Locally Testable Codes and PCPs of Almost-Linear Length*

Oded Goldreich†
Department of Computer Science
Weizmann Institute of Science
Rehovot, Israel.
oded.goldreich@weizmann.ac.il

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

March 2, 2004

Abstract

We initiate a systematic study of locally testable codes; that is, error-correcting codes that admit very efficient membership tests. Specifically, these are codes accompanied with tests that make a constant number of (random) queries into any given word and reject non-codewords 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 that 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 binary (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.

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

*An extended abstract of this work has appeared in the *43rd FOCS*, 2002. A preliminary full version [18] has appeared on ECCC.

Contents

1 Introduction

Locally testable codes are (good) error-correcting codes that admit very efficient codeword tests. Specifically, the testing procedure may use only a *constant number* of (random) queries, and should reject non-codewords with probability proportional to their distance from the code.

Locally testable codes are related to Probabilistically Checkable Proofs (PCPs) and to Property Testing [16, 23]. Specifically, locally testable codes can be thought of as a combinatorial counterparts of the complexity theoretic notion of PCPs, and in fact the use of codes with related features is implicit in known PCP constructions. Local testability of codes is also a special case of property testing (i.e., the study of sublinear-time approximation algorithms for testing whether an input is "close" to one satisfying a fixed property), and indeed the first collection of properties that were shown to be testable also yield constructions of locally testable codes [11].

Locally testable codes were introduced in passing, by Friedl and Sudan [15] and Rubinfeld and Sudan [23]. However, despite the central role of locally testable codes in complexity theoretic and algorthmic research, they have received little explicit attention so far. The primary goal of this work is to initiate a systematic study of locally testable codes. In particular, we focus on the construction of locally testable codes over a binary alphabet and on the development of techniques to reduce the alphabet size of locally testable codes. Studying the length of locally testable codes, we obtain for the first time (even for non-binary alphabets), codes of almost-linear length.

Some well-known examples: To motivate some of the parameters of concern, we start by considering some "trivial codes" that are easily testable. It is easy to test (membership in) the code that contains all strings of a given length (e.g., simply accept every string without looking at it). It is also easy to test the code that consists of only one codeword (e.g., given an arbitrary string x, pick random index i and verify that x and the single codeword agree at the the ith coordinate). Thus, the concept of locally testable codes is interesting mainly in the case of "good" codes; that is, codes that have "many" codewords that are pairwise at "large" distance from each other.

One non-trivial code allowing efficient testing is the Hadamard code: the codewords are linear functions represented by their values on all possible evaluation points. The number of codewords in Hadamard codes grows with the length of the code, and the pairwise distance between codewords is half of the length of the code. So this code does not admit trivial tests as above. It turns out that in this case codeword testing amounts to linearity testing [11], and this can be performed efficiently, though the analysis is quite non-trivial.

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.)

A basic question: The question addressed in this work is whether one can hope for a better relation between the number of information bits, denoted k, and the length of the codeword, denoted n. Specifically, *can n be polynomial or even linear in k?* For a sufficiently large *non-binary* alphabet, Friedl and Sudan [15] showed that n can be made nearly quadratic in k. The main contribution of this paper is the demonstration of the existence of locally testable codes in which n is *almost-linear* in k (i.e., $n = k^{1+o(1)}$), *even for the binary alphabet.*

1.1 Relation to PCP

As mentioned earlier, locally testable codes are closely related to Probabilistically Checkable Proofs (PCPs). A proof system is defined by a (probabilistic) verifier that is given a pair of strings – a purported theorem (assertion) and a claimed proof (evidence) – such that if the theorem is true, then there exists a proof such that the verifier accepts; and if the assertion is not true then no evidence causes the verifier to accept (with high probability). Furthermore, PCP verifiers achieve their goals by making only a small number of queries to the proof, which is given as an oracle. The PCP Theorem [2, 3] shows how to construct PCP verifiers that make only a constant number of queries to the proof oracle.

PCPs achieve their strong features by implicitly relying on objects related to locally testable codes. Indeed the construction of codes over large alphabets that are testable via a small (yet not necessarily constant) number of queries lies at the heart of many PCPs. It is a common belief, among PCP enthusiasts, that the PCP Theorem [2, 3] already provides (binary) locally testable codes. This belief relates to a stronger property in the proof of the PCP theorem which actually provides a transformation from standard witnesses for, say SAT, to PCP-proof-oracles, such that transformed strings are accepted with probability one by the PCP verifier. 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 $\mathrm{poly}(k)$ (or almost linear in k, when using [21]), which are pairwise quite far from each other. The common belief is that the PCP-verifier also yields a codeword test. However, this is not quite true: typically, the analysis only guarantee that each passing oracle can be "decoded" to a corresponding NP-witness, but encoding the decoded NP-witness does not necessarily yield a string that is close to the oracle. In particular, this allows for oracles that are accepted with high probability to be far from any valid codeword. Furthermore, it is not necessarily the case that only codewords pass the test with probability one. For example, part of the proof oracle is supposed to encode an m-variate polynomial of *individual degree* d, yet the (standard) 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).

We conclude that the known constructions of PCPs as such do not yield locally testable codes. However, 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 do yield coding results of the type we seek (e.g., see Theorem 2.3).

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 PCPs. Specifically, we obtain constant-query PCP systems that utilize oracles that are shorter than known before (see Theorem 2.5).

1.2 Relation to Property Testing

Property testing is the study of highly efficient approxiamtion algorithms (tests) for determining whether an input is close to satisfying a fixed property. Specifically, for a property (Boolean function) P, a test may query an oracle for a string x at few positions and accept if $P(x)$ is true, and reject with high probability if $P(\tilde{x})$ is not true for every $\tilde{x}$ that is "close" to x. Property testing was defined in [23] (where the focus was on algebraic properties) and studied systematically in [16] (where the focus was on combinatorial properties).

Viewed from the perspective of property testing, the tester of a local testable code is a tester for the property of being a member of the code, where the notion of "closeness" is based on Hamming distance. Furthermore, in the coding setting, it is especially natural that one is not interested in the exactly deciding whether or not the input is a codeword, but rather in the "approximate" distance of the input from the code (i.e., whether it is a codeword or far from any codeword). Thus locally testable codes are especially well-connected to the theme of property testing. Indeed the first few property tests in the literature (e.g., linearity tests [11], low-degree tests [6, 5, 23, 15]) can be interpreted as yielding some forms of locally testable codes. More recent works on algebraic testing [7, 1] highlight the connections to codes more explicitly. Our work also uses the results and techniques developed in the context of low-degree testing. However, by focussing on the codes explicitly, we highlight some missing connections. In particular, most of the prior work focussed on codes over large alphabets and do not show how to go from testable codes over large alphabets to codes over small alphabets. In this work we address such issues explicitly and resolve them to derive our results. Furthermore, we focus on codes that can be tested by making a constant numbder of queries.

1.3 Relation to Locally Decodable Codes

A task that is somewhat complementary to the task investigated in this paper, is the task of local decoding. This is the task of constructing codes that have very efficient (sub-linear time) decoding algorithms. Specifically, given oracle access to a string that is close to some unknown codeword, the decoding procedure should compute any coordinate of the nearby message while making, say, a constant number of queries to the input oracle. Codes that have such decoding algorithms are termed locally decodable codes. While local testability and local decodability appear related, no general theorems linking the two tasks are known. In fact, known lower bounds suggest that local decodability is "harder" to achieve than local testability. 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 [20]: that is, if q queries are used for recovery then $n = \Omega(k^{1+(1/(q-1))})$.
- For a (large) alphabet that can be viewed as vector space over some field F, we show almost-linear length F-linear codes in which testing requires only *two* queries. In contrast, it was shown that F-linear codes with *two* query recovery require exponential length [17].[1]

1.4 Organization and previous versions

Section 2 provides a formal treatment of the aforementioned notions as well as a formal statement of our main results. Direct and self-contained constructions of locally testable codes (albeit not achieving the best results) are presented in Section 3. We stress that the latter make no reference to PCP, although they do use low-degree tests. Our best constructions of locally testable codes are presented in Section 5, where we adapt standard PCP constructions and combine them with the construction presented in Section 3.2. In Section 4, we adapt some of the ideas presented in Section 3.2 in order to derive improved PCPs. We stress that Sections 4 and 5 can be read independently of one another, whereas they both depend on Section 3.2. Subsequent works and open problems are discussed in Section 6.

[1]An F-linear code over the alphabet $\Sigma = F^\ell$ is a linear space over F (but not necessarily over F^ℓ). In our codes (which support two-query tests) it holds that $\ell = \exp(\sqrt{\log k})$ and $|F| = O(\ell)$, while $n < k^{1+(\log k)^{-0.4999}}$. In contrast, the lower-bound on n (for two-query decoding) established in [17] assert that $n > \exp(\Omega(k - (\ell \cdot \ell')^2))$ in case $F = GF(2^{\ell'})$, which (for $\ell = \exp(\sqrt{\log k}) = k^{o(1)}$ and $\ell' = O(1) + \log \ell$) yields $n > \exp(\Omega(k))$.

The current version differs from our preliminary report [18] in several aspects. Most importantly, we present two definitions of locally-testable codes, whereas only the weaker one has appeared in [18]. Furthermore, we introduce a few modifications to the construction of Section 3.2 in order to obtain locally-testable codes under the stronger definition. More detailed comments appear in the relevant places.

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 $\mathcal{C} : \Sigma^k \to \Sigma^n$, and the elements of $\{\mathcal{C}(a) : a \in \Sigma^k\} \subseteq \Sigma^n$ are called codewords (of $\mathcal{C}$).

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 relative distance between u and v, denoted $\delta(u,v)$, is the ratio $\Delta(u,v)/n$. 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 $\mathcal{C} : \Sigma^k \to \Sigma^n$, denoted $\Delta_{\mathcal{C}}(w)$, is the minimum distance between w and the codewords; that is, $\Delta_{\mathcal{C}}(w) \stackrel{\text{def}}{=} \min_a\{\Delta(w, \mathcal{C}(a))\}$. 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) a third of the distance of the code.

Codeword Tests: two definitions

Loosely speaking, by a codeword test (for the code $\mathcal{C} : \Sigma^k \to \Sigma^n$) we mean a randomized (non-adaptive) oracle machine, called a tester, that is given oracle access to $w \in \Sigma^n$ (viewed as a function $w : \{1, ..., n\} \to \Sigma$). The tester is required to (always) accept every codeword and reject with (relatively) high probability every oracle that is "far" from the code. Indeed, since our focus is on positive results, we use a strict formulation in which the tester is required to accept each codeword with probability 1.[2] (This corresponds to "perfect completeness" in the PCP setting.)

[2]More generaly, one may allow codewords to be rejected with small probability. Note that this relaxation (w.r.t codewords) may be odd if coupled with the stronger definition regarding non-codewords (presented below).

The following two definitions differ by what is required from the tester in case the oracle is not a codeword. The weaker definition (which is the one that appears in our preliminary report [18]) requires that for every $w \in \Sigma^n$, given oracle access to w, the tester rejects with probability $\Omega(\Delta_{\mathcal{C}}(w)/n) - o(1)$. An alternative formulation (of the same notion) is that, for some function $f(n) = o(n)$, every $w \in \Sigma^n$ that is at distance greater than $f(n)$ from $\mathcal{C}$ is rejected with probability $\Omega(\Delta_{\mathcal{C}}(w)/n)$. Either way, this definition (i.e., Definition 2.1) effectively requires nothing with respect to non-codewords that are relatively close to the code (i.e., are at distance at most $f(n)$ from $\mathcal{C}$). A stronger and smoother definition (i.e., Definition 2.2) requires that *every* non-codeword w is rejected with probability $\Omega(\Delta_{\mathcal{C}}(w)/n)$.[3]

Definition 2.1 (codeword tests, weak definition): *A randomized* (non-adaptive) *oracle machine M is called a* weak codeword test for $\mathcal{C} : \Sigma^k \to \Sigma^n$ *if it satisfies the following two conditions:*

1. Accepting codewords*: For any $a \in \Sigma^k$, given oracle access to $w = \mathcal{C}(a)$, machine M accepts with probability 1. That is,* $\Pr[M^{\mathcal{C}(a)}(k,n,\Sigma)=1] = 1$*, for any $a \in \Sigma^k$.*

2. Rejection of non-codeword*: For some constant $c > 0$ and function $f(n) = o(n)$, for every $w \in \Sigma^n$, given oracle access to w, machine M rejects with probability at least $(c \cdot \Delta_{\mathcal{C}}(w) - f(n))/n$.*

   That is, $\Pr[M^w(k,n,\Sigma)=1] < 1 - ((c \cdot \Delta_{\mathcal{C}}(w) - f(n))/n)$*, for any $w \in \Sigma^n$.*

Definition 2.2 (codeword tests, strong definition): *A randomized* (non-adaptive) *oracle machine M is called a* strong codeword test for $\mathcal{C} : \Sigma^k \to \Sigma^n$ (or just a codeword test for $\mathcal{C} : \Sigma^k \to \Sigma^n$) *if it satisfies the following two conditions:*

- Accepting codewords*: As in Definition 2.1, for any $a \in \Sigma^k$, given oracle access to $w = \mathcal{C}(a)$, machine M accepts with probability 1.*

- Rejection of non-codeword*: For some constant $c > 0$ and for every $w \in \Sigma^n$, given oracle access to $w \in \Sigma^n$, machine M rejects with probability at least $c \cdot \Delta_{\mathcal{C}}(w)/n$.*

  That is, $\Pr[M^w(k,n,\Sigma)=1] < 1 - (c \cdot \Delta_{\mathcal{C}}(w)/n)$*, for any $w \in \Sigma^n$.*

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

Relation to property testing: Codewords tests are indeed a special type of property testers (as defined in [23, 16]). However, in the "property testing" literature one typically prefers to provide the tester with a *distance parameter* and require that the tester rejects all objects that are that far from the property with probability at least 2/3 (rather than with probability proportional to their distance). In such a case, the query complexity is measured as a function of the distance parameter and is constant only when the latter parameter is a constant fraction of the maximum possible distance. Strong codeword testers yield property testers with complexity that is inversely proportional to the distance parameter, whereas the complexity of testers derived from *weak codeword tests* is "well behaved" only for large values of the distance parameter.

[3]In both cases we have required that the rejection probability grows linearly with the distance of the oracle from the code. A more general treatment (e.g., requiring a polynomial relationship) makes sense too.

Our main results

Our main result regarding codes is

Theorem 2.3 *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 2.3 (as well as Part 2 of Theorem 2.4) vastly improves over the Hadamard code (in which $n = 2^k$), which is the only locally testable *binary* code previously known. Theorem 2.3 is proven by combining Part 1 of the following Theorem 2.4 with non-standard modifications of standard PCP constructions.

Theorem 2.4 (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 over* binary *alphabet such that $n < k^c$. Furthermore, the code is linear.*

In both cases, the codes have distance $\Omega(n)$.

Part 1 improves over the work of Friedl and Sudan [15], which only yields $n = k^{2+o(1)}$. We comment that (good) *binary* codes cannot be tested using two queries (cf. [9]).

The set of k's for which Theorems 2.3 and 2.4 hold is reasonable dense; in all 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) of Theorem 2.4, if k is in the set then the next integer in the set is smaller than $\exp((\log k)^{0.51}) \cdot k$ (resp., $O(\mathrm{poly}(\log k) \cdot k)$).

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 such 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 $\frac{1}{2}$.

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 the following:

Theorem 2.5 *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 [21] 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 2.4; that is, we describe codes mapping k bits of information to codewords of length k^c, for every $c > 1$, that are locally testable. Although we do not use any variant of the PCP Theorem, our constructions are related to known PCP constructions in the sense that we use codes and analysis that appear, at least implicitly, in PCP constructions. Specifically, we will use results regarding low-degree tests that were proven for deriving the PCP Theorem [2, 3]. We stress that we do not use the (complex) parallelization procedure of [2] nor the full power of the proof composition paradigm of [3], which is more complex than the classical notion of concatenated codes [14] used below.

We start by describing (in Section 3.1) a code over a large alphabet, which we refer to as the FS/RS code. This code, which is a direct interpretation of "low-degree tests", was proposed by Friedl and Sudan [15] and Rubinfeld and Sudan [23]. The size of this code turns out to be nearly quadratic in length (even after using the best possible analysis of low-degree tests). To reduce the size of the code to being nearly linear, we introduce a random truncation technique in Section 3.2. This establishes Part (1) of Theorem 2.4 (which refers to codes over large alphabets). In Sections 3.3 and 3.4 we apply the "code concatenation" technique to reduce the alphabet size of the codes, till we get a binary code, thus establishing Part (2) of Theorem 2.4.

3.1 The Basic Code (FS/RS-Code)

The FS/RS code is based on low-degree multi-variant polynomials over finite fields. We thus start with the relevant preliminaries. 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, $|P_{m,d}| = |F|^{\binom{m+d}{d}} = |F|^{\binom{m+d}{m}}$. (For $m \ll d$, we use $|P_{m,d}| < |F|^{\binom{2d}{m}} < |F|^{(2d/m)^m}$.)

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 $\{\ell_{a,b}(t) \stackrel{\text{def}}{=} ((a_1 + tb_1), ..., (a_m + tb_m)) : t \in F\}$.

The code. We consider a 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. We view L_m as the set of indices (or coordinates) in any $w \in \Sigma^{|L_m|}$; that is, for any $\ell \in L_m$, we denote by w_ℓ the symbol in w having index ℓ. In the code $\mathcal{C}$ that we consider, for every $p \in P_{m,d}$, the symbol of $\mathcal{C}(p)$ that is indexed by $\ell \in L_m$, 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 $q_{a,b}$.

Parameters. 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 $k \approx (d/m)^m$ 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 [2], where it is shown that (for $|F| = \text{poly}(d)$) if the oracle is ϵ-far from the code then this fact is detected with probability $\Omega(\epsilon)$.

We comment that in [2] the test is described in terms of two oracles: a point oracle $f : F^m \to F$ (viewed as the primary or "real" input) and a line oracle $g : L_m \to F^{d+1}$ (viewed as an auxiliary or additional oracle). Indeed, we will also revert to this view in our analysis. Unfortunately, using oracles having different range will complicate the code-concatenation (presented in Section 3.3), and this is the reason that we maintain explicitly only the line-oracle (and refer to the point-oracle only in the analysis). Note that a line-oracle can be used to define a corresponding point-oracle in several natural ways. For example, we may consider the (random) value given to each point by a random line passing through this point, or consider the value given to each point by a (fixed) *canonical* line passing through this point. (The former definition was used in our preliminary report [18], whereas the latter will be used below.)

3.2 Random Truncation of the FS/RS-Code

Our aim in this section is to tighten the relationship between k and n in locally testable codes. In order to get the best possible relation between n and k, one needs to use analyses of low-degree test that allows for $|F|$ to be as small as possible as compared to d. Based on the analysis of [21], [15] show that it suffices to use $|F| = \Theta(d)$. However, even with this best possible analysis, we are still left with n that is quadratic in $|F|^m$, whereas $k = o(d^m) = o(|F|^m)$. (This quadratic blowup comes from the fact that the number of lines (over F^m) is quadratic in the number of points, which in turn upperbounds the number of coefficients of a (generic) m-variant polynomial (over F).) Thus, to obtain n almost-linear in k, we must use a different code.

Overview of our construction: Our main idea here is to project the FS/RS code to a randomly chosen subset of the coordinates. Thus our code is essentially just a projection of the FS/RS code to a random subset of lines in F^m. This subset will have size that is almost-linear in $|F|^m$, and consequently the code will have almost-linear length. However, for technical reasons, we revert to the "point vs. lines" test (rather than using the "intersecting lines" test). Thus our encoding of a polynomial $p \in P_{m,d}$ ends up consisting of two parts:[4] One part simply giving the value of p on every point in F^m, and the other part giving the value of p restricted to a collection of lines, where the lines come from a randomly chosen subset, denoted $R_m \subset L_m$. Given a codeword c supposedly encoding some unknown polynomial p, the codeword test may now pick a random point from F^m and a random line from R_m that passes through this point and verifies that the restriction of p to this line (as asserted by the second part of c) agrees with the value of p on this point (as asserted by the first part of c).

While the code we use is essentially the one above, we modify it slightly to preserve uniformity in its alphabet size. As described so far, the first part of the code uses as its alphabet F, while the second part of the code needs the alphabet F^{d+1}. To make things more uniform, we view the

[4]Only the second part was used in our preliminary report [18]. The corresponding analysis (in [18]) was more complex (as well as more interesting), but yielded a weaker result.

first part also as elements of F^{d+1}. Syntactically this is obtained by repeating each field element (in the first part) for $d+1$ times. Semantically we view this $(d+1)$-sequence as giving the value of p restricted to a degenerate line that starts at the said point and has "slope" 0^m. The value of p restricted to this line ought to be a constant and we need to test this explicitly. So we augment the codeword test by performing both a "point vs. line" test and checking that a random degenerate line is assigned a constant function. Specifically, the value of a desired point (for the "point vs. line" test) is obtained from the corresponding degenerate line (i.e., from the polynomial assigned to this line); that is, in the "point vs. line" test we obtain two polynomials and compare the values assigned by them to two points. For sake of uniformly, we test whether this degenerate line is assigned a constant function by inspecting the values assigned by the corresponding polynomial to two points. Below we give the details of the code, the test, and the analysis.

The truncated code: In what follows, we will fix positive integers m, d and a field F. We will assume $\log\log d \leq m \leq \log d$, and $|F| = \Theta(d)$. Our code will be over the alphabet $\Sigma = F^{d+1}$ corresponding to the vector space of univariate polynomials of degree at most d over F. For the sake of concreteness, we will assume that a polynomial $p(x) = \sum_{i=0}^{d} c_i x^i$ is represented by the vector $\langle c_0, \ldots, c_d \rangle$. Let $L = L_m$ denote the collection of all lines in F^m. Let A denote the collection of degenerate lines; that is, the set $\{\ell_{a,0^m} \in L | a \in F^m\}$. For a (multi-)set $R \subseteq L$, let R' be the (multi-)set $R \cup A$. For such a set R, we define the code $\mathcal{C}^R : P_{m,d} \to \Sigma^{R'}$ as follows: We index the vectors in $\Sigma^{R'}$ by lines $\ell \in R'$. For $p \in P_{m,d}$ and $\ell \in R'$, the ℓth symbol in the encoding $\mathcal{C}^R(p)$ is the polynomial obtained by restricting p to the line ℓ.

Thus, our encoding is simply a projection of the FS/RS code to the coordinates in $A \cup R$, where A is the collection of degenerate lines, while R is an arbitrary subset of L. In what follows, we will show that if R is chosen uniformly at random (with replication from L) and $|R| = \Theta(m|F|^m \log|F|)$, then the code is locally testable. (To shorten our sentences we will simply say "R is chosen randomly" to say "the elements of the multi-set R are chosen uniformly at random from L".) We next describe the parameters of the code, and then describe the codeword test.

The basic parameters: We consider the information length k, the block length n and relative distance of the code. To compare k with n, let us consider the code as a mapping $\Sigma^k \to \Sigma^n$, where $n = |R'| = O(m|F|^m \log|F|)$ and (as before) $k = \log|P_{m,d}|/\log|\Sigma|$. We pick m and d so that $m = \Omega(\log\log d)$ but $m = o(\log d)$, and in such case we have $k \approx d^{m-1}/m^m$ and, for $|F| = O(d)$, we have $n = O(m|F|^m \log|F|) = O(d)^m$. 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 any 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 $\log|\Sigma| \approx d\log d \approx (\log k)^e$.

We next show that when $|F| = \Omega(d)$ and $|R|$ is chosen randomly of size $\Omega(m|F|^m \log|F|)$, the code has relative distance $\delta > 0$, with high probability. A straightfoward attempt to prove this fact is to upperbound the probability that a specific pair of codewords are too close and to use a union bound (over all possible pairs of codewords). This attempt fails, because the former probability is exponential (decreasing) in $|R|$ which is typically $|F|^{k^{1+o(1)}}$, whereas the number of pairs is exponential in $(|F|^k)^2$. However, if the code is "linear" in an adequate sense (as defined below) then it suffices to bound the weight of (non-zero) codewords, which in turn can be done by applying a union bound over all codewords (rather than over all pairs of codewords).

We say that a subset $C \subseteq \Sigma^n$, where $\Sigma \sim F^{d+1}$, is F-linear if it is a linear subspace of $(F^{d+1})^n$ when viewed as a vector space over F. In other words, for every $x, y \in C$ and $\alpha, \beta \in F$, it is the case that $\alpha x + \beta y \in C$, where $\alpha x = (\alpha x_1, \ldots, \alpha x_n)$ when $x = (x_1, \ldots, x_n) \in (F^{d+1})^n$ and αx_i denotes the usual scalar-vector product.

Proposition 3.1 *For every R, the code $\mathcal{C}^R$ is F-linear.*

Proof: For every $p', p'' \in P_{m,d}$ and $\alpha, \beta \in F$, and for every $\ell \in R'$, it holds that $(\alpha\mathcal{C}^R(p') + \beta\mathcal{C}^R(p''))_\ell$ equals $\alpha\mathcal{C}^R(p')_\ell + \beta\mathcal{C}^R(p'')_\ell$. Letting $p(\ell)$ denote the univariant polynomial representing the values of the polynomial p when restricted to the line ℓ, we have $\mathcal{C}^R(p')_\ell = p'(\ell)$ and $\mathcal{C}^R(p'')_\ell = p''(\ell)$. Finally,

$$\mathcal{C}^R(\alpha p' + \beta p'')_\ell \;=\; (\alpha p' + \beta p'')(\ell) \;=\; \alpha p'(\ell) + \beta p''(\ell) \;=\; \alpha\mathcal{C}^R(p')_\ell + \beta\mathcal{C}^R(p'')_\ell$$

where the second equality follows from the fact that $(\alpha p' + \beta p'')(x) = \alpha p'(x) + \beta p''(x)$ for every $x \in F^m$. Thus, $\mathcal{C}^R(\alpha p' + \beta p'') = \alpha\mathcal{C}^R(p') + \beta\mathcal{C}^R(p'')$, and the proposition follows. ■

We are now ready to analyze the relative distance of $\mathcal{C}^R$.

Proposition 3.2 *With probability $1 - o(1)$, for a randomly chosen R, the code $\mathcal{C}^R$ has relative distance at least $\delta = \Omega(1 - d/|F|) > 0$.*

Proof: Informally, the code $\mathcal{C}^L$ has relative distance at least $1 - d/|F|$; and so truncation to a random subset of coordinates should leave it with relative distance at least $\delta = \Omega(1 - d/|F|)$. Below, we formally prove this assertion for $\delta = \frac{1}{2} \cdot (1 - d/|F|)$, but the same argument can be used to establish $\delta = c \cdot (1 - d/|F|)$, for any constant $c < 1$.

Since the code $\mathcal{C}^R$ is F-linear (see Proposition 3.1), the distance between any two different codewords is captured by the weight of some non-zero codeword. Thus, it suffices to lowerbound the weight of all non-zero codewords in $\mathcal{C}^R$. Fix a non-zero polynomial $p \in P_{m,d}$, and consider the corresponding codeword $\mathcal{C}^R(p)$. Our aim is to prove that the probabilility that $\mathcal{C}^R(p)$ has relative weight less than δ is at most $o(|F_{m,d}|^{-1})$.

We first consider $\mathcal{C}^L(p)$. By the folklore property of multivariate polynomials, we have that p evaluates to non-zero values on at least $1 - d/|F|$ fraction of the points in F^m. Extending this fact to lines, we can infer immediately that the restriction of p to $1 - d/|F| = 2\delta$ fraction of the lines is non-zero. (This is true since one can sample a random line by picking a random point x and picking a random line through x, and if the p is non-zero at x, it must be non-zero on the line.) Also, we have that the restriction of p to A is non-zero on at least $1 - d/|F| > \delta$ fraction of the "lines" in A (since these lines are just points in disguise). So in order for $\mathcal{C}^R(p)$ to have fewer than δ fraction of non-zero coordinates, it must be that p is non-zero on fewer than δ fraction of the lines in R. But we also have that the expected fraction of lines in R where p is non-zero, when R is chosen at random, is at least 2δ. Applying Chernoff bound, we find that the probability that this fraction turns out to be less than δ when R is chosen at random, is at most $\exp(-\delta|R|) = o(|F|^{-|F|^m}) = o(|F_{m,d}|^{-1})$. Thus, the probabilility that $\mathcal{C}^R(p)$ has relative weight less than δ is at most $o(|F_{m,d}|^{-1})$. Taking the union bound over all possible polynomials p, we find the probability that $\mathcal{C}^R$ has a codeword of weight less than δ is at most $o(1)$. ■

We now move to describing the codeword test.

The Codeword Test: The test for the code $\mathcal{C}^R$ is a variant of the points-vs-lines test (cf. [2]) that accesses two oracles, one giving the value of a function $f : F^m \to F$ and the other supposedly giving the restriction of f to lines in F^m. The original test picks a random point $x \in F^m$ and a random line $\ell \in L$ passing through x and verifies that $f(x)$ agrees with its supposed restriction of f to the line ℓ. In implementing this test, we modify it in two ways: Firstly, we do not have the restriction of f to all lines, only to those in R (or rather to $R \cup A$), so we modify the above test by picking a random $\ell \in R$ that passes through x. Secondly, we do not (actually) have oracle access to the value of f on individual points, but rather the restriction of f to "degenerate lines" (where a word to be tested is allowed to provide arbitrary degree d polynomials as values of f restricted to these lines). So we test that the restriction of this word to degenerate lines is a constant, and use this constant as the value of the corresponding point. This leads to the following test that, given oracle access to $w \in \Sigma^{A \cup R}$, proceeds as follows:

1. Pick $x \in F^m$ uniformly at random and let $\ell_0 = \ell_{x,0^m} \in A$.
2. Pick $\ell \in R$ uniformly among lines that pass through x.

   (If no such line exists, set $\ell = \ell_0$.)
3. Let $h_0 = w_{\ell_0}$ and $h = w_\ell$.

   (Recall that h_0 and h are univariate degree d polynomials.)
4. Constant Test: Pick random $\alpha \in F$ and verify $h_0(0) = h_0(\alpha)$.
5. Point-vs-Line Test: Let $\beta \in F$ be such that $\ell(\beta) = x$. Verify $h_0(0) = h(\beta)$.

Note that the codeword test makes two queries to w (i.e., for w_{ℓ_0} and w_ℓ), and uses the answers in two different tests. Recall that, for sake of uniformity (with the Point-vs-Line Test), we avoid a seemingly simpler version of the Constant Test (in which one verifiers that h_0 is a constant polynomial). We analyze the codeword test next.

Analysis. It is obvious that the test accepts a valid codeword with probability 1. Below we give a lower bound on the rejection probabibility of non-codewords. As in Proposition 3.2, the lower bound holds for almost all possible R's.

Lemma 3.3 *For at least a $1 - o(1)$ fraction of the possible choices of R of size $n - |F^m| = O(m|F|^m \log |F|)$, every $w \in \Sigma^n$ is rejected by the codeword test with probability $\Omega(\delta_{\mathcal{C}^R}(w))$, where $\delta_{\mathcal{C}^R}(w)$ is the relative distance of w from the code $\mathcal{C}^R$.*

The above lemma improves over the probability bound $\Omega(\delta_{\mathcal{C}^R}(w)/n) - o(1)$ that was established in our preliminary report [18] for a related code (and a related test).

Proof: Let $w = (w', w'')$, where $w' \in \Sigma^{|A|}$ corresponds to the lines in A and $w'' \in \Sigma^{|R|}$ corresponds to the lines in R. Intuitively, the distance of w from $\mathcal{C}^R$ may be due to three sources:

1. The distance of w' from a sequence of constant polynomials. In this case, the Constant Test will cause rejection with probability that is linearly related to the said distance (becuase non-constant polynomials disagree with any constant polynomial on most points). For details see Claim 3.3.1.

2. The distance of the function determined by (the constant term of) the polynomials in w' from a low-degree polynomial. In this case, for all but a $o(1)$ of the choices of R, the Point-vs-Line Test will cause rejection with probability that is linearly related to the said distance. The proof of this claim (see Claim 3.3.2) is the most interesting part of the current analysis. It amounts to showing that, for most choices of R, the modified (Point-vs-Line) low-degree test that selects lines in R performs as well as the original low-degree test (which selects lines in L). The proof relies on the following observations:

   (a) Each possible function $f : F^m \to F$ determines an *optimal* answer for each possible line-query, which in turn assigns each possible query a "rejection value" that is merely the fraction of points on the line for which the (optimal) answer disagrees with the value of f.

   (b) The rejection probability of the original low-degree test is related to the average of these rejection values, *where the average is taken over all lines.*

   (c) The modified test refers to a (random) set of line-queries, and so its rejection probability is related to the average of these rejection values, *where the average is taken over the said set.*

   However, for a random set (of adequate size), with overwhelmingly high probability, the average of values assigned to elements in the set approximates the average of all values.

3. The distance of w'' from the values assigned to lines by the low-degree polynomial that is closest to the function determined by (the constant term of) the polynomials in w'. Suppose that the latter function f is actually a low-degree polynomial. Then, for all but a $o(1)$ of the choices of R, the Point-vs-Line Test will cause rejection with probability that is linearly related to the distance of "$\mathcal{C}^R(f)$ restricted to R" from w''. The claim can be extended to the general case in which f is only close to $P_{m,d}$; for details see Claim 3.3.3.

Turning to the actual proof, we fix some notation first. As above, we view $w \in \Sigma^n$ as indexed by lines in $A \cup R$, and write $w = (w', w'') \in \Sigma^{|A|} \times \Sigma^{|R|}$. We denote by $f_w : F^m \to F$ the function defined by letting $f_w(a) = w'_{\ell_{a,0^m}}(0)$; that is, $f_w(a)$ is the value at zero of the univariate polynomial that is assigned (by w') to the degenerate line passing through a. Let $p_w \in P_{m,d}$ denote the m-variate degree d polynomial closest to f_w (breaking ties arbitrarily). Let $\delta(w) = \delta_{\mathcal{C}^R}(w)$ be the (relative) distance of w from the code $\mathcal{C}^R$. In accordance to the motivational discussion, we consider the following auxiliary distances:

1. $\delta_{\mathrm{con}}(w)$ denotes the fraction of $a \in F^m$ such that $w'_{\ell_{a,0^m}}$ is not a constant polynomial.

2. $\delta_{\mathrm{ldp}}(w)$ denotes the relative distance of f_w from p_w (or equivalently from $P_{m,d}$).

3. $\delta_{\mathrm{agr}}(w)$ denotes the relative distance between the values assigned by p_w to lines in R and w''; that is, $\delta_{\mathrm{agr}}(w) = \Pr_{\ell \in R}[p_w(\ell) \neq w''_\ell]$, where (as above) $p_w(\ell_{a,b})$ denotes the univariant polynomial in $x \in F$ that is obtained from $p_w(a + xb)$.

For a polynomial $p \in P_{m,d}$ and a set of lines $S \subseteq L$, we let $\mathcal{E}^S(p)$ denote the sequence of univariant polynomials representing the values assigned by p to each line in S (i.e., $\mathcal{E}^S(p)_\ell = p(\ell)$ for every $\ell \in S$). Indeed, $\mathcal{C}^R(p) = \mathcal{E}^{A \cup R}(p)$. Using this notation, we have

$$\delta_{\mathrm{agr}}(w) \quad = \quad \frac{\Delta(\mathcal{E}^R(p_w), w'')}{|R|}$$

$$\delta_{\rm con}(w) \;=\; \frac{\Delta(\mathcal{E}^A(f_w),w')}{|A|}$$

$$\delta_{\rm ldp}(w) \;=\; \frac{\Delta(f_w,p_w)}{|A|} \;=\; \frac{\Delta(\mathcal{E}^A(f_w),\mathcal{E}^A(p_w))}{|A|}$$

Lastly, note that

$$(|A|+|R|)\cdot\delta(w) \;\le\; (\delta_{\rm con}(w)+\delta_{\rm ldp}(w))\cdot|A| + \delta_{\rm agr}(w)\cdot|R|$$

Thus, either $\delta_{\rm con}(w)+\delta_{\rm ldp}(w)\ge\delta(w)$ or $\delta_{\rm agr}(w)\ge\delta(w)$.

Claim 3.3.1 *For every $w\in\Sigma^n$, the Constant Test rejects w with probability at least* $(1-(d/|F|))\cdot\delta_{\rm con}(w)$.

Proof: Fix any $w\in\Sigma^n$. Then, with probability $\delta_{\rm con}(w)$, the point $x\in F^m$ selected by the test is such that the degree d univariant polynomial $h_0=w'_{\ell_{x,0^m}}$ is not a constant polynomial. In such a case, with probability at least $1-d/|F|$ over the choice of $\alpha\in F$, the value $h_0(\alpha)$ will disagree with $h_0(0)$, because the constant function $f_0(x)\stackrel{\rm def}{=}h_0(0)$ (for all $x\in F$) is also a degree d polynomial. In such a case, the Constant Test rejects. The claim follows. ■

For the next claim, we rephrase the Point-vs-Line test in terms of the associated functions $f:F^m\to F$ and $g:R\to\Sigma$, where in our application $f=f_w$ and $g(\ell)=w''_\ell$ (for every $\ell\in R$). The test now picks $x\in F^m$ uniformly at random and $\ell\in R$ uniformly among lines passing through x. For β such that $\ell(\beta)=x$, it verifies that $h(\beta)=f(x)$, where h is the univariant polynomial $g(\ell)$. Let $\delta_{\rm ld}(f)=\delta_{P_{m,d}}(f)$ denote the relative distance of f from $P_{m,d}$. Indeed, $\delta_{\rm ld}(f_w)=\delta_{\rm ldp}(w)$.

Claim 3.3.2 *For all but at most an $o(1)$ fraction of the possible choices of R, the following holds: For every $f:F^m\to F$ and $g:R\to\Sigma$, the probability that the Point-vs-Line Test* (of Step 5) *rejects the oracle pair (f,g) is at least $\Omega(\delta_{\rm ld}(f))$.*

In particular, we may conclude that our codeword test rejects any w with probability at least $\Omega(\delta_{\rm ldp}(w))$. Note that Claim 3.3.2 does not refer to the distance of g from being a legitimate line-oracle (let alone one that corresponds to f). Thus, Claim 3.3.2 effectively refers to all possible g's (or rather to the best possible g) that may be paired with f.

Proof: We prove the claim in two steps. First, we fix $f:F^m\to F$ and prove that for all but $\exp(-\delta_{\rm ld}(f)\cdot|R|)$ fraction of R's, the rejection probability of the test on input f and *any* $g:R\to\Sigma$ is $\Omega(\delta_{\rm ld}(w))$. Next, we use a union bound over an approriate collection of functions f, to prove that no function f is rejected with probability less than $\Omega(\delta_{\rm ld}(f))$. An interesting aspect of the second step is that we analyze the performance of the test on all functions by using a union bound only on a small fraction of the possible functions.

Step 1 – overview: Following [23, 2, 3, 21, 15], 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 point-vs-line test for oracle pairs $(f,\cdot)$ 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 point-vs-line test, when the point-oracle equals f, depends only on f (and not on the line-oracle g, which may not be optimal for f). Furthermore, this probability is the average of quantities (i.e., the agreements of f with the best univariate polynomials) that f associates with each of the possible lines. The latter fact holds not only when the test operates with the set of all lines, but also when it oprates with any set of lines R (as in the claim).[5] The key observation is that for a random set R, with overwhelmingly high probability, the average over R of quantities associated with lines in R approximates the average over L of the same quantities.

Step 1 – details: Fix $f : F^m \to F$ an let $\delta = \delta_{\rm ld}(f)$ denote its distance to the nearest low-degree polynomial. Let us denote by $D_\ell(f)$ the fractional disagreement of f, when restricted to line ℓ, with the best univariate polynomial (i.e., the univariate polynomial that is nearest to $f|_\ell$ (f restricted to ℓ)). Note that, on input oracles f and g, the rejection probability of the standard point-vs-line test (which refers to all possible lines), denoted $p_L(f,g)$, is lowerbounded by the average of the $D_\ell(f)$'s over all $\ell \in L$ (with equality holding if, for every line $\ell \in L$, it holds that $g(\ell)$ is a polynomial with maximal agreement with $f|_\ell$). A similar observation holds for the Point-vs-Line Test that refers to the set of lines R, except that now the average is taken over the lines in R. Furthermore, the average is weighted according to the probability that the test inpects the different lines (because a line is selected by uniformly selecting a point and then selecting a random line that passes through this point). Thus, the rejection probability of the Point-vs-Line Test that refers to the set R, denoted $p_R(f,g)$, is lowerbounded by the weighted average of the corresponding $D_\ell(f)$'s.

Using the best-known analysis of the standard low-degree test (in particular, using [15, Theorem 7] to include the case where $|F| = O(d)$), we obtain that

$$p_L(f,g) \geq \tau(f) \stackrel{\rm def}{=} |L|^{-1} \cdot \sum_{\ell \in L} D_\ell(f) = \Omega(\delta)\,.$$

(Actually, we only care about the second inequality.)[6] Now, when R is chosen at random (as a set of $n-|A|$ lines from L), the expected value of $\tau_R(f) \stackrel{\rm def}{=} |R|^{-1} \cdot \sum_{\ell \in R} D_\ell(f)$ equals $\mathbf{E}_\ell[D_\ell(f)] = \tau(f)$. By Chernoff bound, we have that the probability that R is such that $\tau_R(f) < \tau(f)/2$ is exponentially small in $\delta|R|$. That is, for a random set R of $n - |A|$ lines, it holds that

$$(\forall f) \quad \Pr_R[\tau_R(f) < \tau(f)/2] \ < \ \exp(-\Omega(\delta_{\rm ld}(f) \cdot |R|)) \tag{2}$$

In the following two paragraphs we assume that R is such that $\tau_R(f) \geq \tau(f)/2$.

Let us assume that R covers all points uniformly; that is, each point resides on the same number of lines in R (where several appearances on the same line are counted several times). This implies that our test selects lines uniformly in R. Then, the rejection probability of our test (i.e., the point-vs-line test for lines uniformly selected in R), when applied to f and any g, is lower-bounded by the (unweighted) average of the $D_\ell(f)$'s over the lines in R (rather than over the set of all lines, L). It follows that $p_R(f,g) \geq \tau_R(f) \geq \tau(f)/2 = \Omega(\delta_{\rm ld}(f))$. (Recall that $p_R(f,g)$ denotes the said rejection probability.)

Recall that in the previous paragraph we have assumed that R covers all points uniformly (i.e., each point resides on the same number of lines in R). In general, this may not be the case. Yet, with very high probability, a random set R covers *all* points in an *almost uniform* manner, and this "almost uniformity" suffices for extending the above analysis. Specifically, we first note that,

[5] In the latter case, the average is taken according to the distribution on R that is induced by the test. Note that this distribution is not necessarily uniform over R.

[6] The inequality $|L|^{-1} \cdot \sum_{\ell \in L} D_\ell(f) = \Omega(\delta)$ is only implicit in most prior works, but it can also be inferred from the results that are stated explicitly in them (which refer to the rejection probability of the standard test). Specifically, for the optimal g, the rejection probability of the standard point-vs-line test (which refers to all possible lines) equals the average of the $D_\ell(f)$'s (over all $\ell \in L$).

with overwhelmingly high probability, each point in F^m resides on $(1 \pm 0.1) \cdot |R|/|F|^{m-1}$ lines. Indeed, the probability that a particular point resides on a deviating number of lines is at most $\exp(-\Omega(|R|/|F|^{m-1})) = o(|F|^{-m})$, since $|R| \gg m \cdot |F|^{m-1} \log |F|$. Next observe that in the above analysis we assumed that the test selects lines uniformly in R, whereas our test selects lines in R by selecting uniformly a point and then selecting a random line passing through this point. However, as shown in the next paragraph, for R as above (i.e., that covers all points "almost uniformity"), the distribution induced on the selected lines assigns each line in R a probability of $(1 \pm 0.1)^{-1}/|R|$. Thus, the rejection probability may be skewed by a factor of $(1 \pm 0.1)^{-1} = (1 \pm 0.2)$ from the value $|R|^{-1} \cdot \sum_{\ell \in R} D_\ell(f) = \tau_R(f)$, which is analyzed above. We get $p_R(f,g) \geq 0.8 \cdot \tau_R(f) \geq 0.4 \cdot \tau(f) = \Omega(\delta_{\rm ld}(f))$. For future reference, we state the following fact (which, unlike the former inequality, does not assume $\tau_R(f) \geq \tau(f)/2$):

$$\Pr_R[(\forall f,g) \quad p_R(f,g) \geq 0.8 \cdot \tau_R(f) \text{ and } \tau(f) = \Omega(\delta_{\rm ld}(f))] = 1 - o(1) \tag{3}$$

It is left to analyze the distribution induced on lines selected from R (by the aforementioned process), when R covers all points "almost uniformity". For a point x, we let lines(x) denote the set of lines that pass through x. Then, the probability that the line $\ell = (x_1, ..., x_{|F|}) \in R$ is selected is given by

$$\sum_{i=1}^{|F|} \Pr[x_i \text{ is selected}] \cdot \frac{1}{|\text{lines}(x_i)|} \;=\; |F| \cdot \frac{1}{|F|^m} \cdot \frac{1}{(1 \pm 0.1) \cdot |R|/|F|^{m-1}}$$

which equals $(1 \pm 0.1)^{-1} \cdot |R|^{-1}$ as claimed.

Step 2 – overview: Our aim is to show that, for most R's, it is the case that $\tau_R(f) \geq \tau(f)/2$ holds for every f. This suffices to complete the proof of the current claim, because we have shown in Step 1 (see Eq. (3)) that $p_R(f,g) \geq \tau_R(f)$ and $\tau(f) = \Omega(\delta_{\rm ld}(f))$ for every pair (f,g). We would now like to take the union bound over all f's that are at distance δ from $P_{m,d}$ in order to upper bound the fraction of R's such that there exists such a function f for which $\tau_R(f) < \tau(f)/2$. The problem is that the number of such functions is $\binom{|F|^m}{\delta|F|^m} \cdot |P_{m,d}| > (d/m)^m$, whereas (for a random R) we only have $\Pr_R[\tau_R(f) < \tau(f)/2] < \exp(-\Omega(\delta|R|))$ (and in fact $\Pr_R[\tau_R(f) < \tau(f)/2] > \exp(-O(\delta|R|))$). This suffices in case $\delta|R| > (d/m)^m$, which in turn suffices to establish a weak tester (as per Definition 2.1),[7] but we wish to handle the general case (in order to establish a strong tester as per Definition 2.2). Thus, we cluster these functions according to the low-degree function that is closest to them, and show that it is enough to analyze one cluster (e.g., the one of the zero polynomial). The validity of the latter observation relies on properties of the set $P_{m,d}$ that imply that $D_\ell(f) = D_\ell(f+p)$ holds for every function f, polynomial $p \in P_{m,d}$ and line ℓ. The benefit in the said observation is that we need only consider the functions that are closest to some fixed polynomial and are at relative distance δ from it (rather than all functions at relative distance δ from $P_{m,d}$). Thus, we get an upperbound of $\binom{|F|^m}{\delta|F|^m} \cdot \exp(-\Omega(\delta|R|)$, which is negligible (since $|R| \gg |F|^m \log |F|^m$).

Step 2 – details: For any fixed $\delta > 0$, we start by considering the functions that are at relative distance δ from the zero polynomial. The number of such functions is at most

$$\binom{|F|^m}{\delta|F|^m} \;<\; (|F|^m)^{\delta|F|^m} \;=\; \exp(\delta m |F|^m \log |F|)\,.$$

[7]This is all that was established in our preliminary report [18], and the stronger analysis that follows is new.

On the other hand, by Eq. (2), for any such function f, it holds that $\Pr_R[\tau_R(f) < \tau(f)/2] = \exp(-\Omega(\delta_{\mathrm{ld}}(f)\cdot|R|))$, and if this function is closest to the zero polynomial (i.e., $\Delta(f,0) = \Delta_{P_{m,d}}(f)$) then $\delta_{\mathrm{ld}}(f) = \delta$. Thus, using $|R| = c\cdot|F|^m \log|F|^m$ (for an adequate constant c), the probability (over the choices of R) that there exists a function f that is closest to the zero polynomial and is at relative distance δ from it such that $\tau_R(f) < \tau(f)/2$ is upperbounded by

$$\exp(\delta m|F|^m \log|F|)\cdot\exp(-\Omega(\delta\cdot|R|)) \;=\; \exp(-2\delta|F|^m\log|F|^m) \;<\; o(|F|^{-m}),$$

where the last inequality uses $\delta \geq 1/|F|^m$. Summing over all (the $|F|^m$) possibilities of δ, we see that the probability over R, that there exists a function f that is closest to the zero polynomial (among all polynomials in $P_{m,d}$) such that $\tau_R(f) < \tau(f)/2$ is $o(1)$.

To conclude the argument, we use properties of the set $P_{m,d}$. Specifically, suppose that R is such that for every δ and every function f that is closest to the zero polynomial and at distance δ from this polynomial it holds that $\tau_R(f) \geq \tau(f)/2$. Now, consider an arbitrary function f' that is at distance δ from the set $P_{m,d}$, and let $p \in P_{m,d}$ be the polynomial closest to f'. Then, the function $f = f' - p$ is closest to the zero polynomial (and at distance δ from it), and we claim that $\tau(f) = \tau(f')$ and $\tau_R(f) = \tau_R(f')$. The reason being that, for every function f and polynomial $p \in P_{m,d}$ and for every line ℓ, it holds that $D_\ell(f) = D_\ell(f+p)$ (although the polynomials selected to achieve the maximum agreement with f and $f+p$, over the line ℓ, may be different). Indeed, if q is used to achieve the maximum agreement with f over the line ℓ then $q + (p|_\ell)$ will be selected to achieve the maximum agreement with $f+p$, where $p|_\ell$ is the univariant polynomial obtained by restricting the polynomial p to the line ℓ.

Thus, for R as above and for every f, we have $\tau_R(f) = \tau_R(f-p) \geq \tau(f-p)/2 = \tau(f)/2$, where $p \in P_{m,d}$ is closest to f. By Eq. (3), we have $p_R(f,g) \geq 0.8\tau_R(f)$ and $\tau(f) = \Omega(\delta_{\mathrm{ld}}(f))$ for every pair (f,g). Combining all the above, we get $p_R(f,g) \geq 0.8\tau_R(f) \geq 0.4\tau(f) = \Omega(\delta_{\mathrm{ld}}(f))$, and the current claim follows, ■

The last claim, which also relates to the Point-vs-Line Test, is also phrased in terms of the associated functions $f: F^m \to F$ and $g: R \to \Sigma$, where in our application $f = f_w$ and $g(\ell) = w''_\ell$ (for every $\ell \in R$).

Claim 3.3.3 *For all but at most an $o(1)$ fraction of the possible choices of R, the following holds: for every $f: F^m \to F$ and $g: R \to \Sigma$, the probability that the point-vs-line test rejects the oracle pair (f,g) is at least*

$$\frac{1}{2}\cdot\frac{\Delta(g,\mathcal{E}^R(p))}{|R|} - \frac{\Delta(f,p)}{|F^m|},$$

where p is the polynomial in $P_{m,d}$ that is closest to f.

Claim 3.3.3 will be applied to pairs (f_w, w''), in which case $\Delta(w'', \mathcal{E}^R(p_w)) = \delta_{\mathrm{agr}}(w)\cdot|R|$ and $\Delta(f_w, p_w) = \delta_{\mathrm{ldp}}(w)\cdot|F^m|$ (recalling that p_w is the polynomial closest to f_w). Consequently, we will infer that the codeword test reject any w with probability at least $(\delta_{\mathrm{agr}}(w)/2) - \delta_{\mathrm{ldp}}(w)$. Needless to say, this claim will be invoked only in case $\delta_{\mathrm{ldp}}(w) < \delta_{\mathrm{agr}}(w)/2$.

Proof: We first consider what happens when the test is invoked with oracle access to the pair (p,g), rather than to the pair (f,g). This mental experiment is easier to analyze in the case that the test selects lines uniformly in R (which is the case only when R covers all points uniformly). Still, as in the proof of Claim 3.3.2, we extend the analysis to most R (i.e., the R's that cover all points almost uniformly). The claim follows by observing that the test queries the point orcale on a single uniformly distributed point, and so replacing p by f may reduce the rejection probability by at most their relative distance.

As in the proof of Claim 3.3.2, we start by assuming that R covers all points uniformly (i.e., each point resides on the same number of lines in R). In this case, the test selects lines uniformly in R. Thus, with probability $\delta \stackrel{\rm def}{=} \Delta(g, \mathcal{E}^R(p))/|R_m|$, the test selects a line ℓ such that $h \stackrel{\rm def}{=} g(\ell)$ does not agree with p on ℓ. Now, since both h and $\mathcal{E}^R(p)_\ell$ (i.e., the values of p restricted to the line ℓ) are degree d univariant polynomials (and since they disagree), they disagree on at least $|F|-d > 2|F|/3$ of the points on ℓ. Thus, the test will reject the oracle pair (p,g) with probability at least $(2/3)\cdot\delta$. However, in general, R may not cover all points uniformly. Yet (as shown in the proof of Claim 3.3.2), with very high probability, a random set R covers *all* points in an almost uniform manner (i.e., each point in F^m resides on $(1 \pm 0.1)\cdot|R|/|F|^{m-1}$ lines). This "almost uniformity" suffices for extending the above analysis. Specifically, in this case each line is selected (by the test) with probability $(1\pm 0.2)/|R|$, and so the test rejects the oracle pair (p,g) with probability at least $0.8\cdot(2/3)\cdot\delta > \delta/2$.

So far we have analyzed the behavior of the test with respect to the oracle pair (p,g), whereas we need to analyze the behavior with respect to the oracle pair (f,g). Recalling the test makes a single uniformly distributed query to the point oracle, the claim follows. ■

Wrapping things up: Recall that either $\delta_{\rm con}(w) + \delta_{\rm ldp}(w) \geq \delta(w)$ or $\delta_{\rm agr}(w) \geq \delta(w)$. If $\delta_{\rm con}(w) \geq \delta(w)/4$ then invoking Claim 3.3.1 we are done. Similarly, if $\delta_{\rm ldp}(w) \geq \delta(w)/4$ then invoking Claim 3.3.2 we are done. It remains to deal with the case that both $\delta_{\rm con}(w) < \delta(w)/4$ and $\delta_{\rm ldp}(w) < \delta(w)/4$, which implies that $\delta_{\rm agr}(w) \geq \delta(w)$. But now, invoking Claim 3.3.3 and using $(\delta_{\rm agr}(w)/2) - \delta_{\rm ldp}(w) > \delta(w)/4$, the lemma follows. ■

Conclusion: By the above, with probability $1-o(1)$ over the choice of R, the code $\mathcal{C}^R$ has relative constant distance and is locally-testable (using two queries). Using the first parameter-setting (i.e., $d = m^m$), we establish Part 1 of Theorem 2.4.

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 $k' \stackrel{\rm def}{=} d+1$.

The inner-code: 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) $n' = |F|^{d'h} = \exp(d'\cdot(k')^{1/d'}\cdot\log|F|)$. Note that the inner-code has relative distance $(1-(d'/|F|)) > 3/4$.

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 $\overline{r} = (r_1^{(1)}, ..., r_h^{(1)}, ..., r_1^{(d')}, ..., r_h^{(d')}) \in F^{d'h}$ and $(s_1^{(j)}, ..., s_h^{(j)}) \in F^h$, and check whether $v_{\overline{r}} + v_{\overline{s}} = v_{\overline{t}}$, where

$$\begin{aligned}
\overline{s} &= (r_1^{(1)}, ..., r_h^{(1)}, ..., r_1^{(j-1)}, ..., r_h^{(j-1)}, s_1^{(j)}, ..., s_h^{(j)}, r_1^{(j+1)}, ..., r_h^{(j+1)}, ..., r_1^{(d')}, ..., r_h^{(d')}) \text{ and} \\
\overline{t} &= (r_1^{(1)}, ..., r_h^{(1)}, ..., r_1^{(j-1)}, ..., r_h^{(j-1)}, r_1^{(j)} + s_1^{(j)}, ..., r_h^{(j)} + s_h^{(j)}, r_1^{(j+1)}, ..., r_h^{(j+1)}, ..., r_1^{(d')}, ..., r_h^{(d')}) .
\end{aligned}$$

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').[8] The total-low-degree test uses $d'+2$ queries, and so our codeword test uses $3d' + d' + 2 = O(d')$ queries.

Lemma 3.4 *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: If $w' \in F^{|F^{d'h}|}$ (viewed as a function $w' : F^{d'h} \to F$) is at fractional distance at least $\min(\epsilon, 0.4)$ 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 $\min(\epsilon, 0.4) \cdot n'$ from such a polynomial, denoted p', which is unique (since $\min(\epsilon, 0.4) < (|F| - d)/2|F|$). By the hypothesis (regarding the distance of w' from $\mathcal{C}'$), this p' must be non-linear in some block of variables; that is, for some j, the polynomial p' is non-linear in $\{z_i^{(j)} : i \in [h]\}$. With probability at least $1-(d'/|F|) > 0.9$, this non-linearity is preserved when assigning *random* values to the variables of *all the other blocks*; that is, for a random $\overline{r} = (\overline{r}^{(1)}, ..., \overline{r}^{(d')}) \in (F^h)^{d'}$, with probability at least 0.9, the polynomial $q'(\overline{z}^{(j)}) \stackrel{\text{def}}{=} p'(\overline{r}^{(1)}, ..., \overline{r}^{(j-1)}, \overline{z}^{(j)}, \overline{r}^{(j+1)}, ..., \overline{r}^{(d')})$ is not linear, where $\overline{z}^{(j)} = (z_1^{(j)}, ..., z_h^{(j)})$. (Here we used the fact that p' has degree at most d'.)[9] Furthermore, q' (which also has degree at most d') is at distance $1 - (d'/|F|) > 0.9$ from any linear function (in $\overline{z}^{(j)}$). On the other hand, the expected fractional distance between the residual w' and p' under such a random assignment is less than 0.4. Thus, under such random assignment, the expected fractional distance of the residual w' from the set of linear functions in $\{z_i^{(j)} : i \in [h]\}$ is at least $0.9 \cdot 0.9 - 0.4 > 0.4$. 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). ■

The concatenated-code: The concatenated-code obtained by composing the outer-code $\mathcal{C} : \Sigma^k \to \Sigma^n$ with the inner-code $\mathcal{C}' : F^{k'} \to F^{n'}$, where $\Sigma = F^{d+1} = F^{k'}$, maps $(x_1, ..., x_k)$ to $(\mathcal{C}'(y_1), ..., \mathcal{C}'(y_n))$, where $(y_1, ..., y_n) \stackrel{\text{def}}{=} \mathcal{C}(x_1, ..., x_k)$. Thus, the concatenated-code maps $k \cdot k'$-long sequences over F to $n \cdot n'$-long sequences over F. Furthermore, the concatenated-code is linear (over F); that is, for each i, each F-symbol in the sequence $\mathcal{C}'(y_i)$ is a linear combination of the F-symbols in y_i, which in turn are linear combinations of the F-symbols in $(x_1, ..., x_k)$.

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

[9] Write the polynomial p' as the sum of monomials in $\overline{z}^{(j)}$ with coefficients being functions of the other variables. Consider any non-linear monomial in $\overline{z}^{(j)}$ having a non-zero coeffient (which is a polynomial of degree at most $d'-2$ in the other variables). Then, by the Schwarz–Zippel Lemma, with probability at least $1 - ((d'-2)/|F|)$, a random assignment to the other variables will yield a non-zero value, and thus this (non-linear) monomial in $\overline{z}^{(j)}$ will appear in q' (with a non-zero coefficient).

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 (as the tester of the outer-code) two intersecting lines ℓ' and ℓ'', and first applies the inner-code tester to the inner-encoding of the polynomials associated by the outer code to these two lines. To emulate the actual check of 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). (Note that by the modification introduced after the proof of Lemma 3.3, all that the outer-code tester does is to compare the values of univariant polynomials at certain points.) 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. Recall that q' is represented as a sequence of coefficients that, for sake of the inner-code, may be indexed by d'-tuples over $[h]$ such that $q'_{i_1,...,i_{d'}}$ is the coefficient of the $\sum_{j=1}^{d'}(i_j - 1) \cdot h^{j-1}$-th power; that is, $q'(z) = \sum_{i_1,...,i_{d'} \in [h]} q'_{i_1,...,i_{d'}} \cdot z^{\sum_{j=1}^{d'}(i_j-1)\cdot h^{j-1}}$. Observe that the value $q'(t)$ equals the entry of $\mathcal{C}'(q')$ that is associated with the sequence $(t^0, ..., t^{h-1}, t^0, ..., t^{(h-1)h}, ..., t^0, ..., t^{(h-1)h^{d'-1}})$. That is, we consider the entry of $\mathcal{C}'(q')$ that is indexed by 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

$$\sum_{i_1,...,i_{d'} \in [h]} q'_{i_1,...,i_{d'}} \cdot \prod_{j=1}^{d'} t^{(i_j-1)h^{j-1}} = \sum_{i_1,...,i_{d'} \in [h]} q'_{i_1,...,i_{d'}} \cdot t^{\sum_{j=1}^{d'}(i_j-1)h^{j-1}} = q'(t)\,.$$

Self-correction of the desired entry is performed via polynomial interpolation, and requires only $d' + 1$ queries (where each query is uniformly distributed). Thus, the concatenated code can be tested by making $O(d')$ queries: It is clear that our tester makes $2 \cdot (4d' + 2) + 2 \cdot (d' + 1)$ queries and that it accepts each codeword with probability 1, but the rejection probability of the tester does require a detailed analysis (which is provided next). The point is to prove that the composition of tests (for the concatenation-code) does work as one would have expected.

Lemma 3.5 *Let F and $\mathcal{C} = \mathcal{C}^R$ be as in Lemma 3.3, and $\mathcal{C}' : F^{k'} \to F^{n'}$ be as above, where $k' = h^{d'}$ and $n' = |F|^{d'h}$. Then, every $w \in F^{nn'}$ is rejected by the concatenated-code tester with probability that is linearly related to the distance of w from the concatenated-code.*

Proof: Let us denote by ϵ the relative distance of $w = (w_1, ..., w_n) \in (F^{n'})^n$ from the concatenated-code, and let $\epsilon_i \stackrel{\text{def}}{=} \Delta_{\mathcal{C}'}(w_i)/n'$ denote the relative distance of w_i from the inner-code $\mathcal{C}'$. Recall that each of the two lines selected by the outer-code tester is not uniformly distributed in $[n] \equiv A_m \cup R_m$, but it is rather that the first line is selected uniformly in $A_m \equiv [|F|^m]$ whereas the second line is selected uniformly in $R_m \equiv [n] \setminus [|F|^m]$. For a constant $c > 1$ (to be determined), we consider the following two cases:[10]

Case 1: either $|F|^{-m} \cdot \sum_{i=1}^{|F|^m} \epsilon_i > \epsilon/c$ or $(n - |F|^m)^{-1} \cdot \sum_{i=|F|^m+1}^{n} \epsilon_i > \epsilon/c$. In this case, at least one of the w_i probed by the outer-code tester is at expected relative distance at least ϵ/c from the inner-code. Thus, in this case, the inner-code tester will reject with probability $\Omega(\epsilon/c)$, which is $\Omega(\epsilon)$ because c is a constant.

Case 2: both $|F|^{-m} \cdot \sum_{i=1}^{|F|^m} \epsilon_i \le \epsilon/c$ and $(n - |F|^m)^{-1} \cdot \sum_{i=|F|^m+1}^{n} \epsilon_i \le \epsilon/c$. It follows that at least $1 - (\epsilon/2)$ of the w_i's are $(2/c)$-close to the inner-code. Denoting the closest corresponding

[10] Alternatively, instead of consisdering the two partial sums, one may consider a weighted average of all ϵ_i's, where the weights are proportional to the probability that the various lines are queried.

codewords by c_i's, we let d_i denote the decoding of c_i (and of w_i) if $\Delta_{\mathcal{C}'}(w_i, c_i) \leq 2n'/c$ and be arbitrary otherwise. Thus, the relative distance of $(d_1, ..., d_n)$ from the outer-code is at least $\epsilon-(\epsilon/c)-(\epsilon/2) > \epsilon/3$, where the ϵ/c term accounts for the average distance of the c_i's from the w_i's, and the $\epsilon/2$ term accounts for the arbitrary d_i's (which were introduced to facilitate the rest of the analysis). Thus, for some *constant* $c' > 0$ (determined in Lemma 3.3), the outer-code testes rejects $(d_1, ..., d_n)$ with probability at least $c' \cdot \epsilon/3$. (We will set $c = 24(d'+1)/c'$.) The question is what happens when the concatenated-code tester emulates the outer-code.

Recall that, in the current case, *both* the indices probed by the outer-code tester correspond to w_i's that are at expected relative distance at most ϵ/c from the inner-code. It follows that with probability at least $1 - 2 \cdot (2(d'+1)\epsilon/c)$, *both* indices probed by the outer-code tester correspond to w_i's that are at relative distance at most $1/2(d'+1)$ from the inner-code. In this case, with probability at least $(1-(d'+1)\cdot(1/2(d'+1)))^2 = 1/4$, both the self-corrected values (computed by our test) will match the corresponding d_i's. Note that if the above two events occur then our tester correctly emulates the outer-code tester. Thus, our tester rejects if the following three events occur:

1. The outer-code tester would have rejected the two answers (i.e., the two d_i's).
2. The two probed indices correspond to w_i's that are at relative distance at most $1/2(d'+1)$ from the inner-code (and in particular from the corresponding $\mathcal{C}'(d_i)$'s).
3. The self-corrected values match the corresponding d_i's.

By the above, Event 1 occurs with probability at least $c'\epsilon/3$, and Event 2 fails with probability at most $4(d'+1)\epsilon/c = c'\epsilon/6$ (by setting $c = 24(d'+1)/c'$). Thus, our tester rejects w with probability at least $((c'\epsilon/3) - (c'\epsilon/6)) \cdot (1/4) = \Omega(\epsilon)$, where the $1/4$ is due to the probability that Event 3 occurs (conditioned on Events 1 and 2 occuring).

Thus, in both cases, any word that is at relative distance ϵ from the concatenated-code is rejected with probability $\Omega(\epsilon)$. The lemma follows. ■

Other properties: Recall that the inner-code is linear (over F), and 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). The relative distnace of the concatenated code is the product of the relative distances of the outer and inner codes, and thus is a constant. Regarding the parameters of the concatenated code, suppose that in the outer-code we use the setting $d = m^e$ (for any constant $e > 1$), and that 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, $nn' \approx (kk')^{e/(e-1)}$, whereas the alphabet size is $|F| = O(d) \approx (\log k)^e$. For usage in the next subsection, we only care that the alphabet size is $k^{o(1)}$, while the rate is good (i.e., $nn' \approx (kk')^{e/(e-1)}$).

3.4 A Binary Code

The last step is to derive a binary code. This is done by concatenating the code presented in Section 3.3 with the Hadamard code, while assuming that $F = GF(2^{k''})$. That is, 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:

Fact 1: $\langle r, \sum_i \alpha_i a_i \rangle \equiv \sum_i \langle r, \alpha_i a_i \rangle \bmod 2$.

Fact 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.)

Specifically, let use denote by $\mathcal{C} : F^{kk'} \to F^{nn'}$ the code presented in Section 3.3, and let $\mathcal{C}'' : \{0,1\}^{k''} \to \{0,1\}^{n''}$ denote the suitable Hadamard code, where $F = GF(2^{k''}) \equiv \{0,1\}^{k''}$ and $[n''] \equiv \{0,1\}^{k''}$. Then, concatanating these two codes, we obtain a code that maps $(x_1, ..., x_{kk'}) \in (\{0,1\}^{k''})^{kk'}$ to $(\mathcal{C}''(y_1), ..., \mathcal{C}''(y_{nn'}))$, where $(y_1, ..., y_{nn'}) = \mathcal{C}(x_1, ..., x_{kk'})$. Recall that the codeword tester of $\mathcal{C}$ checks a constant number of linear conditions, each depending on a constant number of positions (i.e., F-symbols). By reducing the detection probability, we may assume that it actually checks only one such condition. Then we consider the following tester for the new code: The tester is given oracle access to $w = (w_1, ..., w_{nn'})$, where each $w_i = w_{i,1} \cdots w_{i,n''} \in \{0,1\}^{n''}$, and proceeds as follows:

1. The tester selects the locations $i_1, ..., i_q \in [nn']$ and the linear condition $(\alpha_1, ..., \alpha_q) \in F^q$ to be checked by the codeword tester of $\mathcal{C}$. (These choices are distributed as in the latter tester.)

2. For $j = 1, ..., q$, the tester checks that w_{i_j} is a codeword of $\mathcal{C}''$, by uniformly selecting $r, s \in \{0,1\}^{k''}$ and checking whether $w_{i_j,r} + w_{i_j,s} = w_{i_j,r \oplus s}$.

3. The tester emulates the check $\sum_{j=1}^q \alpha_j d_{i_j} = 0$ of the tester for $\mathcal{C}$, where d_{i_j} is the $\mathcal{C}''$-decoding of w_{i_j} and the arithmetic is over $F = GF(2^{k''})$. This will done by uniformly selecting $r \in \{0,1\}^{k''}$, and checking that $\langle r, \sum_{j=1}^q \alpha_j d_{i_j} \rangle = 0$. By Fact 1, we may check $\sum_{j=1}^q \langle r, \alpha_j d_{i_j} \rangle = 0$. To this end, we should obtain $\langle r, \alpha_j d_{i_j} \rangle$, for r and α_j that are known to us. As stated in Fact 2, the desired bit can be expressed as a linear combination (with fixed coefficients depending only on r and α_j) of the bits of d_{i_j}. That is, $\langle r, \alpha_j d_{i_j} \rangle = \langle r_j, d_{i_j} \rangle$, where r_j is determined by r and α_j. Recall that $\langle r_j, d_{i_j} \rangle = \mathcal{C}''(d_{i_j})_{r_j}$. However, since we may not have a valid codeword of d_{i_j}, we obtain the corresponding entry via self-correction of w_{i_j}. That is, we obtain (a good guess for) $\mathcal{C}''(d_{i_j})_{r_j}$, by using $w_{i_j,r_j \oplus s_j} - w_{i_j,s_j}$, for a uniformly selected $s_j \in \{0,1\}^{k''}$.

   To conclude, we uniformly select $r \in \{0,1\}^{k''}$, and determine $r_1, ..., r_q$ based on r and the α_j's (determined in Step 1). Next, we select uniformly $s_1, ..., s_q \in \{0,1\}^{k''}$, and check that $\sum_{j=1}^q (w_{i_j,r_j \oplus s_j} - w_{i_j,s_j}) = 0$.

Our tester makes $3q + 2q$ to the code, where q is a constant. It is clear that this tester accepts any valid codeword. The analysis of the rejection probability of non-codewords can be carried out analogously to Lemma 3.5 (i.e., considering two cases according to the average distance of the w_i's from valid codewords of $\mathcal{C}''$).[11] It follows that non-codewords are rejected with probability that is proportional to their distance from the code.

The final code maps $\{0,1\}^{kk'k''}$ to $\{0,1\}^{nn'n''}$, where $nn' \approx (kk')^{e/(e-1)}$ and $n'' = 2^{k''} = |F| = \mathrm{poly}(\log k) = k^{o(1)}$. Thus, $nn'n'' \approx (kk'k'')^{e/(e-1)}$. (Also note that the final code is linear and has linear distance.) This establishes Part 2 of Theorem 2.4.

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, $K \stackrel{\mathrm{def}}{=} kk'k'' \approx h^{(e-1)h} \cdot h^e \cdot \log h^e \approx h^{(e-1)h}$. 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 $1/\mathrm{poly}\log\log n$, 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 and improving results of Harsha and Sudan [19].

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 [3]. 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 [19]. Their verifier is a variant of the one constructed by Raz

[11] In the current setting we may treat all the code's coordinates uniformly. The constant c will be set depending on c' and q (i.e., $c = 4q/c'$), where c' is performance proportion established by Lemma 3.5. Thus, each self-correction will incur an error of $2\epsilon/c < c'\epsilon/2q$.

and Safra [22] (see also, Arora and Sudan [4]), which are, in turn, variants of a verifier constructed by Arora et al. [2]. 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, [19], which are abstracted below. Applying the composition lemmas used/developed in [19] 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 $\mathbf{E}_{r \leftarrow \Omega} [\mathrm{Verdict}(r, P_1(q_1(r)), P_2(q_2(r)))]$. 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 $\mathrm{Verdict}' : T \times A \times A \to \{0, 1\}$ 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 $\omega(V) \le \epsilon$, with probability at least $1 - \gamma$ it holds that $\omega(V|_{S,T}) \le 2\epsilon$.*

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 $\omega(V) \le \epsilon$, 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 $\mathbf{E}_{r\in\Omega|q_2(r)=q_2}[\text{Verdict}(r, P_1(q_1(r)), a_2)]$). We stress that this assertion holds both for the original 2IP verifier V as well as for any (S, Ω_S)-sampled verifier.[12] 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., $\epsilon_{q_2} = \mathbf{E}_{r\in\Omega|q_2(r)=q_2}[\text{Verdict}(r, P_1(q_1(r)), P_2^*(q_2))]$). 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 $\mathbf{E}_{r\leftarrow\Omega_S}[\text{Verdict}(r, P_1(q_1(r)), P_2(q_2(r)))] > (3/2)\epsilon$ 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.[13] ■

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 $L \in$* $\text{MIP}_{c,s}[p, r, a]$ (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 $P_1, \ldots, P_p$, 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, which is called the* soundness error.

Harsha and Sudan [19] presented a randomness efficient 3-prover one-round proof system with answer length $\text{poly}(\log n)$ and randomness complexity $(3 + \epsilon)\log_2 n$, where $\epsilon > 0$ is an arbitrary

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

[13] 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).

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 $\Pi \in \mathrm{MIP}_{1,\epsilon}[3, (1+1/m(n))\log n + O(m(n)\log m(n)), m(n)^{O(1)} n^{O(1/m(n))}]$.

Before proving this lemma, let us see some special cases obtained by setting $m(n) = \mathrm{poly}(\log\log n)$ 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 $\Pi \in \mathrm{MIP}_{1,\mu}[3, (1+1/p(\log\log n)) \cdot \log n, 2^{\mathrm{poly}(\log\log n)}]$ 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 $\Pi \in \mathrm{MIP}_{1,\mu}[3, (1 + O((\log\log n)/\sqrt{\log n})) \cdot \log n, 2^{O(\sqrt{\log n}\log\log n)}]$, 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 [19] improving it in two steps. The proof of [19] 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 [19] 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 [19], by applying a random truncation. To get this improvement we need to abstract the verifier of [19] (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 $r \in \Omega$, where Ω denotes the space of verifier's coins. 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 (i.e., $\Omega = \{0,1\}^{O(\log n)}$). 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

$$\Pr_r[q(r) \in Q' \wedge V_1(r, f(Q(r)), P_1(Q_1(r)))] < \frac{\epsilon}{2} \tag{4}$$

$$\Pr_r[q(r) \notin Q' \wedge V_2(r, f(Q(r)), P_2(Q_2(r)))] < \frac{\epsilon}{2} \tag{5}$$

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 $V_1(r_1, a, a_1) \wedge V_2(r_2, a, a_2)$.

Proposition 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

$$\Pr_{q, r_1, r_2}[EV_1(r_1) \wedge EV_2(r_2)] \tag{6}$$

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 (6) is equivalent to selecting each of them uniformly (though not independently). We thus upper bound (6) by

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

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 error of the verifier W_{S_1,S_2} is at most 6ϵ.*

Proof: We start with some notation: Recall that Ω denotes 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 X_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 Y_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 $r_1 \leftarrow Y_1$ and $r_2 \leftarrow Y_2$ (depending on the same random $q \in Q$). In our analysis, we will show that, for a random S_i, the distributions X_i and Y_i are statistically close, where as usual the statistical difference between X_i and Y_i is defined as $\max_{T \subseteq \Omega} \{\Pr_{r_i \leftarrow X_i}[r_i \in T] - \Pr_{r_i \leftarrow Y_i}[r_i \in T]\}$. We will then show that the verifier has low soundness error if it works with the distributions X_1 and X_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 and S_2):

BE1: The statistical difference between X_1 and Y_1 is more than ϵ.

BE2: The statistical difference between X_2 and Y_2 is more than ϵ.

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

$$\Pr_{r_1 \leftarrow X_1}[(q(r_1) \in Q') \wedge V_1(r_1, P(q(r_1)), P_1(q_1(r_1)))] > 2\epsilon.$$

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

$$\Pr_{r_2 \leftarrow X_2}[(q(r_2) \notin Q') \wedge V_2(r_2, P(q(r_2)), P_2(q_2(r_2)))] > 2\epsilon.$$

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 error. 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,

$$\begin{array}{ll}
\Pr_{(r_1,r_2)\leftarrow W_{S_1,S_2}}[V_1(r_1, P(q(r_1)), P_1(q_1(r_1))) \wedge V_2(r_2, P(q(r_2)), P_2(q_2(r_2)))] & \\
\quad \leq \Pr_{r_1\leftarrow Y_1}[(q(r_1) \in Q') \wedge V_1(r_1, P(q(r_1)), P_1(q_1(r_1)))] & \\
\qquad\quad + \Pr_{r_2\leftarrow Y_2}[(q(r_2) \notin Q') \wedge V_2(r_2, P(q(r_2)), P_2(q_2(r_2)))] & \\
\quad \leq \Pr_{r_1\leftarrow X_1}[(q(r_1) \in Q') \wedge V_1(r_1, P(q(r_1)), P_1(q_1(r_1)))] + \epsilon & \\
\qquad\quad + \Pr_{r_2\leftarrow X_2}[(q(r_2) \notin Q') \wedge V_2(r_2, P(q(r_2)), P_2(q_2(r_2)))] + \epsilon & [\neg\text{BE1 and } \neg\text{BE2}] \\
\quad \leq 6\epsilon & [\neg\text{BE3 and } \neg\text{BE4}]
\end{array}$$

Claim 4.9.1 *The probability of event BE1 (resp., BE2) is at most* $1/20$.

Proof: To estimate the statistical difference between X_i and Y_i, note that sampling r_i according to X_i is equivalent to the following process: select $r_i' \leftarrow X_i$ (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 X_i and Y_i equals $\frac{1}{2} \cdot \sum_{q\in Q} |\Pr_{r_i\leftarrow X_i}[q(r_i) = q] - \Pr_{r_i\leftarrow Y_i}[q(r_i) = q]|$, which in turn equals $\frac{1}{2} \cdot \sum_{q\in Q} \left|\Pr_{r_i\leftarrow X_i}[q(r_i) = q] - \frac{1}{|Q|}\right|$. 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

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

(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 $\Pr_{r\leftarrow X_i}[q(r) = q]$ 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 4.9.2 *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

$$\Pr_{S_1}\left[\exists P_1 \text{ s.t. } \Pr_{r_1\leftarrow X_1}[(q(r_1) \in Q') \wedge V_1(r_1, P(q(r_1)), P_1(q_1(r_1)))] \geq 2\epsilon\right] \leq \frac{1}{20}|A|^{-|Q|} \tag{7}$$

The claim will follow by a union bound over the $|A|^{|Q|}$ possible choices of P. For each 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, by Proposition 4.8, it holds that $\mathbf{E}_{q_1\in Q_1}[\epsilon_{q_1}] = \epsilon$. 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 (7). 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 error 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 4.10 *Let $s \stackrel{\text{def}}{=} O(|Q|\max\{\log|A|,\log|Q|\})$ and $t \stackrel{\text{def}}{=} O(|Q|\log|A| + |S_1|\log|A_1| + |S_2|\log|A_2|)$. Suppose that S_1 and S_2 are uniformly selected s-subsets of Q_1 and Q_2, and that T is a uniformly selected t-subset of $\{(r_1,r_2)|(q(r_1)=q(r_1)) \wedge (q_i(r_i) \in S_i, \forall i \in \{1,2\})\}$. Then, with probability at least $\frac{2}{3}$, the verifier $W_{S_1,S_2,T}$ has soundness error at most 7ϵ.*

Using Lemma 4.10, 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 4.10, 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 4.11 (cf. [4] or [8, 22]) *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 $\text{poly}\log\log\log n$. Then to terminate the recursion, we use the following result of [19].

Lemma 4.12 (Lem. 2.6 in [19]) *For every $\epsilon > 0$ and $p < \infty$, there exists a $\gamma > 0$ such that for every $r, a : \mathsf{Z}^+ \to \mathsf{Z}^+$,*

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

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

Theorem 4.13 (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 $\Pi \in \text{PCP}_{1,\frac{1}{2}+\epsilon}[(1 + O(1/\log\log n)) \cdot \log n, 16]$.*

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 $\Pi \in \text{PCP}_{1,\frac{1}{2}+\epsilon}[(1 + O(\log\log n/\sqrt{\log n})) \cdot \log n, 19]$.*

Part 2 implies Theorem 2.5.

Proof: The first part is obtained by starting with Corollary 4.6, and applying Lemma 4.11 twice to get a 9-prover MIP system with answer lengths $\text{poly}(\log\log\log n)$. Applying Lemma 4.12 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 4.11 thrice, obtaining a 12-prover MIP system with answer lengths $\text{poly}(\log\log\log n)$. Applying Lemma 4.12 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/RS-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. Using an off-the-shelve PCP (e.g., the one of [2]) this yields a *weak* locally testable code (i.e., one satisfying Definition 2.1); for details see Section 5.1. As we explain in Section 5.2, using an off-the-shelve PCP fails to provide a locally testable code (i.e., one satisfying Definition 2.2), and some modifications are required in order to redeem this state of affairs (as well as in order to obtain a linear code). Most of the current section will be devoted to implementing these modifications. Still, the easy derivation of the weak result (in Section 5.1) serves as a good warm-up.

5.1 Easy derivation of a weak result

Starting with the code $\mathcal{C}^R : \Sigma^k \to \Sigma^n$ presented in Section 3.2, we augment the code with small PCPs each asserting that a specific pair of polynomials that are provided for a pair of intersecting lines do assign the same value to the point of intersection. The number of such pairs of lines is $|R_m| \cdot |F| = \tilde{O}(|F|^{m+1})$, and each line is assigned a symbol in $\Sigma = F^{d+1}$. The size of each proof is polynomial in the length of the assertion (i.e., is polynomial in $\log_2 |\Sigma|$).

Recall that we are not willing to read the entire two inputs, but must rather perform a constant number of binary queries. Still, all standard PCP constructs (e.g., [13, 2]) can be extended to yield similar results when one is charge for orcale access to the input, which in turn is presented in a suitable error-correcting form. Actually, this property is stated explicitly in [5],[14] and is always referred to when using the PCP as an "inner-verifier" (in the setting of PCP composition). Indeed, we are using the PCPs here as an inner-verifier (but compose it with a codeword-tester rather than with an outer-verifier). Thus, we should actually replace each symbol in the $\mathcal{C}^R$-codeword with its encoding under the suitable code $\mathcal{C}'$ (which encodes m-bit strings by poly(m)-bit $\mathcal{C}'$-codeword). Finally, we should apply a suitable number of repetitions to the new n-sequence such that its length dominates the length of the added PCPs. For a suitable number of repetitions t, the resulting code maps $(x_1, ..., x_k) \in \Sigma^k$ to $((\mathcal{C}'(y_1), ..., \mathcal{C}'(y_n))^t, \pi_1, ..., \pi_r)$, where π_i is a PCP that refers to the i-th pair of intersecting lines and $r = |R_m| \cdot |F|$ is the number of such pairs. We set $t = \omega((\sum_{i=1}^r |\pi_i|)/(\sum_{i=1}^n |\mathcal{C}'(y_i)|))$, which is greater than $|F|$ because $r > n$ and each π_i is longer than each $\mathcal{C}'(y_i)$. The tester for the resulting code emulates the testing of $\mathcal{C}^R$ by inspecting the PCP that refers to the selected line. That is, when testing $(w_1, ..., w_{tn}, w_{tn+1}, ..., w_{tn+r}$, we select a pair of intersecting lines (ℓ_1, ℓ_2) (as done by the tester of $\mathcal{C}^R$), and invoke the PCP-verifier providing it with orcale access to the input-orcales w_{ℓ_1} and w_{ℓ_2} and proof-oracle $\pi_{(\ell_1,\ell_2)}$. In addition, we check that $w_{jt+\ell} = w_\ell$, for uniformly selected $\ell \in [n]$ and $j \in [t-1]$.

By our choice of t, the distance of the new code is determined by the distance of $\mathcal{C}^R$. Indeed, we should not pick t to be too large since this will hurt the rate; for example, $t = (\log n) \cdot (\sum_{i=1}^r |\pi_i|)/(\sum_{i=1}^n |\mathcal{C}'(y_i)|)$ will do. Thus, the block-length of the new code is

$$n' \stackrel{\text{def}}{=} (2\log n) \cdot r \cdot |\pi_1| = \tilde{O}(|R_m| \cdot |F| \cdot \text{poly}(\log_2 |\Sigma|)) = \tilde{O}(n \cdot |F| \cdot \text{poly}(d)).$$

Using $d = m^m$, we get $k' = k \cdot \log_2 |\Sigma| \approx m^{m^2-2m+m}$ and $n' = \tilde{O}(nd^{O(1)}) < m^{m^2+o(m)+O(m)}$, which yields $n' = \exp(\sqrt{\log k'}) \cdot k'$.

In analysing the above codeword test, we observe that if a sequence is ϵ-far from the new code then $\epsilon' = \epsilon - o(1)$ of the distance must be due to the first t blocks (of length n each) of the n'-bit long sequence. Thus, for constant $\epsilon > o(1)$, we can make assertions about this tn-bit long prefix of the sequence. We observe that either the first t blocks (of length n) are $\epsilon'/2$-far from being identical (to the first block) or the average distance of the first block from $\mathcal{C}^R$ is at least $\epsilon'/2$. Noting that the first case is detected with probability $\Omega(\epsilon)$ by the auxiliary ("repetition") test, we focus on the second case and consider what happens when invoking the PCP verifier. Indeed, standard PCP verifier will reject oracles that are far from the set of valid statements (in this case far from all codewords). Thus (substituting parameters), we obtain:

Theorem 5.1 (weak version of Theorem 2.3): *For every $c > 0.5$ and infinitely many k's, there exist* weak *locally testable codes with binary alphabet such that $n = \exp((\log k)^c) \cdot k = k^{1+o(1)}$. Furthermore, these codes have distance $\Omega(n)$.*

In contrast to Theorem 2.3, the codes asserted in Theorem 5.1 only have *weak* codeword tests (i.e., tests satisfying Definition 2.1). Furthermore, these codes are not necessarily linear.

5.2 Problems with an easy derivation of the strong result

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.

[14] In fact, the presentation of Babai et al [5] is in these terms.

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.[15] This problem was avoided in Section 5.1 by discarding non-codewords that are close to the code and making the PCPs themselves a small part of the codeword. Thus, non-uniqueness of decoding of PCPs by itself could not make the sequence too far from the code, and so nothing is required if we use the weak definition of testing. However, when we seek to achieve the stronger definition, this problem becomes relevant (and cannot be avoided).

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 an F-linear code (and an F-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.

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.

5.3 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 such 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 $\Sigma' = F^{\ell'}$

[15] 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).

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 5.2 *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.*[16] *Machine* Verdict *satisfies the following properties:*

  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 $i \le p$ 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 $\Pi \in (F^{\ell'})^N$,

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

[16] 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.

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 5.2 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 5.2 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 5.2. The first proposition merely serves as a warm-up towards the second one.

Proposition 5.3 *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).[17] 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 5.4 (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 $P : \mathcal{L}_{F,q \cdot b} \times (F^b)^q \to (F^a)^m$.*

[17] 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).

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 5.4, 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.[18]

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. ■

Whereas Proposition 5.4 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 5.4 (or alternatively only with Lemma 5.5 and Proposition 5.3), but it seems more convenient to (have and) work with both.[19]

Lemma 5.5 (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

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

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

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 $\epsilon\leq\delta_1\delta_2$. We need to show that there exist vectors $(x_1,\ldots,x_p)\in L$ such that $\Delta(E(x_i),X_i)\leq\epsilon n_1n_2$ 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

$$\begin{aligned}\frac{\Delta(E(x_i),X_i)}{n_1n_2} &\leq \frac{\Delta_1(E_1(x_i),Y_i)}{n_1\ell'}+\frac{\Delta_2(E_2(Y_i),X_i)}{n_1n_2}\\ &\leq \frac{p_1}{\gamma_1}+\mathbf{E}_{R_1}[\epsilon_2(R_1)]\end{aligned}$$

$$\leq \quad \frac{\epsilon}{3} + (\mathbf{E}_{R_1}[p_2(R_1)/\gamma_2] + p_1) \; < \; \epsilon$$

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). ■

5.4 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. [2]. 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 5.6 *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 linear combination v of the constraints of L, (i.e., picks a random vector $w \in F_2^{p\ell}$ and sets $v = w \cdot L$), 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. [2, 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 5.7 *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. [2] (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 5.7.

Overview of the proof of [2, Thm. 2.1.9]. We use the formalism of [19] to interpret the main steps in the proof of [2]. (In particular, whenever we refer to a step as "standard", such a step is performed explicitly in [19].) As a first step in their proof, Arora et al. [2] 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 [2], 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 [2] 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 5.7): 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 [2] (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:

$$g_0(z_1, \ldots, z_4, \alpha_1, \ldots, \alpha_4) \stackrel{\text{def}}{=} f(z_1)$$

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

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 [2], 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 [2].

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. ■

5.5 Combining all the constructions

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

Theorem 5.8 (Theorem 2.3, 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 5.5) the $(F_2, (p,\ell) \to (p+3, \text{poly}(\log p\ell)), 1/2, \gamma)$-linear inner verifier of Lemma 5.7 with the $(F_2, (p+3, \text{poly}(\log p\ell)) \to (p+4, 1), 1/2, \gamma)$-linear inner verifier of Lemma 5.6, 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 $r_1 = O(\log p'\ell') + 2(p'+3)\cdot \text{poly}(\log p'\ell') = \text{poly}(\log \ell')$ coins, encoding of length $n_1 = \text{poly}(\ell')\cdot \exp(\text{poly}(\log p'\ell'))$, and proofs of length $m_1 = \text{poly}(n_1)$.

Similarly, for any constant p, composing the verifier of Lemma 5.7 with V_1 (while setting $p' = p+3$ and $\ell' = \text{poly}(\log \ell)$),[20] 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 $r_2 = O(\log p\ell) + r_1 = O(\log \ell) + \text{poly}(\log\log \ell)$ coins, encoding of length $n_2 = \text{poly}(\ell)\cdot n_1 = \text{poly}(\ell)$, and proofs of length $m_2 = \text{poly}(n_2)$.

Our final step will be to compose (as per Proposition 5.4) the truncated version of the FS/RS-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/RS-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 5.4 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 $m_2 = \text{poly}(n_2) = \text{poly}(\ell) = \text{poly}(d)$, we get $M = N\cdot\exp(O(\log^c K)) = K\cdot\exp(O(\log^c K))$. The theorem follows. ■

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

5.6 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 5.9 *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 3.3 (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 5.9, we show that *our main result regarding locally testable codes* (i.e., Theorem 2.3) *holds also with tester that make only three queries.* The latter assertion is an immediate corollary of the following proposition.

Proposition 5.10 *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 5.9, 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),[21] 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:[22] 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

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

[22] 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.

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 5.10 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.

6 Subsequent Work 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, stated in our preliminary report [18], has been 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).

In our preliminary report [18] we also suggested the following seemingly easier goal of de-randomizing the linearity test of Blum, Luby and Rubinfeld [11]. Recall that in order 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 $G \times G$).[23] The challenge suggested in [18] was to present an explicit construction of such a set R.

The latter challenge as well as the more general goal of de-randomizing all our results were recently resolved by Ben-Sasson, Sudan, Vadhan and Wigderson [10]. Specifically, they showed that for low-degree testing one may use a small set of lines that consists of all lines going in a small set of directions. They also showed that this result suffices also for the derandomization of our PCP result.

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

[23]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 $f : G \to H$.

encoding the latter by the original code, preserves many properties of the code but does not necessarily preserve local-testability.[24]

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 $\ell(n) = \text{poly}(\log n) \cdot n$ (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?

Acknowledgments

We are grateful to Salil Vadhan for suggesting some modifications to the construction and analysis in Section 3.2 , yielding stronger results with simpler proofs.

[24]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] N. Alon, T. Kaufman, M. Krivelevich, S. Litsyn, and D. Ron. Testing Low-Degree Polynomials over GF(2). In *Proc. RANDOM 2003*, Lecture Notes in Computer Science, vol. 2754, pages 188-199, Springer, 2003.

[2] 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.

[3] 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.

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

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

[6] L. Babai, L. Fortnow, and C. Lund. Non-Deterministic Exponential Time has Two-Prover Interactive Protocols. *Computational Complexity*, 1: 3-40 (1991).

[7] M. Bellare, D. Coppersmith, J. Håstad, M. Kiwi, and M. Sudan. Linearity testing over characteristic two. *IEEE Transactions on Information Theory*, 42(6):1781-1795, November 1996.

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

[9] E. Ben-Sasson, O. Goldreich and M. Sudan. Bounds on 2-Query Codeword Testing. In the proceedings of *RANDOM'03*, Springer LNCS, Vol. 2764, pages 216–227, 2003.

[10] E. Ben-Sasson, M. Sudan, S. Vadhan and A. Wigderson. Randomness-efficient low degree tests and short PCPs via epsilon-biased sets. In *35th STOC*, pages 612–621, 2003.

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

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

[13] 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.

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

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

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

[17] 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.

[18] O. Goldreich and M. Sudan. Locally Testable Codes and PCPs of Almost-Linear Length. ECCC Report TR02-050, 2002.

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

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

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

[22] 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.

[23] 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. [12]). 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 [19], 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 [19]. 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 $d, k, s \le b(n)$ 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 [19]. Specifically, while [19] 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 [19, 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. ■