rd STOC*, pages 21–31, 1991.

[5] M. Bellare, S. Goldwasser, C. Lund, and A. Russell. Efficient probabilistically checkable proofs and applications to approximation. In *26th STOC*, 1994.

[6] E. Ben-Sasson, O. Goldreich, and M. Sudan. Impossibility results for 2-query codeword testing. In preparation, 2002.

[7] M. Blum, M. Luby and R. Rubinfeld. Self-Testing/Correcting with Applications to Numerical Problems. *JCSS*, Vol. 47, No. 3, pages 549–595, 1993.

[8] N. Creignou, S. Khanna, and M. Sudan. *Complexity Classifications of Boolean Constraint Satisfaction Problems*. SIAM Press, Philadeplhia, PA, USA, March 2001.

[9] U. Feige, S. Goldwasser, L. Lovász, S. Safra, and M. Szegedy. Approximating Clique is almost NP-complete. *JACM*, Vol. 43, pages 268–292, 1996. Preliminary version in *32nd FOCS*, 1991.

[10] G.D. Forney. *Concatenated Codes*. MIT Press, Cambridge, MA 1966.

[11] K. Friedl and M. Sudan. Some Improvements to Low-Degree Tests. In the *3rd Israel Symp. on Theory and Computing Systems (ISTCS)*, 1995.

[12] O. Goldreich, S. Goldwasser, and D. Ron. Property testing and its connection to learning and approximation. *JACM*, pages 653–750, July 1998.

[13] O. Goldreich, H. Karloff, L.J. Schulman and L. Trevisan. Lower Bounds for Linear Locally Decodable Codes and Private Information Retrieval. In the *Proc. of the 17th IEEE Conference on Computational Complexity*, 2002.

[14] O. Goldreich and A. Wigderson. On derandomizing the BLR test. Private communication, June 2002.

[15] P. Harsha and M. Sudan. Small PCPs with Low Query Complexity. *Computational Complexity*, 9(3-4):157-201, 2000.

[16] J. Katz and L. Trevisan. On The Efficiency Of Local Decoding Procedures For Error-Correcting Codes. In the *32nd STOC*, 2000.

[17] A. Polishchuk and D.A. Spielman. Nearly-linear size holographic proofs. In *26th STOC*, pages 194–203, 1994.

[18] R. Raz and S. Safra. A sub-constant error-probability low-degree test, and a sub-constant error-probability PCP characterization of NP. In *29th STOC*, 1997.

[19] R. Rubinfeld and M. Sudan. Robust characterization of polynomials with applications to program testing. *SIAM Journal on Computing*, Vol. 25 (2), pages 252–271, 1996.

A The Gap Polynomial-Constraint-Satisfaction Problem

We start by recalling the "Gapped Polynomial Constraint Satisfaction Problem" and introducing a restricted version of this problem.

Constraint satisfaction problems (CSPs) are a natural class of "optimization" problems where an instance consists of t Boolean constraints $C_1, \ldots, C_t$ placed on n variables taking on values from some finite domain, say $\{0, \ldots, D\}$. Each constraint is restricted in that it may only depend on a small number w of variables. The goal of the optimization problem is to find an assignment to the n variables that maximizes the number of constraints that are satisfied. The complexity of the optimization task depends on the nature of constraints that may be applied, and thus each class of constraints gives rise to a different optimization problem (cf. [8]). CSPs form a rich subdomain of optimization problems that include Max 3SAT, Max 2SAT, Max Cut, Max 3-Colorability etc. and have been easy targets of reductions from PCPs.

Following Harsha and Sudan [15], we consider algebraic variants of CSPs. These problems come with some syntactic differences: The domain of the value that a variable can take on will be associated with a finite field F; the index set of the variables will now by F^m for some integer m, rather than being the set $[n]$; and thus an assignment to the variables may be viewed naturally as a function $f : F^m \to F$. Thus the optimization problem(s) ask for functions that satisfy as many constraints as possible. In this setting, constraints are also naturally interpreted as algebraic functions, say given by an algebraic circuit.

The interesting (non-syntactic) aspect of these problems is when we optimize over a restricted class of functions, rather than the space of all functions. Specifically, we specify a degree bound d on the function $f : F^m \to F$ and ask for the maximum number of constraints satisfied by degree d polynomial functions f. Under this restriction on the space of solutions, it is easier to establish NP-hardness of the task of distinguishing instances where all constraints are satisfiable, from instances where only a tiny fraction of constraints are satisfiable. This motivates the "Gapped Polynomial CSP", first defined by Harsha and Sudan [15]. Here we consider a restriction on the class of instances, where each constraint, in addition to being restricted to apply only to w variables, is restricted to apply only to variables that lie on some "2-dimensional variety" (i.e., the names/indices of the variables that appear in a constraint must lie on such a variety). We define this notion first.

A set of points $x_1, \ldots, x_k \in F^m$ is said to lie on a 2-dimensional variety of degree r if there exists a function $Q = (Q_1, \ldots, Q_m) : F^2 \to F^m$ where each Q_i is a bivariate polynomial of degree r, such that there exist points $y_1, \ldots, y_k \in F^2$ such that $x_j = Q(y_j)$ for every $j \in [k]$.

Definition A.1 (rGapPCS (restricted Gap Polynomial Constraint Satisfaction)) *For $\epsilon : \mathsf{Z}^+ \to \mathsf{R}^+$ and $r, m, b, q : \mathsf{Z}^+ \to \mathsf{Z}^+$, the promise problem $\mathrm{rGapPCS}_{\epsilon,r,m,b,q}$ has as instances tuples $(1^n, d, k, s, F; C_1, \ldots, C_t)$, where $d, k, s \leq b(n)$ are integers, F is a field of size $q(n)$ and $C_j = (A_j; x_1^{(j)}, \ldots, x_k^{(j)})$ is algebraic constraint given by an algebraic circuit A_j of size s on k inputs and the variable names $x_1^{(j)}, \ldots, x_k^{(j)} \in F^m$, where for $m = m(n)$ and for every $j \in [t]$ the points $\{x_i^{(j)}\}_i$ lie on some 2-dimensional variety of degree r.*

YES-instances: *$(1^n, d, k, s, F; C_1, \ldots, C_t)$ is a* YES*-instance if there exists a polynomial $p : F^m \to F$ of total degree at most d such that for every $j \in \{1, \ldots, t\}$, the constraint C_j is satisfied by p; that is, $A_j(p(x_1^{(j)}), \ldots, p(x_k^{(j)})) = 0$.*

NO-instances: *$(1^n, d, k, s, F; C_1, \ldots, C_t)$ is a* NO*-instance if for every polynomial p of degree d, at most $\epsilon(n) \cdot t$ constraints are satisfied.*

The following lemma is a slight variant of Lemma 3.16 in [15]. Specifically, while [15] use the generic fact that any w points lie in a c-dimensional variety of degree $cw^{1/c}$, we note that the specific $O(m(n)b(n))$ points chosen for each constraint (in the reduction) lie on a 2-dimensional variety of degree $O(m(n))$. This is because each constraint refers to $O(m(n)b(n))$ points that lie on one out of $O(m(n))$ lines.

The following lemma simply lists conditions on the parameters which allows GapPCS to be NP-hard. We describe the actual choice of parameters in a corollary to be described shortly.

Lemma A.2 *There exists a constant c and polynomials p_1, p_2 such that for any collection of functions $\epsilon : \mathsf{Z}^+ \to \mathsf{R}^+$ and $m, r, b, q, \ell : \mathsf{Z}^+ \to \mathsf{Z}^+$ such that $b(n) \geq \log n$, $(b(n)/m(n))^{m(n)} \geq n$, $r(n) \geq cm(n)$, $q(n) \geq (b(n)/\epsilon(n))p_1(m(n))$, and $\ell(n) \geq p_2(b(n))(q(n))^{m(n)}$, SAT reduces to $\mathrm{rGapPCS}_{\epsilon,r,m,b,q}$ under $\ell(n)$-length preserving reductions,*

On the other hand, when applying the MIP system of [15, Section 3.6] to restricted GapPCS instances, we get:

Lemma A.3 *There exists a polynomial p such that if $\epsilon : \mathsf{Z}^+ \to \mathsf{R}^+$ and $r, m, b, q : \mathsf{Z}^+ \to \mathsf{Z}^+$, satisfy $q(n) \geq \mathrm{poly}(r(n))(b(n)/\epsilon(n))$ then the promise problem $\mathrm{rGapPCS}_{\epsilon,r,m,b,q}$ has a 3-prover MIP proof with perfect completeness, soundness $O(\epsilon(n))$, answer length $\mathrm{poly}(b(n)) \log q(n)$, and randomness $O(\log N + m(n) \log q(n))$, where N denotes the size of the GapPCS instance and n denotes the first parameter in the instance. Furthermore, the size of the first prover oracle is $q(n)^{m(n)}$, and its answer length is $\log q(n)$.*

The lemma above allows us to work with the GapPCS problem for an appropriate choice of the parameters ϵ, m, b, q, ℓ. Combining the above two lemmas, we state the resulting corollary regarding 3-prover MIPs for SAT, where we restrict attention to the case of constant $\epsilon > 0$.

Corollary A.4 *For every constant $\epsilon > 0$ and $m : \mathsf{Z}^+ \to \mathsf{Z}^+$, let $\ell(n) = m(n)^{O(m(n)} \cdot n^{1+O(1/m(n))}$. Then SAT reduces in probabilistic polynomial time under $\ell(n)$-length preserving reductions to a promise problem that has a 3-prover proof system with perfect completeness, soundness ϵ, logarithmic randomness, and answer length $m(n)^{O(1)} \cdot n^{O(1/m(n))}$, in which the first prover has size linear in the instance size.*

Proof: Assume without loss of generality that $m(n) \leq \log n/(3 \log\log n)$. (For larger $m(\cdot)$, the requirements on both the function $\ell(n)$ and the answer length become weaker.) Set $b(n) = m(n)n^{1/m(n)}$. Note that this makes $b(n) \geq \log n$ (and $(b(n)/m(n))^{m(n)} \geq n$) as required in Lemma A.2. Next, set $r(n) = cm(n)$, where c is from Lemma A.2, and set $q(n) = (b(n)/\epsilon)\mathrm{poly}(m(n)) = \mathrm{poly}(m(n))n^{1/m(n)}/\epsilon$ such that it satisfies the requirements in both Lemmas A.2 and A.3. Finally, set $\ell(n) = \mathrm{poly}(b(n))q(n)^{m(n)} = \mathrm{poly}(m(n)) \cdot n^{O(1/m(n))} \cdot m(n)^{O(m(n))} \cdot n \cdot \epsilon^{-m(n)} = O(m(n)^{O(m(n))} \cdot n^{1+O(1/m(n))})$. This setting satisfies all the conditions of Lemmas A.2 and A.3, which yields a 3-prover proof system for SAT in which the answer lengths are bounded by $\mathrm{poly}(b(n)) \log q(n) = m(n)^{O(1)} \cdot n^{1/O(m(n))}$. Furthermore, the size of the first prover is $q(n)^{m(n)} < \ell(n)$, as required. ■