Strong Locally Testable Codes with Relaxed Local Decoders*

Oded Goldreich, Tom Gur, and Ilan Komargodski

Weizmann Institute of Science
Rehovot, Israel
{oded.goldreich,tom.gur,ilan.komargodski}@weizmann.ac.il

Abstract

Locally testable codes (LTCs) are *error-correcting codes* that admit very efficient codeword tests. An LTC is said to be strong if it has a *proximity-oblivious* tester; that is, a tester that makes only a *constant number* of queries and reject non-codewords with probability that depends solely on their distance from the code.

Locally decodable codes (LDCs) are complimentary to LTCs. While the latter allow for highly efficient rejection of strings that are far from being codewords, LDCs allow for highly efficient recovery of individual bits of the information that is encoded in strings that are close to being codewords.

Constructions of strong-LTCs with nearly-linear length are known, but the existence of a constant-query LDC with *polynomial* length is a major open problem. In an attempt to bypass this barrier, Ben-Sasson *et al.* (SICOMP 2006) introduced a natural relaxation of local decodability, called relaxed-LDCs. This notion requires local recovery of nearly all individual information-bits, yet allows for recovery-failure (but not error) on the rest. Ben-Sasson *et al.* constructed a constant-query relaxed-LDC with nearly-linear length (i.e., length $k^{1+\alpha}$ for an arbitrarily small constant $\alpha > 0$, where k is the dimension of the code).

This work focuses on obtaining strong testability and relaxed decodability *simultaneously*. We construct a family of binary linear codes of nearly-linear length that are both strong-LTCs (with one-sided error) and constant-query relaxed-LDCs. This improves upon the previously known constructions, which either obtain weak LTCs or require polynomial length.

Our construction heavily relies on *tensor codes* and PCPs. In particular, we provide *strong canonical* PCPs *of proximity* for membership in any linear code with constant rate and relative distance. Loosely speaking, these are PCPs *of proximity* wherein the verifier is proximity oblivious (similarly to strong-LTCs) and every valid statement has a unique *canonical* proof. Furthermore, the verifier is required to reject non-canonical proofs (even for valid statements).

As an application, we improve the best known separation result between the complexity of *decision* and *verification* in the setting of property testing.

1998 ACM Subject Classification F.1.3 [Computation by Abstract Devices] Complexity Measures and Classes

Keywords and phrases Locally Testable Codes, Locally Decodable Codes, PCPs of Proximity

Digital Object Identifier 10.4230/LIPIcs.CCC.2015.1

* This research was partially supported by the Minerva Foundation with funds from the Federal German Ministry for Education and Research, and by a grant from the I-CORE Program of the Planning and Budgeting Committee, the Israel Science Foundation and the Citi Foundation.

30th Conference on Computational Complexity (CCC'15).
Editor: David Zuckerman; pp. 1–41

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

1 Introduction

Locally testable codes (LTCs) are error-correcting codes that can be tested very efficiently. Specifically, a code is said to be an LTC if there exists a probabilistic algorithm, called a tester, that is given a *proximity parameter* $\varepsilon > 0$ and oracle access to an input string (an alleged codeword), makes a small number (e.g., $\mathsf{poly}(1/\varepsilon)$) of queries to the input and is required to accept valid codewords, and reject with high probability input strings that are ε-far from being a codeword (i.e., reject strings that disagree with any codeword on ε fraction of the bits). The systematic study of LTCs was initiated by Goldreich and Sudan [13], though the notion was mentioned, in passing, a few years earlier by Friedl and Sudan [8] and Rubinfeld and Sudan [20].

A natural strengthening of the notion of locally testable codes (LTCs) is known as strong-LTCs. While LTCs (also referred to as weak-LTCs) allow for a different behavior of the tester for different values of the proximity parameter, strong-LTCs are required to satisfy a strong *uniformity* condition over all values of the proximity parameter. In more detail, the tester of a strong-LTC does *not* get a proximity parameter as an input, and is instead required to make only a *constant* number of queries and reject non-codewords with probability that is related to their distance from the code. See [13, 10] for a discussion on both types of local testability. We note that from a property testing point of view, strong-LTCs can be thought of as codes that can be tested by a *proximity-oblivious tester* (see [12]).

The two most fundamental parameters of error-correcting codes (and strong-LTCs in particular) are the *distance* and the codeword *length*. Throughout this work we will only consider codes with constant relative distance, and so our main parameter of interest is the length, which measures the amount of redundancy of information in each codeword. By this criterion, constructing a strong-LTC with linear length (and constant relative distance) is the holy grail of designing efficient locally testable codes. Although recently some progress was made towards showing the impossibility of such linear length LTCs [5, 3], there are known constructions of strong-LTCs with relatively good parameters: Goldreich and Sudan [13] constructed a strong-LTC with constant relative distance and nearly-linear length, where throughout this paper a code of dimension k is said to have nearly-linear length if its codewords are of length $k^{1+\alpha}$ for an arbitrarily small constant $\alpha > 0$. Furthermore, recently Viderman [23] constructed a strong-LTC with constant relative distance and quasilinear length (i.e., length $k \cdot \mathsf{polylog}k$).

Another natural local property of codes is *local decodability*. A code is said to be a locally decodable code (LDC) if it allows for a highly efficient recovery of any individual bit of the message encoded in a *somewhat corrupted* codeword. That is, there exists a probabilistic algorithm, called a decoder, that is given a location i and oracle access to an input string w that is promised to be sufficiently close to a codeword. The decoder is allowed to make a small (usually constant) number of queries to the input w and is required to decode the i^{th} bit of the information that corresponds to the codeword that w is closest to. Following the work of Katz and Trevisan [17] that formally defined the notion of LDCs, these codes received much attention and found numerous applications (see e.g., [21, 25] and references therein). They are also related to *private information retrieval* protocols [4] (see [9] for a survey).

Despite much attention that LDCs received in recent years, the best known LDCs are of super-polynomial length (cf. [7], building on [24]). While the best known lower bound (cf. [17]) only shows that any q-query LDC must be of length $\Omega\left(k^{1+\frac{1}{q-1}}\right)$ (where k is the dimension of the code), the existence of a constant-query LDC with *polynomial* length remains a major open problem.

In an attempt to bypass this barrier, Ben-Sasson *et al.* [1] introduced a natural relaxation of the notion of local decodability, known as relaxed-LDCs. This relaxation requires local recovery of most (or nearly all) individual information-bits, yet allows for recovery-failure (but not error) on the rest. Specifically, a code is said to be a relaxed-LDC if there exists an algorithm, called a (relaxed) decoder, that has oracle access to an input string that is promised to be sufficiently close to a codeword. Similarly to LDCs, the decoder is allowed to make few queries to the input in attempt to decode a given location in the message. However, unlike LDCs, the relaxed decoder is allowed to output an abort symbol on a small fraction of the locations, which indicates that the decoder detected a corruption in the codeword and is unable to decode this specific information-bit. Note that the decoder must still avoid errors (with high probability).

Throughout this work, unless explicitly stated otherwise, when we say that a code is a relaxed-LDC, we actually mean that it is a relaxed-LDC with *constant* query complexity.

Ben-Sasson *et al.* [1] constructed a relaxed-LDC with nearly-linear length. More generally, they showed that for every constant $\alpha > 0$ there exists a relaxed-LDC (with constant relative distance) that maps k-bit messages to $k^{1+\alpha}$-bit codewords and has query complexity $O\left(1/\alpha^2\right)$. While these relaxed-LDCs are dramatically shorter than any known LDC, they do not break the currently known lower bound on LDCs (cf. [17]), and hence it is it still an open question whether relaxed-LDC are a strict relaxation of LDCs.

1.1 Obtaining Local Testability and Decodability Simultaneously

In this work, we are interested in short codes that are both (strongly) locally testable and (relaxed) locally decodable.[1] The motivation behind such codes is very natural, as the notion of local decodability is complimentary to the notion of local testability: The success of the decoding procedure of a locally decodable code is pending on the promise that the input is sufficiently close to a valid codeword. If the locally decodable code is also locally testable, then this promise can be verified by the testing procedure. However, recall that there are no known constant-query LDCs with even polynomial length, let alone such that are also locally testable. Hence, we focus on relaxed-LDCs.[2]

There are a couple of known constructions of codes that are both locally testable and relaxed decodable (with constant query complexity). Ben-Sasson *et al.* [1] observed that their relaxed-LDC can be modified to also be a weak-LTC (i.e., an LTC that is not strong), while keeping its length nearly-linear. However, the local testability of their code is inherently *weak* (see Section 1.3 for details). In a recent development, Gur and Rothblum [15] constructed a relaxed-LDC that is also a strong-LTC, albeit with *polynomial length*.

In this paper, we improve upon the aforementioned results of [1] and [15], achieving the best of both worlds. That is, we construct a code that is both a strong-LTC and a relaxed-LDC with *nearly-linear* length.

▶ **Theorem 1.1** (informal). *There exists a binary linear code that is a* relaxed-LDC *and a (one-sided error)* strong-LTC *with constant relative distance and nearly-linear length.*

[1] Note that although the notion of local testability and decodability are related, LTCs do not imply LDCs (i.e., there are LTCs that are not LDCs) and vice-versa. (See [18].)

[2] A different possible approach to solve this problem is to settle for codes with *long length*. Indeed, there are codes with *exponential* length that are both (constant-query) LDCs and LTCs, e.g., the Hadamard code. Another approach to solve this problem is to settle for codes with *large query complexity*. In a recent work, Guo, Kopparty, and Sudan [14] constructed very short length codes that are both locally testable and locally decodable, albeit with large (yet needless to say, sub-linear) query complexity.

A formal statement of Theorem 1.1 is given in Section 3. We remark that we actually prove a slightly stronger claim; namely, that any good linear code can be augmented (by appending additional bits to each codeword) into a code that is both a relaxed-LDC and a strong-LTC, at the cost of increasing the codeword length from linear to nearly-linear.

On Invoking Testers Prior to Decoders

Recall that for a code that is both locally testable and decodable, the promise (that the input is close to a codeword) required by the decoder can be eliminated by invoking the tester first. However, doing so can potentially hamper the decodability, since the tester is allowed to reject codewords that are only slightly corrupted. Fortunately, our tester is smooth (i.e., it queries each of the n bits of a codeword with probability $\Theta(1/n)$), and thus invoking the strong-tester a carefully chosen number of times (rejecting if one of the invocations rejected) will result in a tolerant tester (see [16, 19]). Such a tester will reject inputs that do not satisfy the promise of the decoder, yet still accept slightly-corrupted codewords (with high probability).

1.2 Strong Canonical PCPs of Proximity

The notion of PCPs of proximity plays a major role in many constructions of LTCs and relaxed-LDCs, as well as in our own. Loosely speaking, PCPs of proximity (PCPPs) are a variant of PCP proof systems, which can be thought of as the PCP analogue of *property testing*. Recall that a standard PCP is given explicit access to a statement (i.e., an input that is supposedly in some NP language) and oracle access to a proof (i.e., a "probabilistically checkable" NP witness). The PCP verifier is required to probabilistically verify whether the (explicitly given) statement is correct, by making few queries to the alleged proof. In contrast, a PCPP is given oracle access to a statement and to a proof, and is only allowed to make a small number of queries to both the statement and the proof. Since a PCPP verifier only sees a small part of the statement (typically, only a constant number of bits), it cannot be expected to verify the statement precisely. Instead, it is required only to accept correct statements and reject statements that are far from being correct (i.e., far in Hamming distance from any valid statement).

PCPs of proximity were first studied by Ben-Sasson *et al.* [1] and by Dinur and Reingold [6] (wherein they are called assignment testers). The main parameters of interest in a PCPP system for some language L are its *query complexity* (i.e., the total number of queries to the input and to the proof that the PCPP verifier makes in order to determine membership in L) and its proof *length*, which can be thought as measuring the amount of redundancy of information in the proof. Ben-Sasson *et al.* [1] showed a PCPP for any language in NP, with constant query complexity and nearly-linear length (in fact, the length is $n^{1+o(1)}$, where n is the length of the corresponding NP-witness).

As we have already noted, PCPPs have a central theoretical significance as the property testing analogue of PCP proof-systems. Moreover, PCPPs were shown to be useful in various applications, e.g., for PCP composition and alphabet reduction [1, 6], and for locally testable and locally decodable codes [1, 13, 15]. Further information regarding the latter application follows.

The notion of locally testable codes and PCPs of proximity are tightly connected. Not only that PCPPs (and PCPs in general) can be thought of as the computational analogue of the (combinatorial) notion of LTCs, but also any code can be made locally testable by using an adequate PCPP. Specifically, Ben-Sasson *et al.* [1] showed that any linear code can be

transformed to a (weak) LTC by appending each codeword with a PCPP proof that ascertains that the codeword is indeed properly encoded.[3] However, since there is no guarantee that every two different proofs for the same statement are far (in Hamming distance), in order to prevent deterioration of the distance of the code two additional steps are taken: Firstly, the appended PCPP proof should be uniquely determined per codeword (i.e., each codeword has a *canonical* proof), and secondly, each codeword is repeated many times so that the PCPP part constitutes only a small fraction of the total length.

The drawback of the foregoing approach is that it results in locally testable codes that are *inherently* weak (i.e., codes that do not allow for proximity-oblivious testing). To see this, note that PCPPs only guarantee that false assertions are rejected (with high probability), while true assertions can be accepted even if the proof is incorrect. Hence, corruptions in the PCPP part are not necessarily detectable and the *canonicity* of the PCPP proofs may not be verified, ruling out the possibility of a (strong) tester that is uniform over all possible values of the proximity parameter.[4] Moreover, when trying to build strong-LTCs, an additional problem that arises is that, by definition, PCPPs do not necessarily provide strong soundness, i.e., reject false proofs with probability that depends only on their distance from a correct proof.

Motivated by constructing *strong* locally testable codes, Goldreich and Sudan [13, Section 5.3] considered a natural strengthening of the notion of PCPPs, known as strong canonical PCPs of proximity (hereafter scPCPP), which addresses the aforementioned issues. Loosely speaking, scPCPP are PCPPs with *strong soundness* that are required to reject "wrong" proofs, even for correct statements. Moreover, they require that each correct statement will only have one acceptable proof. In more detail, scPCPP are PCPP with two additional requirements: (1) *canonicity*: for every true statement there exists a unique proof (called the canonical proof) that the verifier is required to accept, and any other proof (even for a correct statement) must be rejected, and (2) *strong soundness*: the scPCPP verifier is required to be proximity oblivious and reject any pair of statement and proof with probability that is related to its distance from a true statement and its corresponding canonical proof. A formal definition of scPCPPs can be found in Section 2.4.

Given a construction of adequate scPCPPs, the aforementioned strategy of appending each codeword with an efficient scPCPP (which ascertains membership in a code) will allow to transform any code to a strong-LTC. Unfortunately, unlike standard PCPPs, for which there are efficient constructions for any language in NP, there are no known constructions of general-purpose scPCPPs. Yet, Goldreich and Sudan constructed a mechanism, called *linear inner proof systems* (LIPS), which is closely related to some special-purpose scPCPPs. Loosely speaking, the LIPS mechanism allows to transform linear strong locally testable codes over a large alphabet into strong locally testable codes over a smaller alphabet (see [13, Section 5.2] for further details). By a highly non-trivial construction and usage of the LIPS mechanism, Goldreich and Sudan showed efficient constructions of strong-LTCs. Unfourtunately, their constructions do not meet our needs. Nevertheless, building upon their techniques, we show *strong canonical* PCPs *of proximity* with polynomial length for any good linear code.

[3] Note that membership in any linear code is in P, and so, the efficient PCPP for NP of Ben-Sasson *et al.* [1] can handle these statements.

[4] In contrast, note that for *weak* LTCs this problem can be ignored by simply making the PCPPs themselves a sufficiently small part of the codewords. Recall that *weak* LTCs are allowed to only work for values of the proximity parameter that are sufficiently large to ensure that the concatenation of a corrupted codeword and its corresponding PCPP sequence will include (significant) corruption in the codeword part. Thus, there is no need to verify the canonicity or even validity of the PCPP proof. However, when we seek to achieve the stronger definition of LTCs (i.e., strong-LTCs), this problem becomes relevant (and cannot be ignored).

▶ **Theorem 1.2** (scPCPP for good codes – informal)**.** *Let C be a linear code with constant relative distance and linear length. Then, there exists a* scPCPP *with polynomial proof length for membership in the set of all codewords of C.*

In fact, we actually prove a slightly stronger statement. Specifically, our scPCPPs satisfy two additional properties that will be useful for our main construction: The scPCPP proofs are linear (over GF(2)), and the queries that the verifier makes are roughly uniform. We remark that not only that the scPCPPs in Theorem 1.2 are crucial to our construction (see Section 1.4 for details), we also view these scPCPPs as interesting on their own. A formal statement of Theorem 1.2 and its proof are presented in Section 6.

1.3 Previous Works and Techniques

In this subsection, we survey the previous works and techniques regarding relaxed-LDCs upon which we build. We start by recalling the construction of the (nearly) quadratic length relaxed-LDC of Ben-Sasson *et al.* [1, Section 4.2]. The core idea that underlies their construction is to partition each codeword into three parts: The first providing the distance property, the second allowing for "local decodability", and the third ascertaining the consistency of the first two parts. The natural decoder for such a code will verify the consistency of the first two parts via the third part and decode according to the second part in case it detects no consistency error. Details follow.

Let C be any good linear code (i.e., a code with constant relative distance and linear length). Ben-Sasson *et al.* construct a new code C' whose codewords consist of three parts of equal length: (1) repetitions of a good codeword $C(x)$ that encodes the message x; (2) repetitions of the explicitly given message x; and (3) PCPPs that ascertain the consistency of each individual bit in the message x (which is explicitly given in the second part) with the codeword $C(x)$ (which is explicitly given in the first part). We remark that since the total length of the PCPPs is significantly longer than the statements they ascertain, the desired length proportions are obtained by the repetitions in the first two parts. Observe that the first part grants the new code C' good distance (although it may *not* be locally decodable), the second part allows for a highly efficient decoding of the message (at the cost of reducing the distance), and the third part is needed in order to guarantee that the first two parts refer to the same message. The (relaxed) decoder for C' will use the PCPPs in the third part in order to verify that the first part (the codeword $C(x)$) is consistent with the bit we wish to decode in the second part (the message x). If the PCPP verifier detects no error, the decoder returns the relevant bit in the second part; otherwise, it returns an abort symbol.

In order to implement the aforementioned relaxed-LDC, an adequate PCPP is needed; that is, an efficient PCPP for verifying the consistency of each individual bit in a message x with the codeword $C(x)$. We note that such statements are in P. Recall that Ben-Sasson *et al.* [1, Section 3] showed PCPPs with nearly-linear length for *any* language in NP. Hence, the consistency of each message bit with a codeword of C can be guarantied by a PCPP of length that is nearly-linear in the length of C. Since C' is obtained by augmenting a good linear code C with a single PCPP proof per every message bit (claiming consistency between that bit and the codeword of C), the length of C' is (nearly) quadratic (i.e., length $k^{2+\alpha}$ for an arbitrarily small constant $\alpha > 0$, where k is the dimension of the code). We note that Ben-Sasson *et al.* showed that the length of C' can be improved to nearly-linear by, roughly speaking, partitioning the message into blocks of various lengths and decoding based on a chain of consistent blocks.[5]

[5] To obtain length $k^{1.5+\alpha}$, the message is partitioned into $\sqrt{k}$ blocks, each of length $\sqrt{k}$. Then, the

Recall that *any* code can be transformed to a weak locally testable code by appending adequate PCPPs to it (See [1, Section 4.1]). Applying this transformation to the relaxed-LDC does not hamper the relaxed decodability of the code, and only increase its length by a moderate amount (since the PCPPs are of nearly-linear length); hence this transformation yields a (constant query) relaxed-LDC with nearly-linear length that is also a (weak) LTC. We stress that the aforementioned transformation yields local testability that is *inherently weak* due to the fact that it uses standard PCPPs. However, if the PCPPs in use were actually scPCPPs (of nearly-linear length), then the foregoing code would have been strongly testable.

In a recent work, Gur and Rothblum [15] constructed scPCPPs with polynomial length for the particular family of linear length statements that are needed for the [1] relaxed-LDC. By using these scPCPPs in the construction of [1], they obtained a relaxed-LDC that is also a strong-LTC, albeit with polynomial length (due to the length of their scPCPPs). While we conjecture it is feasible to construct nearly-linear length scPCPPs for P (which contains the family of statements we wish to have scPCPPs for) and even for unique-NP (also known as the class US),[6] we do not obtain such scPCPPs here. Instead, we take an alternative approach, which circumvents this challenge, as described in the next subsection.

1.4 Our Techniques

In this subsection, we present our main techniques and ideas for constructing a relaxed-LDC with nearly-linear length that is also a strong-LTC. Our starting point is the (*weakly* testable) relaxed-LDC construction of Ben-Sasson *et al.* [1]. However, we wish to replace the PCPPs that they use with scPCPPs, in order to achieve *strong* local testability.

Since we do not have general-purpose scPCPPs (let alone of near-linear length), we construct special-purpose scPCPPs that allow us to ascertain the particular statements we are interested in (see Theorem 1.2). It is crucial to note that the scPCPPs we are able to construct are with *polynomial* proof length (and not nearly-linear length, as we would have hoped). Recall that the statements that are needed for the construction of Ben-Sasson *et al.* (i.e., ascertaining the consistency of each bit of the message with the entire codeword for *decodability*, and ascertaining the validity of the codeword for *testability*) are linear in the length of the message. Therefore, applying our scPCPPs in a naive way (i.e., replacing the PCPPs in the construction of Ben-Sasson *et al.* with our scPCPPs) would yield codes with *polynomial* length, whereas we are aiming for nearly-linear length. Instead, we use an alternative approach.

The key idea is to provide scPCPPs that only refer to sufficiently short statements such that even with the polynomial blow-up of the scPCPP, the length of each proof would still be sub-linear. Specifically, instead of providing proofs for the validity of the entire codeword and the consistency of each message bit with the entire codeword (as in [1]), we provide proofs for the consistency of each message bit with "small" parts of the code and for the validity of

original message, as well as each of the smaller blocks is encoded by an error-correcting code. For each of the encoded smaller blocks, the following PCPPs are added: (1) a PCPP that ascertains the consistency of the encoded block with the encoded original-message; and (2) PCPPs that ascertains the consistency of each bit in the encoded block with the entire encoded block. Observe that the total encoding length decreased, since there are $\sqrt{k}$ proofs of statements of length $O(k)$ and k proofs of statements of length $O(\sqrt{k})$, thus, the total length is nearly-linear in $k^{3/2}$. By repeating this process, we can reduce the length of the code to $k^{1+\alpha}$ for an arbitrarily small constant $\alpha > 0$ (see [1, Section 4.2] for details).

[6] We note that the class unique-NP(i.e., the class of NP problems with unique witnesses) seems more likely to have scPCPPs than NP. This is because a language in NP may have many witnesses per instance, and it is not clear how to recognize the "canonical" NP-witness.

these small parts. If each part is sufficiently small (i.e., of length k^α for an arbitrarily small constant $\alpha > 0$, where k is the length of the message), then we can still obtain a code with nearly-linear length, even when providing polynomial length proofs for all of the small parts.

The caveat, however, is that proving that each message bit is consistent with a small part (or *local view*) of a codeword does *not* necessary imply that the message bit is consistent with the entire codeword. Similarly, partitioning a codeword into small parts and proving the validity of each part does not imply the validity of the entire codeword. Therefore, we need the base code (to which we append scPCPPs) to be highly structured so that, loosely speaking, the *local* consistency and validity we are able to ascertain can be used to enforce *global* consistency and validity. Concretely, the strategy we employ is using *tensor codes* and proving that this family of codes has features that allow us to overcome the aforementioned caveat. Details follow.

Given a linear code $C : \{0,1\}^k \to \{0,1\}^n$, the tensor code $C \otimes C : \{0,1\}^{k^2} \to \{0,1\}^{n^2}$ consists of all $n \times n$ matrices whose rows and columns are codewords of C. Similarly, the d-dimensional tensor code $C^{\otimes d} = \underbrace{C \otimes C \otimes \cdots \otimes C}_{\text{d times}} : \{0,1\}^{k^d} \to \{0,1\}^{n^d}$ is defined in the natural way. Namely, $C^{\otimes d}$ consists of all $\underbrace{n \times n \times \cdots \times n}_{\text{d times}}$-dimensional tensors such that each (axis-parallel) line in the tensor is a codeword of C.[7] (See Section 2.3 for the exact definitions.)

Towards obtaining relaxed local decodability, we show that tensor codes satisfy a feature, which we call *local propagation*, that allows us to verify *global* consistency statements (such as the ones that are used in the [1] relaxed-LDC) by verifying *local* consistency statements, which we can afford to prove with *polynomial* length scPCPPs; the *local propagation* feature of tensor codes is discussed in Section 4. Hence, we can ascertain that the value at each point in the tensor is consistent with the entire codeword by verifying the consistency of a constant number of randomly selected statements regarding small parts of the tensor (specifically, statements of consistency between the value at a point in the tensor and a line that passes through it). We remark that Theorem 1.2 can be used to derive polynomial-length scPCPPs for such statements (see Section 6). Therefore, we can replace the nearly-linear length PCPPs that are used in [1] with our polynomial length scPCPPs, while preserving the functionality of relaxed local decoding and keeping the total length of the construction nearly-linear. (See Section 4 for a more detailed high-level description of our approach, followed by a full proof in Section 4.2.)

Recapping, so far our construction is as follows. Let C be a good linear code and $d \in \mathbb{N}$ be a sufficiently large constant. Each codeword of our code consists of the following equal-length parts: (1) repetitions of the tensor codeword $C^{\otimes d}(x)$ that encodes the message x; (2) repetitions of the explicitly given message x; and (3) scPCPPs for small statements (specifically, regarding the consistency of each point in the tensor $C^{\otimes d}(x)$ with each line that passes through it), which are used to ascertain the consistency of each individual bit in the message x with the codeword $C^{\otimes d}(x)$.[8]

Finally, we augment the aforementioned construction with a forth and last part that allows us to obtain *strong* local testability. The naive approach is to append a scPCPP

[7] Axis-parallel lines in high-dimensional tensors simply generalize the notion of rows and columns in $n \times n$ matrices.

[8] We remark that the actual construction differs slightly from the above in that, for convenience, we use *systematic* tensor codes that contain the message explicitly in the encoding, instead of providing repetitions of the message as a part of the code.

that ascertains the validity of all three parts of our code. However, since the length of our scPCPPs is polynomial in the length of the statement, this approach would yield codes with long (polynomial) length. Instead, recall that we can (strongly) test the consistency of the first two parts via the third part (which is also strongly testable, since it is a scPCPP). Thus, in order to obtain strong local testability it suffices to ascertain that the first part is a valid codeword of $C^{\otimes d}$ using scPCPPs. Luckily, tensor codes also satisfy the *robustness* feature, which allows us to ascertain the validity of an entire codeword of $C^{\otimes d}$ by ascertaining the validity of small parts of the codeword. Detail follows.

Loosely speaking, a code is said to be *robust* if the corruption in a random "local view" of a codeword is proportional to the corruption in the entire codeword. In more detail, we use a recent result of Viderman [22] (building on [2]) that states that the corruption in a random 2-dimensional (axis-parallel) plane of a corrupted codeword of a binary tensor code $C^{\otimes d}$ (where $d \geq 3$) is proportional to the corruption in the entire codeword. This feature allows us to ascertain the validity of the first part (i.e., the tensor codeword $C^{\otimes d}(x)$) by only providing scPCPPs for short statements that refer to 2-dimensional planes in $C^{\otimes d}(x)$. (See Section 5 for a more detailed high-level description, followed by a full proof.)

1.5 Applications to Property Testing

As an application of our main result (Theorem 1.1) we improve on the best known separation result (due to [15]) between the complexity of *decision* and *verification* in the setting of *property testing*.

The study of property testing, initiated by Rubinfeld and Sudan [20] and Goldreich, Goldwasser and Ron [11], considers highly-efficient randomized algorithms that solve approximate decision problems, while only inspecting a small fraction of the input. Recently, Gur and Rothblum [15] initiated the study of $\mathcal{MA}$ proofs of proximity (hereafter $\mathcal{MAP}$s), which can be viewed as the NP analogue of property testing. They reduced the task of separating the power of property testers and $\mathcal{MAP}$s to the design of very local codes, both in terms of testability and decodability. Furthermore, they noticed that for such a separation, *relaxed decodability* would suffice.

Gur and Rothblum used several weaker codes to obtain weaker separation results than the one we obtain here. Specifically, they either show a smaller gap between the query complexity of testers and $\mathcal{MAP}$s, or show a separation for a limited range of the proximity parameter. In contrast, by plugging-in the code of Theorem 1.1, we obtain the best known (exponential) separation result between the power of $\mathcal{MAP}$s and property testers.

▶ **Theorem 1.3** (Informal). *There exists a property that requires $n^{0.999}$ queries for every property tester but has an $\mathcal{MAP}$ that uses a proof of* logarithmic *length and makes* poly$(1/\varepsilon)$ *queries.*

For more information regarding this application, we refer the reader to Section 7.

1.6 Organization

In Section 2 we provide the preliminaries. In Section 3 we describe the construction of the codes that establish Theorem 1.1. In Section 4 and Section 5 we establish the relaxed local decodability and strong local testability (respectively) of the codes. In Section 6 we construct the scPCPPs needed for our construction, and finally, in Section 7 we present an application of our codes for property testing.

2 Preliminaries

We start with some general notation. We denote by $[n]$ the set of numbers $\{1, 2, \ldots, n\}$. For $i \in [n]$ and for $x \in \{0,1\}^n$, denote by x_i the i^{th} bit of x. For $x, y \in \{0,1\}^n$, we denote by $\Delta(x,y)$ the Hamming distance between x and y, and denote by $\delta(x,y)$ the *relative* (Hamming) distance between x and y, i.e., $\delta(x,y) = \Delta(x,y)/n$. We say that x is δ-close to (respectively, δ-far from) y if the relative distance between x and y is at most δ (respectively, at least δ).

Given a set S, we denote by $s \in_R S$ the distribution that is obtained by selecting uniformly at random $s \in S$. For a randomized algorithm A, we write $\Pr_A[\cdot]$ (or $\mathbb{E}_A[\cdot]$) to state that the probability (or expectation) is over the internal randomness of the algorithm A.

(Non) Uniformity

Throughout this paper, for the simplification of the presentation, we formally treat algorithms (testers, decoders, and verifiers) as (non-uniform) *polynomial-size circuits*. We note, however, that all of our algorithms can be made uniform by making straightforward modifications. Furthermore, it will be convenient for us to view the length $n \in \mathbb{N}$ of objects as fixed. We note that although we fix n, it should be viewed as a generic parameter, and so we allow ourselves to write asymptotic expressions such as $\mathsf{poly}(n)$, $O(n)$, etc. In contrast, when we say that something is a constant, we mean that it is independent of the length parameter n.

2.1 Error Correcting Codes

Let $k, n \in \mathbb{N}$. A binary linear code $C : \{0,1\}^k \to \{0,1\}^n$ of distance d is a linear mapping over $\mathsf{GF}(2)$, which maps messages to codewords, such that the Hamming distance between any two codewords is at least $\mathsf{d} = \mathsf{d}(n)$. The relative distance of C, denoted by $\delta(C)$, is given by d/n. The length of a code is $n = n(k)$. By slightly abusing notation, we say that we can construct a code C with nearly linear length if for any constant $\alpha > 0$ we can construct a code $C\colon \{0,1\}^k \to \{0,1\}^n$, where $n = k^{1+\alpha}$. For any $x \in \{0,1\}^n$, denote the relative distance of x to the code C by $\delta_C(x) = \min_{y \in C}\{\delta(x,y)\}$.

We say that C is systematic, if the first k bits of every codeword of C contain the message; that is, if for every $x \in \{0,1\}^k$ and every $i \in [k]$ it holds that $C(x)_i = x_i$. Since C is a linear code, we may assume without loss of generality that it is systematic.

2.2 Local Testability and Decodability

Following the discussion in the introduction, *strong* locally testable codes are defined as follows.

▶ **Definition 2.1** (strong-LTC)**.** A code $C : \{0,1\}^k \to \{0,1\}^n$ is a strong-LTC, if there exists a probabilistic algorithm (tester) T that, given oracle access to $w \in \{0,1\}^n$, makes $O(1)$ queries to w, and satisfies:

1. Completeness: For any codeword w of C it holds that $T^w = 1$.
2. Strong Soundness: For all $w \in \{0,1\}^n$,

$$\Pr_T[T^w = 0] \geq \mathsf{poly}\big(\delta_C(w)\big).$$

We say that a tester makes *nearly-uniform* queries if it queries each bit in the (alleged) codeword input $w \in \{0,1\}^n$ with probability $\Theta(1/n)$.

Following the discussion in the introduction, *relaxed* locally decodable codes are defined as follows.

▶ **Definition 2.2** (relaxed-LDC)**.** A code $C : \{0,1\}^k \to \{0,1\}^n$ is a relaxed-LDC if there exists a constant $\delta_{\mathsf{radius}} \in (0, \delta(C)/2)$, a constant $\rho > 0$ and a probabilistic algorithm (decoder) D that, given oracle access to $w \in \{0,1\}^n$ and explicit input $i \in [k]$, makes $O(1)$ queries to w, and satisfies:

1. Completeness: For any $i \in [k]$ and $x \in \{0,1\}^k$ it holds that $D^{C(x)}(i) = x_i$.
2. Relaxed Soundness: For any $i \in [k]$ and any $w \in \{0,1\}^n$ that is δ_{radius}-close to a codeword $C(x)$,[9] it holds that

$$\Pr_D[D^w(i) \in \{x_i, \perp\}] \geq 2/3.$$

3. Success Rate: For every $w \in \{0,1\}^n$ that is δ_{radius}-close to a codeword $C(x)$, and for at least a ρ fraction of the indices $i \in [k]$, with probability at least $2/3$ the decoder D outputs the i^{th} bit of x. That is, there exists a set $I_w \subseteq [l]$ of size at least ρk such that for every $i \in I_w$ it holds that $\Pr_D[D^w(i) = x_i] \geq 2/3$.

We remark that our definition is slightly *stronger* than the one given in [1] as we require *prefect* completeness (i.e., that the decoder *always* outputs the correct value given oracle access to a valid codeword of the code C).

2.3 Tensor Codes

Tensor codes are defined as follows.

▶ **Definition 2.3.** Let $C : \{0,1\}^k \to \{0,1\}^n$ be a linear code. The tensor code $C \otimes C : \{0,1\}^{k^2} \to \{0,1\}^{n^2}$ is the code whose codewords consists of all $n \times n$ matrices such that each axis-parallel line (i.e., a row or a column) in the matrix is a codeword of C. Similarly, given $d \in \mathbb{N}$, the tensor code $C^{\otimes d} : \{0,1\}^{k^d} \to \{0,1\}^{n^d}$ is the code whose codewords consists of all d-dimensional tensors such that each axis-parallel line in the tensor is a codeword of C.

It is well-known that for every $d \in \mathbb{N}$ the tensor code $C^{\otimes d}$ is a linear code with relative distance $\delta(C)^d$ (see e.g., [2]). Given a message $x \in \{0,1\}^{k^d}$ and coordinate $\bar{\imath} = (\bar{\imath}_1, \ldots, \bar{\imath}_d) \in [n]^d$, we denote the value of $C^{\otimes d}(x)$ at coordinate $\bar{\imath}$ by $C^{\otimes d}(x)_{\bar{\imath}}$.

▶ Remark. By the definition of tensor codes, if a linear code C is systematic, then the tensor code $C^{\otimes d}$ is also a systematic code;[10] that is, for every $x \in \{0,1\}^{k^d}$ and $\bar{\imath} \in [k]^d$ it holds that $C^{\otimes d}(x)_{\bar{\imath}} = x_{\bar{\imath}}$.

Next, we provide notations for the restriction of tensors to lines and planes. We start by defining axis-parallel lines.

▶ **Definition 2.4** (Axis-Parallel Lines)**.** For $j \in [d]$ and $\bar{\imath} = (i_1, \ldots, i_d) \in [n]^d$, we denote by $\ell_{j,\bar{\imath}}$ the j^{th} axis-parallel line passing through $\bar{\imath}$. That is,

$$\ell_{j,\bar{\imath}} = \{(i_1, \ldots, i_{j-1}, x, i_{j+1}, \ldots, i_d)\}_{x \in [n]}.$$

We denote by $\mathsf{Lines}(n,d)$ the *multi-set* that contains all axis-parallel lines that pass through each point $\bar{\imath} \in [n]^d$.[11] That is, $\mathsf{Lines}(n,d) = \{\ell_{j,\bar{\imath}}\}_{\bar{\imath} \in [n]^d, j \in [d]}$. Lastly, given a tensor $w \in \{0,1\}^{n^d}$ we denote by $w|_{\ell_{i,j}} \in \{0,1\}^n$ the restriction of w to the line $\ell_{i,j}$, i.e., the j^{th} axis-parallel line that passes through $\bar{\imath}$.

[9] Note that since $\delta_{\mathsf{radius}} < \delta(C)/2$, for every $x \in \{0,1\}^n$ that is δ_{radius}-close to C there exists a *unique* codeword x' of C such that x is $\delta_{C'}(x)$-close to x'.

[10] We view the restriction of the tensor $C^{\otimes d}$ to the coordinates in $[k]^d$ as the prefix of $C^{\otimes d}$.

[11] Note that each axis-parallel line in $\{0,1\}^{n^d}$ appears n times in $\mathsf{Lines}(n,d)$.

Next, we define axis-parallel planes.

▶ **Definition 2.5** (Axis-Parallel (2-dimensional) Planes)**.** For $j_1 < j_2 \in [d]$ and $\bar{\imath} = (i_1, \ldots, i_d) \in [n]^d$, we denote by $\mathsf{p}_{j_1,j_2,\bar{\imath}}$ the $(j_1, j_2)^{\text{th}}$ axis-parallel plane passing through the point $\bar{\imath}$. That is

$$\mathsf{p}_{j_1,j_2,\bar{\imath}} = \{(i_1, \ldots, i_{j_1-1}, x_1, i_{j_1+1}, \ldots, i_{j_2-1}, x_2, i_{j_2+1}, \ldots, i_d)\}_{x_1,x_2 \in [n]}.$$

We denote by $\mathsf{Planes}(n,d)$ the *set* of all (distinct) axis-parallel planes in all directions in $\{0,1\}^{n^d}$.[12] Lastly, for a tensor $w \in \{0,1\}^{n^d}$ and a plane $\mathsf{p} \in \mathsf{Planes}(n,d)$ we denote by $w|_{\mathsf{p}} \in \{0,1\}^{n^2}$ the restriction of w to the coordinates in the plane p.

Throughout this work we deal with axis-parallel lines (respectively, axis-parallel planes); hence, for brevity, we will sometimes refer to an *axis-parallel line* (respectively, *axis-parallel plane*) simply as a *line* (respectively, *plane*). We remark that the multi-set $\mathsf{Lines}(n,d)$ contains $d \cdot n^d$ lines and the set $\mathsf{Planes}(n,d)$ contains $\binom{d}{2} \cdot n^{d-2}$ planes. We omit the parameters n and d when they are clear from the context.

Testing Tensor Codes

The next theorem, which is implicit in [22], shows that for every $d \geq 3$ and every linear code C, testing the tensor-code $C^{\otimes d}$ can be reduced to testing whether a random plane in C is a codeword of $C^{\otimes 2}$.

▶ **Theorem 2.6.** *Let C be a linear binary code and $d \geq 3$ an integer. Then, there exists a constant $c_{\mathsf{robust}} \in (0,1)$ such that for every tensor $w \in \{0,1\}^{n^d}$ it holds that*

$$\mathop{\mathbb{E}}_{\mathsf{p} \in_R \mathsf{Planes}} \left[\delta\left(w|_{\mathsf{p}}, C^{\otimes 2}\right)\right] > c_{\mathsf{robust}} \cdot \delta_{C^{\otimes d}}(w).$$

Specifically, in [22, Theorem A.5] it is shown that for $d \geq 3$, if a codeword w of a tensor code $C^{\otimes d}$ is corrupted, then the corruption in a random $(d-1)$-dimensional subplane of w is proportional to the corruption in the entire tensor w. By applying this result recursively (a constant number of times), we obtain Theorem 2.6. For completeness, we provide the proof of Theorem 2.6 in Appendix C.

2.4 PCPs of Proximity

Strong canonical PCPs of proximity were defined as follows in [13, Section 5.3].

▶ **Definition 2.7** (scPCPPs)**.** Let V be a probabilistic algorithm (verifier) that is given *oracle* access to an input $x \in \{0,1\}^n$ and *oracle* access to a proof $\pi \in \{0,1\}^{\ell(n)}$, where $\ell : \mathbb{N} \to \mathbb{N}$ satisfies $\ell(n) \leq \exp\left(\mathsf{poly}(n)\right)$. We say that V is a strong (canonical) PCPP verifier for language L if it makes $O(1)$ queries and satisfies the following two conditions:

- Canonical Completeness: For all $x \in L$, there exists a *unique* canonical proof for x, denoted $\pi_{\mathsf{canonical}}(x)$, such that the verifier always accepts the pair $(x, \pi_{\mathsf{canonical}}(x))$; i.e., $V^{x,\pi_{\mathsf{canonical}}(x)} = 1$.

[12] Unlike the *multi-set* $\mathsf{Lines}(n,d)$, which contains n copies of each line, there is no redundancy in the *set* $\mathsf{Planes}(n,d)$.

- Strong Canonical Soundness: For any input $x' \in \{0,1\}^n$ and proof $\pi' \in \{0,1\}^{\ell(|x|)}$ the verifier rejects with probability at least $\mathsf{poly}\big(\delta_{\mathsf{PCPP}}(x',\pi')\big)$, where

$$\delta_{\mathsf{PCPP}}(x',\pi') \triangleq \min_{x\in\{0,1\}^n}\left\{\max\left(\frac{\Delta(x,x')}{n}\ ;\ \frac{\Delta(\pi_{\mathsf{canonical}}(x),\pi')}{\ell(n)}\right)\right\}, \tag{1}$$

where for any $x \notin L$ we define $\pi_{\mathsf{canonical}}(x) = \lambda$ and say that any π' is 1-far from λ.

We say that a scPCPP verifier makes *nearly-uniform* queries if it queries each bit in the input x with probability $\Theta(1/|x|)$ and queries each bit in the proof $\pi(x)$ with probability $\Theta(1/|\pi|)$.

We stress that these scPCPPs have *one-sided error* (i.e., they always accept inputs in L coupled with their canonical proofs). Note that the *canonical* aspect is reflected in the dependence of $\delta_{\mathsf{PCPP}}(x',\pi')$ on $\Delta(\pi_{\mathsf{canonical}}(x),\pi')$, whereas the *strong-soundness* aspect is reflected in the tight relation between the rejection probability and $\delta_{\mathsf{PCPP}}(x',\pi')$.

3 The Main Construction

In this section we describe our construction of a family of binary linear codes that are both (constant-query) relaxed-LDCs and strong-LTCs with constant relative distance and nearly-linear length. Our codes rely heavily on special-purpose *strong canonical* PCPs *of proximity* (with polynomial proof length), which we construct in Section 6, and so, we start by stating these scPCPPs. Our first family of scPCPPs is for good linear codes.

▶ **Theorem 3.1** (scPCPPs for good codes). *Let $C : \{0,1\}^k \to \{0,1\}^n$ be a linear code with constant relative distance and linear length. Then, there exists a* scPCPP *for codewords of C (i.e., for the set $\{C(x)\}_{x\in\{0,1\}^k}$). Furthermore, the proof length of the* scPCPP *is* $\mathsf{poly}(n)$*, the* scPCPP *verifier makes nearly-uniform queries, and the canonical* scPCPP *proofs are linear (over* GF(2)*).*

As a corollary of Theorem 3.1, we obtain a family of scPCPPs for *half-spaces* of any good linear code. That is, scPCPPs that ascertain membership in the set of all codewords wherein one given location is set to a specific value (for example, all codewords that have 1 in their first location).

▶ **Theorem 3.2** (scPCPPs for half-spaces of good codes). *Let $C : \{0,1\}^k \to \{0,1\}^n$ be a linear code with constant relative distance and linear length. Let $i \in [k]$ be a location in a message and $b \in \{0,1\}$ a bit. Then, there exists a* scPCPP *for $C_{i,b}$, where $C_{i,b}$ is the set of all codewords w of C such that the i^{th}-bit of w equals b (i.e., $w_i = b$). Furthermore, the proof length of the* scPCPP *is* $\mathsf{poly}(n)$*, the* scPCPP *verifier makes nearly-uniform queries, and the* scPCPP *proofs are linear (over* GF(2)*).*

See Section 6 for the full proofs of Theorems 3.1 and 3.2. Equipped with the foregoing scPCPPs, we describe the construction of our code, which consists of three parts. (See Section 2 for relevant notation.)

Tensor code part

Let $C_0 : \{0,1\}^k \to \{0,1\}^n$ be a systematic linear code with linear length (i.e., $n = \Theta(k)$) and constant relative distance $0 < \delta(C_0) < 1$. Let $d \geq 3$ be a sufficiently large constant (to be determined later). Let $C \triangleq (C_0)^{\otimes d} : \{0,1\}^{k^d} \to \{0,1\}^{n^d}$ be the d-tensor product of C_0. By Section 2.3, since C_0 is systematic, then C is also systematic. Recall that $\delta(C) = \delta(C_0)^d$, hence $\delta(C)$ is a constant.

We augment the code C with scPCPPs that ascertain the validity of each *plane* in C (using Theorem 3.1) and scPCPPs that ascertain the consistency of each bit in C with each line that passes through it (using Theorem 3.2). Details follow.

Plane scPCPPs part

Let $C(x)$ be a codeword of the tensor code C. For every plane p in the tensor $C(x)$ we use our scPCPP*s for good codes* to prove that the restriction of $C(x)$ to the plane p (denoted by $C(x)|_{\mathsf{p}}$) is a codeword of $C_0^{\otimes 2}$. Specifically, for a codeword w of $C_0^{\otimes 2}$ we denote by $\pi_{\mathsf{plane}}(w)$ the corresponding canonical proof for the scPCPP verifier of Theorem 3.1. Then, for every message $x \in \{0,1\}^{k^d}$ we define $\pi_{\mathsf{planes}}(x)$ as the sequence of the canonical proofs for *all* planes in $C(x)$; that is,

$$\pi_{\mathsf{planes}}(x) = \{\pi_{\mathsf{plane}}(C(x)|_{\mathsf{p}})\}_{\mathsf{p}\in\mathsf{Planes}},$$

where Planes is the set of *all* (2-dimensional) axis-parallel planes in $\{0,1\}^{n^d}$ (see Definition 2.5).

We append $\pi_{\mathsf{planes}}(x)$ to the codeword $C(x)$. Note that

$$|\pi_{\mathsf{planes}}(x)| = \binom{d}{2} n^{d-2} \cdot |\pi_{\mathsf{plane}}(C(x)|_{\mathsf{p}})| \leq n^{d+O(1)}.$$

We stress that the constant in the $O(1)$ notation does *not* depend on d. These scPCPPs will be used for the local testability of our code (see Section 5).

Point-line scPCPPs part

Let $C(x)$ be a codeword of the tensor code C. For every point $\bar{\imath} = (i_1, \ldots, i_d) \in [n]^d$ and every direction $j \in [d]$ we use our scPCPP*s for half-spaces of good codes* to prove that the restriction of $C(x)$ to the line that passes through point $\bar{\imath}$ in direction j (denoted by $C(x)|_{\ell_{j,\bar{\imath}}}$) is a codeword of C_0 that is consistent with value of $C(x)$ at point $\bar{\imath}$.[13] Specifically, for a codeword w of C_0 and index $s \in [n]$ we denote by $\pi_{\mathsf{line}}(w,s)$ the canonical proof for the scPCPP verifier of Theorem 3.2 (which corresponds to codewords of C_0 whose s^{th}-bit equals to w_s). Then, for every message $x \in \{0,1\}^{k^d}$ we define $\pi_{\mathsf{lines}}(x)$ as the set of the canonical proofs for *all* lines passing through each point in $C(x)$; that is,

$$\pi_{\mathsf{lines}}(x) = \{\pi_{\mathsf{line}}(C(x)|_{\ell_{j,\bar{\imath}}}, i_j)\}_{\ell_{j,\bar{\imath}}\in\mathsf{Lines}},$$

where $\mathsf{Lines} = \{\ell_{j,\bar{\imath}}\}_{\bar{\imath}\in[n]^d, j\in[d]}$, as in Definition 2.4 (i.e., the set Lines contains all axis-parallel lines that pass through each point $\bar{\imath} \in [n]^d$).

We append $\pi_{\mathsf{lines}}(x)$ to the codeword $C(x)$. Note that $|\pi_{\mathsf{lines}}(x)| = d \cdot n^d \cdot |\pi_{\mathsf{line}}(C(x)|_\ell)| \leq n^{d+O(1)}$, where the constant in the $O(1)$ notation does *not* depend on d. These scPCPPs will be used for the relaxed local decodability of our code (see Section 4).

Putting it all together

Our construction is obtained by combining the tensor codeword $C(x)$ with $\pi_{\mathsf{lines}}(x)$ and $\pi_{\mathsf{planes}}(x)$, while ensuring that the three parts are of equal length. That is, for $k' = k^d$ define

[13] Note that the $\bar{\imath}^{\text{th}}$-bit of $C(x)$ is, in fact, the $i_j{}^{\text{th}}$-bit of the line $C(x)|_{\ell_{j,\bar{\imath}}}$.

$C' : \{0,1\}^{k'} \to \{0,1\}^{n'}$ as follows.

$$C'(x) \triangleq \left((C(x))^{t_1} \ , \ (\pi_{\mathsf{lines}}(x))^{t_2} \ , \ (\pi_{\mathsf{planes}}(x))^{t_3} \right)$$

where t_1, t_2 and t_3 are the *minimal* integers such that $|C(w)|^{t_1} = |\pi_{\mathsf{lines}}(w)|^{t_2} = |\pi_{\mathsf{planes}}(w)|^{t_3}$.[14]

Length and relative-distance of C'

For sufficiently large d the length of C' is nearly-linear. To this end, observe that for every $x \in \{0,1\}^{k^d}$ it holds that $|C(x)| = n^d$, $|\pi_{\mathsf{lines}}(x)| \leq \mathsf{poly}(n)$ and $|\pi_{\mathsf{planes}}(x)| \leq \mathsf{poly}(n^2)$. Hence, for every constant $\alpha > 0$, there exists some constant $d > 0$ so that

$$n' = n^{d+O(1)} = (O(1) \cdot k)^{d+O(1)} \leq (k')^{1+\alpha}.$$

The code C' has constant relative distance since the relative distance of C (denoted by $\delta(C)$) is constant, and since repetitions of C constitute a third of the length of C'; that is, $\delta(C') \geq \frac{\delta(C)}{3}$. In the next sections we prove the following theorem.

▶ **Theorem 1.1** (restated). *For every constant $\alpha > 0$, there exists some constant $d \geq 0$ so that the code $C' : \{0,1\}^{k'} \to \{0,1\}^{n'}$, as defined above, is a linear binary code that is a* relaxed-LDC *and a* strong-LTC *with constant relative distance.*

Specifically, in Section 4 we prove the relaxed-LDC feature of C', and in Section 5 we prove the strong-LTC feature of C'.

(Alleged) Codeword Notations

Consider an arbitrary string $w \in \{0,1\}^{n'}$ (which we think of as an alleged codeword). We view w as a string composed of three parts (analogous to the three parts of the construction above):

1. $\bar{c} = (c_1, \ldots, c_{t_1})$: the t_1 alleged repetitions of the tensor code part.
2. $\bar{p}^{\mathsf{lines}} = \left(\bar{p}_1^{\mathsf{lines}}, \ldots, \bar{p}_{t_2}^{\mathsf{lines}}\right)$: the t_2 alleged repetitions of the scPCPP proofs for all the point-line pairs (i.e., lines passing through all coordinates in all directions). For every $i \in [t_2]$, the string $\bar{p}_i^{\mathsf{lines}}$ consists of scPCPP proofs for every point-line pair, i.e., $\bar{p}_i^{\mathsf{lines}} = \{p_i^{\ell}\}_{\ell \in \mathsf{Lines}}$.
3. $\bar{p}^{\mathsf{planes}} = \left(\bar{p}_1^{\mathsf{planes}}, \ldots, \bar{p}_{t_3}^{\mathsf{planes}}\right)$: the t_3 alleged repetitions of the scPCPP proofs for all the (2-dimensional) planes. For every $i \in [t_3]$, the string $\bar{p}_i^{\mathsf{planes}}$ consists of scPCPP proofs for every plane, i.e., $\bar{p}_i^{\mathsf{planes}} = \{p_i^{\mathsf{p}}\}_{\mathsf{p} \in \mathsf{Planes}}$.

4 Establishing the Relaxed-LDC Property

In this section we prove that the code C', which was defined in Section 3, is a relaxed locally decodable code.

▶ **Theorem 4.1.** *The code $C' : \{0,1\}^{k'} \to \{0,1\}^{n'}$ is a* relaxed-LDC.

[14] Ignoring integrality issues, we can say that we "blow" the lengths of the two shorter parts to match the length of the longest part, which (in case of our implementation of the scPCPPs) is the part of the *plane* scPCPPs. Hence, actually, $t_3 = 1$.

In order to prove Theorem 4.1, it would be convenient to use an alternative definition of relaxed-LDCs, which implies the standard definition (Definition 2.2) by applying known transformations. Specifically, in Section D (following [1, Section 4.2]) we show that it suffices to relax the soundness parameter in Definition 2.2 to $\Omega(1)$ (instead of 2/3), and replace the *success rate* condition with the following *average smoothness* condition. Loosely speaking, *average smoothness* requires that the decoder makes nearly uniform queries on average (over all indices to be decoded). By the foregoing, to prove Theorem 4.1 it suffices to show that the code C' satisfies the following definition.

▶ **Definition 4.2** (*Modified* relaxed-LDCs)**.** A code $C : \{0,1\}^k \to \{0,1\}^n$ is a *modified* relaxed-LDC if there exists a constant $\delta_{\mathsf{radius}} \in (0, \delta(C)/2)$ and a probabilistic algorithm (decoder) D that, given oracle access to $w \in \{0,1\}^n$ and explicit input $i \in [k]$, makes $q = O(1)$ queries to w, and satisfies:

1. Completeness: For any $i \in [k]$ and $x \in \{0,1\}^k$ it holds that $D^{C(x)}(i) = x_i$.
2. Modified Relaxed Soundness: For any $i \in [k]$ and any $w \in \{0,1\}^n$ that is δ_{radius}-close to a codeword $C(x)$ it holds that

$$\Pr_D[D^w(i) \in \{x_i, \perp\}] = \Omega(1).$$

   where $\delta_{\mathsf{radius}} \in (0, \delta(C')/2)$, the decoding radius of C, is a universal constant, to be determined later.
3. Average Smoothness: for every $w \in \{0,1\}^n$ and $v \in [n]$,

$$\Pr_{i,j,r}\left[\mathcal{D}^w(i,j,r) = v\right] < \frac{2}{n},$$

   where $\mathcal{D}^w(i,j,r)$ denotes the distribution of the j^{th} query of the decoder D^w on coordinate i and coin tosses r, where the probability is taken uniformly over all possible choices of $i \in [k]$, $j \in [q]$, and coin tosses r.

We remark that in [1, Section 4.2], the definition of average smoothness also requires a matching lower bound, i.e., the decoder should satisfy $\frac{1}{2n} < \Pr_{i,j,r}\left[\mathcal{D}^w(i,j,r) = v\right] < \frac{2}{n}$. However, for our applications it suffices to only require the upper bound. We note that the lower bound can be easily obtained by adding (random) dummy queries.

We start by showing a decoder that satisfies the first two aforementioned conditions (i.e., the *completeness* condition and the *modified relaxed soundness*). Next, in Section 4.3 we show how to obtain a related decoder that also satisfies the *average smoothness* condition.

The Setting

Consider an arbitrary input $w \in \{0,1\}^{n'}$ such that $0 \leq \delta_{C'}(w) < \delta_{\mathsf{radius}}$. We view w as a string composed of three parts as in Section 3, i.e., $w = (\bar{c}, \bar{p}^{\mathsf{lines}}, \bar{p}^{\mathsf{planes}})$. We stress that any part of w might suffer from corruptions, and so, we have to be able to decode correctly assuming that not too many corruptions have occurred (i.e., less than δ_{radius} fraction). Denote by x the unique string such that w is $\delta_{C'}(w)$-close to $C(x)$ (see footnote 9).

High-Level Idea

Recall that a valid codeword of C' consists of three (repeated) parts: (1) a systematic tensor code C, (2) point-line scPCPPs, and (3) plane scPCPPs. Our general approach is to decode according to the prefix of the first part (which allegedly contains the message x explicitly (since we use a systematic code), and to use the second part to ensure that each bit in

message x is consistent with the rest of the (tensor) codeword $C(x)$. (The third part is not used here; it is only used for the testability of the code.) Thus, the task of (relaxed) decoding the i^{th} bit of the message is reduced to verifying that the explicitly given value of the i^{th} bit of the message is consistent with the rest of the codeword.

Towards this end, recall that the second part of each codeword contains scPCPPs that ascertain the consistency of each bit in the tensor with each line that passes through it, but not consistency with the entire tensor. Therefore, in order to verify the consistency of each message bit with the entire codeword, our decoder uses a feature of tensor codes, which we call *local propagation*. This feature allows us to verify the consistency of a single message bit with the entire codeword by verifying the consistency of a carefully chosen sequence of d point-line pairs (using the *point-line* scPCPP). Details follow.

Loosely speaking, the *local propagation* feature of tensor codes implies that if one corrupts a single point in a codeword and attempts to keep most local views (say, lines in the tensor) consistent with this corruption, then a chain of highly structured modifications must be made that causes the "corruption" to propagate throughout the entire tensor. This is best exemplified by our decoder, which is tailored to take advantage of the foregoing phenomena.

Our decoder is given a coordinate $\bar{\imath} = (i_1, \ldots, i_d) \in [k]^d$ and oracle access to an alleged codeword w as above. The decoder looks for "inconsistencies" in w and if it finds any, it outputs $\perp$. Otherwise, it simply output $w_{\bar{\imath}}$ (which should contain the $\bar{\imath}^{\text{th}}$ bit of the message). Since our base code C_0 has constant relative distance, in order to "corrupt" the point $\bar{\imath}$ in the tensor code without causing the lines that pass through $\bar{\imath}$ to be inconsistent with the corrupted value at $\bar{\imath}$, one has to corrupt a *constant fraction* of each line on which $\bar{\imath}$ resides. Thus, our decoder uses the scPCPPs to verify that a line ℓ that passes through $\bar{\imath}$ is consistent with the value at $\bar{\imath}$, assuring that a constant fraction of many lines on which $\bar{\imath}$ resides is corrupted.

Similarly, in order to "corrupt" a constant fraction of the line ℓ in the tensor codeword without causing inconsistency between the corrupted points in ℓ and the lines that pass through these corrupted points, one has to change a *constant fraction* of each line that passes through a corrupted point in ℓ (therefore, corrupting a *constant fraction* of each *plane* wherein the line ℓ resides). Thus, our decoder uses the scPCPPs to verify that the line that passes through a random point $\bar{\imath}'$ in ℓ (which is corrupted with probability $\Omega(1)$) is consistent with the value at $\bar{\imath}'$, assuring that a constant fraction of many planes on which line ℓ resides were corrupted.

Thus, if the $\bar{\imath}^{\text{th}}$ point of the tensor codeword (i.e., the bit we wish to decode) is *corrupted*, then by iteratively continuing this procedure d times, and only performing d point-line consistency tests, the decoder can detect the corruption in $\bar{\imath}$ with high probability, unless a large fraction of the codeword is corrupted (i.e., the corruption at a single point, $\bar{\imath}$, propagated to the entire tensor).

We remark that in the proof that C' is a relaxed-LDC we do not use the *strongness* and *canonicity* properties of the scPCPPs (they are only used to prove that C' is a strong-LTC). Furthermore, since in the following we only wish to present a decoder satisfies Condition 1 and 2 of Definition 4.2, we can allow the decoder to output a "don't-know" symbol whenever the codeword is corrupted.[15] Thus, we are not concerned with corruptions in the scPCPP

[15] Recall that the *completeness* condition of Definition 4.2 requires the decoder to successfully decode valid codeword, and the *modified relaxed soundness* condition requires that the decoder does not make a mistake in the decoding with probability at least $\Omega(1)$. However, the decoder is allowed to output a "don't-know" symbol with arbitrary probability on any (even on only slightly) corrupted codeword.

parts, since a corruption in these parts can only increase the rejection probability for strings that are not codewords. Regarding inputs that are legal codewords, there are no corruptions and hence, no "inconsistencies". Thus, for legal codewords our tester will always output the correct value.

4.1 Warm-up: Two-Dimensional Tensors

Before we proceed to prove Theorem 4.1, we sketch a proof for two-dimensional tensor codes; that is, when we set $d = 2$ in the construction that appears in Section 3. In this warm-up, towards the end of simplifying the presentation, we make the following assumptions: We omit the third part of the codeword (i.e., the plane scPCPPs), and we omit the repetitions of the first and second parts of the code (i.e., the tensor code, and the point-line scPCPPs) and assume instead that the lengths of the first and the second parts are equal. We note that both assumptions can be easily removed (see Section 4.2 for details).

Let $w = (c, p)$ be an alleged codeword that consists of two parts of equal length: (1) c, an alleged 2-dimensional tensor code $C_0^{\otimes 2} : \{0,1\}^{k^2} \to \{0,1\}^{n^2}$, and (2) p, a sequence of alleged scPCPPs for every pair of point $\bar{\imath}$ in $[n]^2$ and line ℓ in $C_0^{\otimes 2}$ that passes through $\bar{\imath}$; each scPCPP ascertains that the line ℓ is a codeword of C_0 that is consistent with the value at the point $\bar{\imath}$.

Given a point $\bar{\imath} = (i_1, i_2) \in [k]^2$, the decoder first runs the point-line scPCPP that corresponds to $\bar{\imath}$ and the line $\ell_{1,\bar{\imath}} = \{(x, i_2)\}_{x \in [n]}$ passing through $\bar{\imath}$ in direction "1" (i.e., parallel to the first axis), and outputs $\perp$ if the scPCPP verifier rejected. Otherwise, the decoder picks a random point $\bar{\imath}' = (i_1', i_2')$ on the line $\ell_{1,\bar{\imath}}$, runs the corresponding scPCPP for $\bar{\imath}'$ and the line $\ell_{2,\bar{\imath}'} = \{(i_1', x)\}_{x \in [n]}$ that passes through $\bar{\imath}'$ in direction "2", and output $\perp$ if the scPCPP verifier rejected. If none of the scPCPP verifiers rejected, the verifier outputs $c_{\bar{\imath}}$.

For the *completeness* condition, assume that the decoder is given a valid codeword. In this case, the first part is indeed a valid copy of $C_0^{\otimes 2}(x)$, and the second part consists of the canonical proofs for $C_0^{\otimes 2}(x)$. Hence, all of the scPCPP verifiers accept, and since $C_0^{\otimes 2}(x)_{\bar{\imath}} = x_{\bar{\imath}}$, the decoder succeeds in decoding $x_{\bar{\imath}}$.

For the (modified) *relaxed soundness* condition, assume that the decoder is given a corrupted codeword $w = (c, p)$ that is δ-close to a valid codeword $C_0^{\otimes 2}(x)$, where $\delta \leq \delta_{\mathsf{radius}}$ for a sufficiently small (constant) *decoding radius* δ_{radius}. Note that if $c_{\bar{\imath}} = x_{\bar{\imath}}$, then the decoder satisfies the soundness condition (since it always outputs either x_i or $\perp$); hence, we assume that $c_{\bar{\imath}} \neq x_{\bar{\imath}}$. In this case, when the decoder runs the scPCPP verifier for $\bar{\imath}$ and (the restriction of c to) $\ell_{1,\bar{\imath}}$ it does not reject (with high probability) only if $C|_{\ell_{1,\bar{\imath}}}$ is "close" to a codeword of C_0 that disagrees with c on $\bar{\imath}$ (since the $i_2{}^{\text{th}}$ bit of this codeword of C_0 must be different than $x_{\bar{\imath}}$). Since C_0 is a code with constant relative distance, this implies that a constant fraction of the line $\ell_{1,\bar{\imath}}$ must be corrupted (i.e., the restriction of c to the line $\ell_{1,\bar{\imath}}$ is $\Omega(1)$-far from its corresponding line in $C(x)$) for the scPCPP verifier to accept. Finally, if the decoder selected $\bar{\imath}'$ that is one of the $\Omega(n)$ corrupted points on $\ell_{1,\bar{\imath}}$, then by the same argument, a constant fraction points on the restriction of c to the line $\ell_{2,\bar{\imath}'}$ (that passes through $\bar{\imath}'$) must be corrupted. We deduce that in order to both scPCPP verifiers to accept (and hence defy the soundness condition), c must contain $\Omega(n^2)$ corrupted points, i.e., c should be β-far from $C_0^{\otimes 2}(x)$ for some constant β. By fixing $\delta_{\mathsf{radius}} < \beta$, we prevent this possibility.

The relaxed-LDC **Procedure for** C'

Input: a coordinate $\bar{\imath} \in [k]^d$ and an oracle access to a string $w = (\bar{c}, \bar{p}^{\mathsf{lines}}, \bar{p}^{\mathsf{planes}})$.

For $s \in [n]$ and $b \in \{0,1\}$ let $V_{s,b}$ be a scPCPP verifier that refers to an input of the form $z \in \{0,1\}^n$, and asserts that there exists $y \in C_0$ such that $z = y$ and $z_s = b$.

1. Choose a random copy of a tensor code $\mathbf{c}$ in $\bar{c}$ and a random copy of a set of point-line proofs $\bar{p}$ in $\bar{p}^{\mathsf{lines}}$. That is, choose uniformly at random $\mathsf{r} \in [t_1]$ and $\mathsf{r}' \in [t_2]$, and set $\mathbf{c} \triangleq c_{\mathsf{r}}$ and $\bar{p} \triangleq \{p^{j,\bar{\imath}}\}_{\bar{\imath} \in [n]^d,\, j \in [d]} \triangleq \bar{p}^{\mathsf{lines}}_{\mathsf{r}'}$.
2. Initialize a set of points P_1 to contain the singleton $\bar{\imath}$; i.e., $P_1 = \{\bar{\imath}\}$.
3. For $j = 1$ until $j = d$:
   a. Select uniformly at random a point $\bar{u} = (u_1, \ldots, u_d)$ from the set P_j.
   b. Verify that the j^{th}-axis-parallel line passing through $\bar{u}$ is a legal codeword of C_0 and that it is consistent with the value at $\mathbf{c}_{\bar{u}}$. That is, run the scPCPP verifier $V_{s,\mathbf{c}_{\bar{u}}}$, where $s \triangleq u_j$, with proof oracle $p^{j,\bar{u}}$ and input that consists of the j^{th}-axis-parallel line passing through $\bar{u}$ in $\mathbf{c}$. In other words, we run $V_{s,\mathbf{c}_{\bar{u}}}$ on input $\mathbf{c}|_{\ell_{j,\bar{u}}}$ and proof $p^{j,\bar{u}}$.
   c. If V rejects, output $\perp$ and halt.
   d. If $j < d$, fix P_{j+1} to be a set of points in $[n]^d$ that reside on the j^{th}-axis-parallel lines passing through the points in P_j. That is, $P_{j+1} = \{\ell_{j,\bar{z}}\}_{\bar{z} \in P_j}$, where $\ell_{j,\bar{z}}$ (defined in Definition 2.4) is the j^{th} axis-parallel line passing through the point $\bar{z}$.
4. Query $\mathbf{c}_{\bar{\imath}}$ and return its value.

Figure 1 Relaxed local decoder D for C'.

4.2 The General Case

We proceed with the full proof that C' has a decoder that satisfies the first two conditions in the definition of a relaxed-LDC (i.e., the *completeness* and *(modified) relaxed soundness* conditions of Definition 4.2). We generalize the decoder of Section 4.1 to d-dimensional tensors and ensure it works without the assumptions that were made there for simplicity. The decoder D is formally described in Figure 1.

Let $\bar{\imath} \in [k]^d$. The *completeness* of the decoder is immediate from the construction: If the input is a codeword, i.e., $w = C'(x)$ and all of the scPCPPs proofs are the canonical proofs for $C'(x)$ (i.e., $\bar{p}^{\mathsf{lines}}$ and $\bar{p}^{\mathsf{planes}}$), then all of the executions of the scPCPP verifiers accept (since the scPCPP verifiers are with one-sided error). Recalling that, by definition, $C(x)_{\bar{\imath}} = C_0^{\otimes d}(x)_{\bar{\imath}} = x_{\bar{\imath}}$, the decoding procedure $D^w(\bar{\imath})$ returns $x_{\bar{\imath}}$ with probability 1, as required.

Next, we prove the *(modified) relaxed soundness* of the decoder. Let $w \in \{0,1\}^n$ be a corrupted codeword that is δ_{radius}-close to a codeword $C(x)$, where δ_{radius} is a sufficiently small constant, to be determined later. We partition the analysis into three cases (Claims 4.3 and 4.4 and Theorem 4.5) that we analyze in the rest of this section. We begin with the following two simple claims.

The first claim shows that probability $\Omega(1)$, the random copy $\mathbf{c}$ in $(c_1, \ldots, c_{t_1})$ that is chosen in Step 1 cannot be "too far" from the codeword $C(x)$.

▶ **Claim 4.3.** *With probability at least* $1/4$*, the random copy* $\mathbf{c}$ *is* $4\delta_{C'}(w)$*-close to* $C(x)$*, where* $\mathbf{c}$ *is chosen uniformly at random from* $\bar{c}$*. That is,*

$$\Pr_{\mathbf{c} \in_R (c_1, \ldots, c_{t_1})} [\delta_C(\mathbf{c}) \leq 4\delta_{C'}(w)] \geq \frac{1}{4}.$$

Proof. Since $|\bar{c}| = \left|\bar{p}^{\mathsf{lines}}\right| = \left|\bar{p}^{\mathsf{planes}}\right|$, then $\bar{c} = (c_1, \ldots, c_{t_1})$ is $3\delta_{C'}(w)$-close to $C(x)^{t_1}$. This means that the expected relative distance of a random $\mathbf{c} \in \{c_1, \ldots, c_{t_1}\}$ from $C(x)$ is at most $3\delta_{C'}(w)$. Hence, by Markov's inequality, $\mathbf{c}$ is $4\delta_{C'}(w)$-far from $C(x)$ with probability at most $3/4$. ◀

Therefore, throughout the rest of the proof we fix a random copy $\mathbf{c}$ and assume that it is $4\delta_{C'}(w)$-close to $C(x)$. This only costs us at most a constant factor in the success probability of the decoder. Having fixed $\mathbf{c}$, recall that for $\bar{\imath} \in [n]^d$, the notation $\mathbf{c}_{\bar{\imath}}$ refers to the value of $\mathbf{c}$ at point $\bar{\imath}$. The next claim shows that if the bit we are trying to decode is not "corrupted" (in the random copy $\mathbf{c}$), then the decoder D never outputs a mistake.

▶ **Claim 4.4.** *If $\mathbf{c}_{\bar{\imath}} = x_{\bar{\imath}}$, then $\Pr_D[D^w(\bar{\imath}) \in \{x_{\bar{\imath}}, \perp\}] = 1$.*

Proof. By the definition of the decoder (see Figure 1), regardless of the rest of the values in the input, D always outputs either $\mathbf{c}_{\bar{\imath}}$ or $\perp$. ◀

The main part of the analysis takes place in the next lemma, where we assume that $\mathbf{c}_{\bar{\imath}} \neq x_{\bar{\imath}}$ and $\mathbf{c}$ is close to $C(x)$, and prove that the decoder succeeds with constant probability, as required. Recall that $\delta_{C'}(w) < \delta_{\mathsf{radius}}$, where δ_{radius} is a sufficiently small constant, to be determined later.

▶ **Lemma 4.5.** *Suppose that $\mathbf{c}$ is $4\delta_{C'}(w)$-close to $C(x)$ and that $\mathbf{c}_{\bar{\imath}} \neq x_{\bar{\imath}}$. Then,*

$$\Pr_D[D^w(\bar{\imath}) \in \{x_{\bar{\imath}}, \perp\}] = \Omega(1).$$

Proof. We say that a point $\bar{u} \in [n]^d$ in the tensor code $\mathbf{c}$ is corrupted if $\mathbf{c}_{\bar{u}} \neq C(x)_{\bar{u}}$. Since we assume that $\mathbf{c}$ is corrupted in the point $\bar{\imath}$ (which we wish to decode), by the definition of the decoder, the probability that D makes a mistake is equal to the probability that D reaches Step 4 and outputs $\mathbf{c}_{\bar{\imath}}$.

Recall that P_j is the set of points that we consider in the j^{th} iteration of the decoder. The set P_1 is the singleton that contains $\bar{\imath}$; i.e., $P_1 = \{\bar{\imath}\}$ and for every $j \in \{2, \ldots, d+1\}$ we recursively define P_j as the set of all points that reside on the $(j-1)$-axis-parallel lines that pass through points in P_{j-1} (see Step 3d). Note that for every $j \in [d]$ the cardinality of P_j is equal to the number of points in a codeword of $C_0^{\otimes j-1}$; that is, $|P_j| = n^{j-1}$. Hence, the number of points in all lines that pass through points in P_j (i.e., n^j) equals the number of points in a codeword of $C_0^{\otimes j}$. We will show that in order to corrupt $\mathbf{c}_{\bar{\imath}}$ without being detected by the scPCPPs, one has to corrupt a constant fraction of a large portion of the lines that pass through points in P_d, which in turn implies that one has to corrupt a constant fraction of the tensor code C, in contradiction to our assumption that $\delta_{C'}(w) < \delta_{\mathsf{radius}}$, for a sufficiently small constant δ_{radius}.

Consider the first iteration of Step 3 (where $j = 1$). Denote by $s \triangleq i_1$ the index of the bit that we wish to decode in the line $\mathbf{c}|_{\ell_{1,\bar{\imath}}}$, and denote by $b \triangleq \mathbf{c}_{\bar{\imath}}$ the value of $\mathbf{c}$ at $\bar{\imath}$.

We verify that the line that passes through $\bar{\imath}$ in the 1-direction is a codeword of C_0 that is consistent with the value of $\mathbf{c}$ at $\bar{\imath}$. This is done by running the verifier $V_{s,b}$ on input $\mathbf{c}|_{\ell_{1,\bar{\imath}}}$ and proof $p^{1,\bar{\imath}}$. Recall that the relative distance of C_0 (i.e., $\delta(C_0)$) is a constant. Since $\bar{\imath}$ is corrupted (i.e., $b = \mathbf{c}_{\bar{\imath}} \neq C(x)_{\bar{\imath}}$), if the line $\mathbf{c}|_{\ell_{1,\bar{\imath}}}$ is $\delta(C_0)/2$-close to the line $C(x)|_{\ell_{1,\bar{\imath}}}$ (which is a codeword of C_0 that is *inconsistent* with $\mathbf{c}_{\bar{\imath}}$), then $\mathbf{c}|_{\ell_{1,\bar{\imath}}}$ is $\delta(C_0)/2$-far from any codeword $y \in C_0$ that is *consistent* with $\mathbf{c}_{\bar{\imath}}$ (i.e., such that $y_s \neq C(x)_{\bar{\imath}}$). In this case, the verifier $V_{s,b}$ rejects $\mathbf{c}|_{\ell_{1,\bar{\imath}}}$ with probability at least $\mathsf{poly}(\delta(C_0)/2) = \Omega(1)$ (regardless of the corresponding proof), as required. Hence, in the following we assume that the line $\mathbf{c}|_{\ell_{1,\bar{\imath}}}$ is $\delta(C_0)/2$-far from

$C(x)|_{\ell_{1,\bar{i}}}$, and therefore P_2 contains a constant fraction of at least $\beta_1 \triangleq \delta(C_0)/2$ corrupted points.

We proceed by induction. Consider the j^{th} iteration, where $2 \leq j \leq d$. We show that if the set of points that we consider in the j^{th} iteration (the set P_j) contains a constant fraction of corrupted points, then either the decoder rejects with constant probability in the j^{th} iteration, or P_{j+1} contains a constant fraction of corrupted points (we denote this probability by β_{j+1}).

▶ **Claim 4.6.** *Let $2 \leq j \leq d$ and let $0 < \beta_j \leq 1$ be a constant. If P_j contains a at least a β_j fraction of corrupted points, then either:*

1. *The decoder rejects with probability at least $\Omega(1)$ in the j^{th} iteration; or,*
2. *P_{j+1} contains at least $\beta_{j+1} \triangleq \frac{\beta_j \cdot \delta(C_0)}{4}$ fraction of corrupted points.*

Proof of Claim 4.6. Consider the j^{th} iteration of Step 3. The decoder selects uniformly at random a point $\bar{u} = (u_1, \ldots, u_d) \in P_j$. Denote by $s = u_j$ the index of the bit that we wish to decode on the line $\mathbf{c}|_{\ell_{j,\bar{u}}}$ (which passes through $\bar{u}$ in the j^{th}-direction), and denote by $b \triangleq \mathbf{c}_{\bar{u}}$ the value of $\mathbf{c}$ at $\bar{u}$. By the hypothesis, $\bar{u}$ is corrupted with probability at least β_j.

Next, the verifier $V_{s,b}$ is executed on input $\mathbf{c}|_{\ell_{j,\bar{u}}}$ and proof $p^{j,\bar{u}}$. Observe that if a fraction of *at most* $\beta_j/2$ of the j-axis-parallel lines that pass through points in P_j (i.e., $\left\{\mathbf{c}|_{\ell_{j,\bar{z}}}\right\}_{\bar{z} \in P_j}$) are $\delta(C_0)/2$-far (each) from their corresponding lines in $C(x)$, then the decoder outputs $\perp$ with probability at least $\beta_j/2 \cdot \mathsf{poly}\left(\delta(C_0)/2\right) = \Omega(1)$, as required. This is because in this case, with probability at least $\beta_j/2$, we hit a line that is $\delta(C_0)/2$-close to its corresponding line in $C(x)$ (but the value of this line in u_j differs from $C(x)_{\bar{u}}$). As in the first iteration, this implies that this line is $\delta(C_0)/2$-far from any codeword $y \in C_0$ such that $y_s \neq C(x)_{\bar{u}}$, and hence the verifier $V_{s,b}$ rejects $\mathbf{c}|_{\ell_{j,\bar{u}}}$ with probability at least $\mathsf{poly}\left(\delta(C_0)/2\right)$ (regardless of the corresponding proof).

Otherwise (i.e., if the above case does not hold), *at least* $\beta_j/2$ of the lines in $\left\{\mathbf{c}|_{\ell_{j,\bar{z}}}\right\}_{\bar{z} \in P_j}$ are $\delta(C_0)/2$-far (each) from their corresponding lines in $C(x)$. Therefore, P_{j+1} contains at least a $\frac{\beta_j \cdot \delta(C_0)}{4}$ fraction of corrupted points. ◀

Note that P_{d+1} is the set of all points in $[n]^d$. By solving the recurrence relation, we get that $\beta_{d+1} = \frac{\delta(C_0)^d}{2^{2d-1}}$.[16] Recall that according to the hypothesis of the lemma, $\mathbf{c}$ is $4\delta_{\mathsf{radius}}$-close to $C(x)$. Fix the decoding radius δ_{radius} to a sufficiently small constant such that $4\delta_{\mathsf{radius}} < \beta_{d+1}$. Thus, Claim 4.6 implies that in one of the iterations the decoder must reject with probability at least $\Omega(1)$, as required. ◀

Remarks

The codewords of C' are of the form $w = (\bar{c}, \bar{p}^{\mathsf{lines}}, \bar{p}^{\mathsf{planes}})$, where the three parts are of equal length. The fact that the length of each of the three parts is proportional to the others is critical. The length of $\bar{c}$ must be proportional to the length of w in order for our code to have constant relative distance (recall that there is no guarantee on the distance of the scPCPPs). Moreover, the length of each of the scPCPP parts, $\bar{c}$ and $\bar{p}^{\mathsf{lines}}$, should be proportional to the length of w in order to obtain the average smoothness requirement (see Section 4.3).

[16] Recall that the fraction of corrupted points in P_2 is at least $\delta(C_0)/2$, and that for $2 \leq j \leq d$ the fraction of corrupted points in P_{j+1} (which we denote by β_{j+1}) is at least $\frac{\beta_j \cdot \delta(C_0)}{4}$.

We remark that we chose our tensor code to be *systematic* only for the sake of convenience. Instead, we could have added the message itself (repeated to obtain the proper length) as a fourth part to the code C'.[17]

Next, we note that for the proof that our code C' is a relaxed-LDC we only use the point-line scPCPPs and ignore the plane scPCPPs (i.e., the third part of w). Furthermore, we do not use the fact that the point-line scPCPPs are neither *strong* nor *canonical*. That is, to get only a relaxed-LDC with nearly-linear length it is enough to augment a good systematic tensor code (i.e., a tensor product of a systematic linear code with constant rate and constant relative distance) with a "regular" PCPP. However, the plane scPCPPs and the *strongness* and *canonicity* of the PCPPs will be heavily used in the proof that C' is also a strong-LTC (see Section 5).

4.3 Obtaining Average Smoothness

In this subsection, we conclude the proof that $C' : \{0,1\}^{k'} \to \{0,1\}^{n'}$ is a relaxed-LDC. Recall that in Section 4.2 we showed a decoder D for C' (described in Figure 1) that satisfies the first two conditions of Definition 4.2, i.e., the *completeness* and *(modified) relaxed soundness* conditions. Next, we show that D can be modified such that it also satisfies the third and final condition of Definition 4.2, i.e., the *average smoothness* condition (which, roughly speaking, requires that the decoder makes nearly-uniform queries on average).

Denote by $\mathcal{D}^w(i,j,r)$ the j^{th} query of the decoder D on coordinate $i \in [k']$, coin tosses r, and input oracle w. Recall that D satisfies the *average smoothness* condition if for every $w \in \{0,1\}^{n'}$ and $v \in [n']$, it holds that

$$\Pr_{i,j,r}\left[\mathcal{D}^w(i,j,r) = v\right] < \frac{2}{n'}, \tag{2}$$

where the probability is taken uniformly over all possible choices of $i \in [k']$, $j \in [q]$ (where q is the number of queries that D makes), and coin tosses r.

Firstly, we can relax the condition in Equation (2) and replace it with the condition

$$\Pr_{i,j,r}\left[\mathcal{D}^w(i,j,r) = v\right] = O\left(\frac{1}{n'}\right). \tag{3}$$

To see this, note that if the decoder D (which makes $q = O(1)$ queries) satisfies Equation (3), then we can obtain a decoder D' that makes $q' = O(q)$ queries and satisfy Equation (2) simply by running D and adding $O(q)$ uniformly distributed "dummy" queries (whose answers the decoder ignores).

Secondly, note that by the construction of D (of Figure 1), each of the scPCPPs verifiers that are being emulated by D makes nearly-uniform queries (see Theorems 3.1 and 3.2) to the statement it refers to and to its corresponding proof. Observe that on a random index $\bar{u} \in [k]^d$ the decoder D invokes the verifier of the *point-line* scPCPP on uniformly selected lines in a uniformly selected copy of the tensor code. Since the length of the first and second part of each codeword of C' (i.e., the tensor code and the *point-line* scPCPPs) constitutes a constant fraction of the length of each codeword of C', the decoder D satisfies Equation (3). Finally, by the foregoing discussion, D can be modified to satisfy Equation (2).

[17] Actually, this approach (of adding the message itself to the output of the code) was taken in previous constructions of relaxed-LDC (see [1, 15]). By using a systematic tensor code, we circumvented this unnecessary complication.

5 Establishing the Strong-LTC Property

In this section we prove that the code C', which was defined in Section 3, is a strong locally testable code.

▶ **Theorem 5.1.** *The code $C' : \{0,1\}^{k'} \to \{0,1\}^{n'}$ as defined in Section 3 is a* strong-LTC. *Furthermore, it has a tester that makes nearly-uniform queries.*

In order to prove Theorem 5.1 we need to present a tester T that is given an oracle access to $w \in \{0,1\}^{n'}$, makes $O(1)$ queries to w, and satisfies the following: For all $w \in C$ it holds that $T^w = 1$, and for all $w \notin C$ it holds that $\Pr_T[T^w = 0] \geq \mathsf{poly}\left(\delta_{C'}(w)\right)$.

5.1 Outline of the Tester and its Analysis

Recall that each codeword of C' consists of three parts: (1) an alleged d-dimensional tensor code $C = C_0^{\otimes d} : \{0,1\}^{k^d} \to \{0,1\}^{n^d}$, (2) alleged scPCPPs for every 2-dimensional plane in C; each scPCPP ascertains that the given plane is consistent with C, and (3) alleged scPCPPs for every pair of point $\bar{\imath}$ in C and line ℓ in C that passes through $\bar{\imath}$; each scPCPP ascertains that a line ℓ is a codeword of C_0 that is consistent with the value at a point $\bar{\imath}$.

For the simplicity of the exposition, we omit the repetitions of the three parts of the code (i.e., the tensor code, the *point-line* scPCPPs, and the *plane* scPCPP*s*) and assume instead that the length of the each part is equal. We note that this assumption can be easily removed by using an additional consistency test. See the full details in Section 5.2.

The key idea is that by the *robustness* property of tensor codes, the corruption rate of a codeword is proportional to the corruption rate of a random plane in the codeword. Hence, in order to ensure that the tensor code part of C' is valid, our tester use the *plane* scPCPPs to ascertain that a random plane is close to being valid. We note that for the tester, we do not need the *point-line* scPCPPs (which we only need for the decoder); however, since we need to ensure that also the point-line scPCPPs part is not corrupted, our tester also verifies a random *point-line* scPCPPs.

Clearly, this tester always accepts valid codewords. To analyze what happens with non-codewords consider a string that is somewhat far from C'. In this case, one of the following three cases must hold:

1. The tensor code part is far from a legal codeword of $C^{\otimes d}$.
2. The tensor code part is close to a legal codeword of $C^{\otimes d}$ but the *plane* scPCPP proofs part is far from the corresponding canonical proofs.
3. The tensor code part is close to a legal codeword of $C^{\otimes d}$ but the *point-line* scPCPP proofs part is far from the corresponding canonical proofs.

To ensure that in the first case the tester succeeds (i.e., rejects with sufficiently high probability), it is enough to test that a random plane in $\mathbf{c}$ is close to a codeword of $C^{\otimes 2}$. To accomplish this, we choose uniformly at random a (2-dimensional, axis-parallel) plane and run the corresponding *plane* scPCPP verifier. This suffices, since Theorem 2.6 asserts that if a tensor $\mathbf{c}$ is far from a legal codeword of $C^{\otimes d}$, then a random (2-dimensional, axis-parallel) plane in $\mathbf{c}$ must also be far from a legal codeword of $C^{\otimes 2}$.

The second and third cases are similar, and so, we only sketch how to handle the second case. Assume that the tensor is close to a codeword but the *plane* scPCPPs are far from the corresponding canonical proofs. From this assumption we can deduce that there are many planes that are close to legal codewords of $C^{\otimes 2}$, but whose corresponding scPCPPs are far from the canonical proofs. Thus, choosing a random plane and running the corresponding

The strong-LTC **Procedure for** C'

Input: oracle access to a string $w = (\bar{c}, \bar{p}^{\mathsf{lines}}, \bar{p}^{\mathsf{planes}})$.

For $s \in [n]$ and $b \in \{0,1\}$ let $V^{\mathsf{line}}(s,b)$ be a scPCPP verifier that refers to an input of the form $z \in \{0,1\}^n$ and asserts that there exists $y \in C_0$ such that $z = y$ and $z_s = b$.

Let V^{plane} be a scPCPP verifier that refers to an input of the form $z \in \{0,1\}^{n^2}$ and asserts that there exists $y \in C_0^{\otimes 2}$ such that $z = y$.

Choose a random copy of each of the three replicated parts of w. That is, choose uniformly at random a copy $\mathbf{c}$ in $\bar{c}$, a copy $\bar{p}_{\mathsf{line}} = \{p^{j,\bar{\imath}}\}_{\{\bar{\imath}\in[n]^d,\, j\in[d]\}}$ in $\bar{p}^{\mathsf{lines}}$, and a copy $\bar{p}_{\mathsf{plane}} = \{p^{\mathsf{p}}\}_{\{\mathsf{p}\in\mathsf{Planes}\}}$ in $\bar{p}^{\mathsf{planes}}$.

Accept if none of the following tests reject:

1. **The repetition test:** We query two random copies from the tensor part of w and check if they agree on a random location. More accurately, we select uniformly at random $\mathsf{r}, \mathsf{r}' \in [t_1]$ and reject if and only if c_{r} and $c_{\mathsf{r}'}$ disagree on a *random* coordinate.
2. **The *plane* scPCPP consistency test:** Choose a uniformly at random a plane $\mathsf{p} \in \mathsf{Planes}$. Reject if the verifier V^{plane} rejects on the plane p (i.e., input $\mathbf{c}|_{\mathsf{p}}$) and the proof p^{p}.
3. **The *point-line* scPCPP consistency test:** Choose uniformly at random a coordinate $\bar{u} = (u_1, \ldots, u_d) \in [n]^d$ and a direction $j \in [d]$ in $\mathbf{c}$. Reject if the verifier $V^{\mathsf{line}}(u_j, \mathbf{c}_{\bar{u}})$ rejects on the line passing through $\bar{u}$ in direction j and the proof $p^{j,\bar{u}}$. In other words, we reject if $V^{\mathsf{line}}(u_j, \mathbf{c}_{\bar{u}})$ rejects on input $\mathbf{c}|_{\ell_{j,\bar{u}}}$ and proof $p^{j,\bar{u}}$.

Figure 2 Strong local tester for C'.

plane scPCPP verifier ensures that the tester rejects with a sufficiently high probability. This is due to the *strongness* and *canonicity* features of our scPCPPs.

To conclude, the tester consists of three parts: (1) a repetition test, wherein we verify the repetition structure of the tensor, (2) *plane* scPCPP consistency test, wherein we verify that a random plane in the tensor is a legal codeword; this test ensures that both the tensor code part consists of valid codewords and its *plane* scPCPPs are the corresponding canonical proofs, and (3) *point-line* scPCPP consistency test, which we perform only to verify that the *point-line* scPCPPs consists of the canonical proofs that corresponds to the tensor part of the code.

5.2 The Full Proof

We proceed with the full proof of Theorem 5.1, which formalizes the intuition given in the previous section. We show a strong-LTC procedure for C'. The tester T is formally described in Figure 2. Note that since both the *point-line* and *plane* scPCPP verifiers make nearly-uniform queries (and the three parts of each codeword are of equal length), then the tester T also makes nearly-uniform queries.

Consider an arbitrary input $w \in \{0,1\}^{n'}$ such that $\delta_{C'}(w) \geq 0$. We view w as a string composed of three parts as in Section 3, i.e., $w = (\bar{c}, \bar{p}^{\mathsf{lines}}, \bar{p}^{\mathsf{planes}})$. The *completeness* of the tester is immediate: Indeed, if the input is a codeword, i.e., $w = C'(x)$, then the first part of w consists of identical copies of a tensor code, and hence the codeword repetition test accepts with probability 1. Similarly, the second and third parts consists of the canonical *point-line* and *plane* scPCPP proofs for the aforementioned tensor code, respectively; hence the (one-sided error) scPCPP verifiers will accept with probability 1.

Next, we prove the *soundness* of the tester. We partition the analysis into three cases (Claim 5.2 and Lemmas 5.3 and 5.4), which we analyze in the rest of this section.

Let $\hat{c} \in \{0,1\}^{n^d}$ be a tensor that is closest on average to the tensors in $\bar{c}$, i.e., a string that minimizes $\Delta(\bar{c}, \hat{c}^{t_1}) = \sum_{i=1}^{t_1} \Delta(c_i, \hat{c})$. The first (and standard) claim shows that if $\bar{c}$ is far from consisting of t_1 identical tensors, then the repetition test (of Step 1) rejects with high probability. Let γ be a constant set to $\delta(C)/(24d)$ (for the purpose of Lemma 5.4).

▶ **Claim 5.2.** *If* $\delta(\bar{c}, \hat{c}^{t_1}) \geq \frac{\gamma}{5} \cdot \delta_{C'}(w)$*, then* $\Pr_T[T^w = 0] \geq \mathsf{poly}\big(\delta_{C'}(w)\big)$.

Proof. Suppose that $\delta(\bar{c}, \hat{c}^{t_1}) \geq \frac{\gamma}{5} \cdot \delta_{C'}(w)$. The codeword repetition test rejects with probability at least

$$\mathop{\mathbb{E}}_{\mathsf{r},\mathsf{r}' \in_R [t_1]} \left[\frac{\Delta(c_{\mathsf{r}}, c_{\mathsf{r}'})}{n^d} \right] \geq \mathop{\mathbb{E}}_{\mathsf{r} \in_R [t_1]} \left[\frac{\Delta(c_{\mathsf{r}}, \hat{c})}{n^d} \right] = \frac{\Delta(\bar{c}, \hat{c}^{t_1})}{t_1 n^d}.$$

Therefore, $\Pr_T[T^w = 0] \geq \frac{\gamma}{5} \cdot \delta_{C'}(w) \geq \mathsf{poly}\big(\delta_{C'}(w)\big)$. ◀

The following lemma shows that if $\bar{c}$ consists of t_1 nearly identical tensors that are far from a codeword of C, then due to the *robustness* feature of tensor codes, a random plane in a random copy in $\bar{c}$ will be far from valid, and hence, Step 2 of the tester rejects with high probability.

▶ **Lemma 5.3.** *Assume* $\delta(\bar{c}, \hat{c}^{t_1}) < \frac{\gamma}{5} \cdot \delta_{C'}(w)$*. If* $\bar{c}$ *is* $\gamma \cdot \delta_{C'}(w)$*-far from* C^{t_1}*, then* $\Pr_T[T^w = 0] \geq \mathsf{poly}\big(\delta_{C'}(w)\big)$.

Proof. Observe that a random copy $\mathbf{c}$ of a tensor code in $\bar{c}$ is $\Omega\big(\delta_{C'}(w)\big)$-far from C with high probability. This is because $\delta_{C^{t_1}}(\bar{c}) \leq \delta_{C^{t_1}}(\hat{c}^{t_1}) + \delta(\hat{c}^{t_1}, \bar{c})$, which implies $\delta_C(\hat{c}) > \frac{4\gamma}{5} \cdot \delta_{C'}(w)$. Since at least $2/3$ of the c_i's are $3 \cdot \frac{\gamma}{5} \cdot \delta_{C'}(w)$-close to $\hat{c}$, these c_i's are $\frac{\gamma}{5} \cdot \delta_{C'}(w)$-far from C.

Next, by the *robustness* feature of tensor codes, we deduce that if the randomly selected tensor code $\mathbf{c}$ is $\Omega\big(\delta_{C'}(w)\big)$-far from being valid, then a random plane of $\mathbf{c}$ is also $\Omega\big(\delta_{C'}(w)\big)$-far from being valid. Specifically, by Theoren 2.6, there exists a constant $c_{\mathsf{robust}} \in (0,1)$ such that for every tensor $w \in \{0,1\}^{n^d}$ we have

$$\mathop{\mathbb{E}}_{\mathsf{p} \in_R \mathsf{Planes}} \left[\delta\left(w|_{\mathsf{p}}, C^{\otimes 2} \right) \right] > c_{\mathsf{robust}} \cdot \delta_{C^{\otimes d}}(w).$$

Hence, by an averaging argument,

$$\Pr_{\mathsf{p} \in \mathsf{Planes}} \left[\delta_{C^{\otimes 2}}(c|_{\mathsf{p}}) > \frac{c_{\mathsf{robust}}}{2} \cdot \frac{\gamma}{5} \cdot \delta_{C'}(w) \right] > \frac{c_{\mathsf{robust}}}{2} \cdot \frac{\gamma}{5} \cdot \delta_{C'}(w). \tag{4}$$

Note that, by Equation (4), with probability $\Omega\big(\delta_{C'}(w)\big)$ we select a plane that is $\Omega\big(\delta_{C'}(w)\big)$-far from a codeword of $C_0^{\otimes 2}$. Given such plane, the scPCPP verifier V^{plane} rejects with probability $\Omega\big(\delta_{C'}(w)\big)$. Thus, the tester T rejects with probability $\mathsf{poly}(\delta_{C'}(w))$ over the internal randomness of T. ◀

In the next lemma, we complete the analysis by assuming that $\bar{c}$ is sufficiently close to a codeword of C^{t_1}, and showing that in this case most of the "corruption" takes place in the parts of the scPCPP proofs, and hence the scPCPP consistency tests will reject with high probability.

▶ **Lemma 5.4.** *If* $\bar{c}$ *is* $\gamma \cdot \delta_{C'}(w)$*-close to being a codeword of* C^{t_1}*, then* $\Pr_T[T^w = 0] \geq \mathsf{poly}\big(\delta_{C'}(w)\big)$.

Proof. Recall that $\gamma = \frac{\delta(C)}{24d} < \frac{\delta(C)}{2}$. Therefore, our assumption that $\bar{c}$ is $\gamma \cdot \delta_{C'}(w)$-close to being a codeword of C^{t_1} implies that there exists a *unique* codeword c' of C that minimizes the distance of $\bar{c}' \triangleq (c')^{t_1}$ from $\bar{c}$. Let w' be the codeword of C' that consists of repetitions of the tensor code c' and its canonical scPCPP proofs; that is, Let $w' = \left(\bar{c}', \left(\pi_{\mathsf{lines}}(c')\right)^{t_2}, \left(\pi_{\mathsf{planes}}(c')\right)^{t_3}\right)$ be a codeword of C'. Denote by x the inverse of w' (i.e., , $w' = C'(x)$).

It is convenient to introduce notations for the fraction of corruptions in each part of C'. Towards this end, denote the fraction of errors in the first part of the code (the copies of the tensor code) by $\delta_{\bar{c}} = \delta(\bar{c}, \bar{c}')$. Analogously, denote by $\delta_{\bar{p}^{\mathsf{lines}}}$ and $\delta_{\bar{p}^{\mathsf{planes}}}$ the fraction of errors in the second and third parts of w (*point-line* scPCPPs and *plane* scPCPPs), respectively. Denote by $\delta_{\bar{p}^{\mathsf{total}}} = \left(\delta_{\bar{p}^{\mathsf{lines}}} + \delta_{\bar{p}^{\mathsf{planes}}}\right)/2$ the total fraction of errors in the second and third part of w together.

Observe that assuming the hypothesis of Lemma 5.4 (i.e., $\bar{c}$ is sufficiently close to $\bar{c}'$), the scPCPPs part (i.e., $\bar{p}^{\mathsf{lines}}$ and $\bar{p}^{\mathsf{planes}}$) must be somewhat far from the corresponding set of canonical scPCPP proofs; that is, assuming $\delta_{\bar{c}} < \delta_{C'}(w)$, then $\delta_{\bar{p}^{\mathsf{total}}} \geq \delta_{C'}(w)$. Therefore, since $\delta_{\bar{c}} \leq \gamma \cdot \delta_{C'}(w) < \delta_{C'}(w)$, we may assume that either: (1) the *plane* scPCPPs are sufficiently corrupted, i.e., $\delta_{\bar{p}^{\mathsf{planes}}} > \delta_{C'}(w)$, or (2) the *point-line* scPCPPs are sufficiently corrupted, i.e., $\delta_{\bar{p}^{\mathsf{lines}}} > \delta_{C'}(w)$. We claim that in the first case the *plane* scPCPP consistency test will reject with high probability, and in the second case the *point-line* scPCPP consistency test will reject with high probability. We prove this in the following two claims, from which Lemma 5.4 follows.

▶ **Claim 5.5.** *Assuming $\bar{c}$ is $\gamma \cdot \delta_{C'}(w)$-close to being a codeword of C^{t_1}, if $\delta_{\bar{p}^{\mathsf{planes}}} > \delta_{C'}(w)$, then* $\Pr_T[T^w = 0] \geq \mathsf{poly}\left(\delta_{C'}(w)\right)$.

▶ **Claim 5.6.** *Assuming $\bar{c}$ is $\gamma \cdot \delta_{C'}(w)$-close to being a codeword of C^{t_1}, if $\delta_{\bar{p}^{\mathsf{lines}}} > \delta_{C'}(w)$, then* $\Pr_T[T^w = 0] \geq \mathsf{poly}\left(\delta_{C'}(w)\right)$.

Claims 5.5 and 5.6 follow immediately from the *canonicity* and *strong soundness* features of the scPCPPs (along with averaging arguments). Since the proofs of Claims 5.5 and 5.6 are similar, we conclude the proof of Lemma 5.4 by showing Claim 5.5 and defer the proof of Claim 5.6 to Section E.

Proof of Claim 5.5. Loosely speaking, the hypothesis of the claim guarantees that: (1) $\bar{c}$ is close to being a *unique* codeword $C(x)^{t_1}$, and hence (by averaging arguments), most restrictions of a random copy $\mathbf{c}$ in $\bar{c} = (c_1, \ldots, c_{t_1})$ to a plane cannot be significantly corrupted; (2) the *plane* scPCPPs are far, on average, from the canonical proofs that corresponds to $C(x)$, and thus many *plane* scPCPPs are far from the canonical proofs for the planes of $C(x)$ they correspond to. By the foregoing, we conclude that there are many planes in $\mathbf{c}$ that are close to planes of $C(x)$ but their alleged *plane* scPCPP proofs are far from their canonical proofs. Thus, by the *canonicity* and *strong soundness* features of the scPCPPs, the verifier will reject with high probability. Details follow.

By the claim's hypothesis, $\bar{c}$ is $\delta_{\bar{c}}$-close to $C(x)^{t_1}$, where $\delta_{\bar{c}} \leq \gamma \cdot \delta_{C'}(w)$. Hence, by an averaging argument, with probability at least $2/3$ the random copy $\mathbf{c}$ is $3\delta_{\bar{c}}$-close to $C(x)$. Assume from now on that this is indeed the case. We say that a point $\bar{\imath} \in [n]^d$ in $\mathbf{c}$ is corrupted if $\mathbf{c}_{\bar{\imath}} \neq C(x)_{\bar{\imath}}$, and so, there are at most $3\delta_{\bar{c}} n^d$ corrupted points in $\mathbf{c}$. Since there are $\binom{d}{2} n^{d-2}$ axis-parallel planes in $\mathbf{c}$, then on average, the number of corrupted points in a random axis-parallel plane in $\mathbf{c}$ is at most $\frac{3\delta_{\bar{c}} n^d}{\binom{d}{2} n^{d-2}} < 3\delta_{\bar{c}} n^2$. Thus, by an averaging argument, we obtain that at most $\frac{\delta_{\bar{p}}}{4}$ fraction of the axis-parallel planes in $\mathbf{c}$ contain at least $\frac{4}{\delta_{\bar{p}}} \cdot 3\delta_{\bar{c}} n^2$ corrupted points.

Secondly, we note that a random copy of the *plane* scPCPP proofs contains a fraction of $\Omega\big(\delta_{C'}(w)\big)$ corrupted points with probability $\Omega\big(\delta_{C'}(w)\big)$. That is, by an averaging argument, with probability at least $\delta_{\bar{p}} \triangleq \delta_{\bar{p}^{\mathsf{planes}}}/2$ the random copy $\bar{p}$ in $\bar{p}^{\mathsf{planes}}$ is $\delta_{\bar{p}}$-far from its corresponding set of canonical proofs, $\pi_{\mathsf{planes}}(x) = \{\pi_{\mathsf{plane}}(C(x)|_{\mathsf{p}})\}_{\mathsf{p}\in\mathsf{Planes}}$. Assume from now on that $\bar{p}$ is $\delta_{\bar{p}}$-far from $\pi_{\mathsf{planes}}(x)$. Then, by an averaging argument, we obtain that at least $\delta_{\bar{p}}/2$ fraction of the proofs in $\bar{p} = \{p^{\mathsf{p}}\}_{\mathsf{p}\in\mathsf{Planes}}$ are $\delta_{\bar{p}}/2$-far from their corresponding (canonical) proofs $\pi_{\mathsf{planes}}(x)$.

By combining the conclusions of the last two paragraphs, we deduce that $\Omega\big(\delta_{C'}(w)\big)$-fraction of the planes p in $\mathbf{c}$ are both $\delta(C_0^{\otimes 2})/2$-close to the restriction of the tensor codeword $C(x)$ to p, and their corresponding proofs are $\Omega\big(\delta_{C'}(w)\big)$-corrupted; that is, a fraction of at least $\frac{\delta_{\bar{p}}}{4}$ of the axis-parallel planes p in $\mathbf{c}$ are $\delta(C_0^{\otimes 2})/2$-close to $C(x)|_{\mathsf{p}}$ (recall that $\frac{4}{\delta_{\bar{p}}} \cdot 3\delta_{\bar{c}} < 12\gamma < \delta(C)/2 \leq \delta(C_0^{\otimes 2})$), and in addition, their corresponding (alleged) *plane* scPCPP proofs in $\{p^{\mathsf{p}}\}_{\mathsf{p}\in\mathsf{Planes}}$ are $\delta_{\bar{p}}/2$-far from their (correct) canonical proofs in $\pi_{\mathsf{planes}}(x)$. Denote the set of planes that satisfy the foregoing condition by BAD.

Observe that for every plane $\mathsf{p} \in \text{BAD}$, in order for input $\mathbf{c}|_{\mathsf{p}}$ and proof p^{p} to be a valid claim (for the input-proof language that V^{plane} verifies), one must make at least one of the following changes: (1) change a fraction of at least $\frac{\delta_{\bar{p}}}{2}$ of the proof p^{p} such that it matches $\pi_{\mathsf{plane}}\left(C(x)|_{\mathsf{p}}\right)$, or (2) change a fraction of at least $\delta(C_0^{\otimes 2})/2$ of $\mathbf{c}|_{\mathsf{p}}$ (since p^{p} might be a valid proof for input $C_0^{\otimes 2}(y) \neq \mathbf{c}|_{\mathsf{p}}$). Thus, for every $\mathsf{p} \in \text{BAD}$, the probability that V^{plane} rejects input $c|_{\mathsf{p}}$ and proof p^{p} is at least polynomial in $\delta_{C'}(w)$.

Putting it all together, with probability $2/3$ we hit a random copy $\mathbf{c}$ of the tensor code that is $3\delta_{\bar{c}}$-close to $C(x)$. Furthermore, with probability at least $\delta_{\bar{p}}$ we hit a random copy $\bar{p}$ that is $\delta_{\bar{p}}$-corrupted, and subsequently, with probability $\delta_{\bar{p}}/2$ we hit a *plane* scPCPP proof that is $\delta_{\bar{p}}/2$-corrupted. Finally, assuming the foregoing, the scPCPP verifier V^{plane} rejects with probability $\mathsf{poly}\left(\delta_{C'}(w)\right)$. Therefore,

$$\Pr_T[T^w = 0] \geq \frac{2}{3} \cdot \delta_{\bar{p}} \cdot \frac{\delta_{\bar{p}}}{2} \cdot \mathsf{poly}\left(\delta_{C'}(w)\right) \geq \mathsf{poly}\big(\delta_{C'}(w)\big).$$

◀

This concludes the proof of Lemma 5.4. ◀

6 Strong Canonical PCPs of Proximity

In this section we construct scPCPPs with *polynomial* proof length for any good linear code (see Theorem 3.1) and for any *half-space* of a any good linear code (see Theorem 3.2). Our starting point (see Corollary 6.2) is the following result of [15],[18] which in turn builds upon [13, Section 5.2]: For any good code $C : \{0,1\}^k \to \{0,1\}^{ck}$, there exists a strong-LTC $C' : \{0,1\}^k \to \{0,1\}^{\mathsf{poly}(k)}$ such that the first half of $C'(x)$ consists of c blocks, each depending only on a k-bit long block of $C(x)$. Using this result, we construct a scPCPP for any good code C, where this construction applies the above result to several auxiliary codes that are derived from C.

6.1 scPCPPs for Good Codes

We start by recalling the statement of Theorem 3.1.

[18] Actually, Corollary 6.2 is a straightforward generalization of [15, Corollary B.3].

▶ **Theorem 3.1** (restated). *Let $C : \{0,1\}^k \to \{0,1\}^n$ be a linear code with constant relative distance and linear length. Then, there exists a* scPCPP *for codewords of C (i.e., for the set $\{C(x)\}_{x\in\{0,1\}^k}$). Furthermore, the proof length of the* scPCPP *is* poly(n)*, the* scPCPP *verifier makes nearly-uniform queries, and the canonical* scPCPP *proofs are linear (over* GF(2)*).*

The main technical tool upon which we rely (when proving Theorem 3.1) is the *linear inner proof systems* (hereafter, LIPS), constructed by Goldreich and Sudan. Loosely speaking, the LIPS mechanism allows to transform linear strong locally testable codes over a large alphabet into strong locally testable codes over a smaller alphabet (see [13, Section 5.2]). We encapsulate our usage of the LIPS mechanism in the following theorem, which generalizes [13, Theorem 5.20] and [13, Proposition 5.21]. Throughout this section, denote $\mathbb{F} = \mathsf{GF}(2)$.

▶ **Theorem 6.1.** *Let $\Sigma = \mathbb{F}^b$. For infinitely many k, there exists $n = \mathsf{poly}(k)$ and a linear code $E : \Sigma \to \mathbb{F}^n$ with constant relative distance such that the following holds. Suppose that $C : \Sigma^K \to \Sigma^N$ is a* strong-LTC *that is linear over $\mathbb{F}$ and has a (non-adaptive) tester that uses r random bits and makes nearly-uniform queries. Then, there exists $\ell = \mathsf{poly}(k)$ such that ℓ is a multiple of n, and a linear* strong-LTC *$C'' : \mathbb{F}^{bk} \to \mathbb{F}^{2^{r+1}\cdot\ell}$ such that the $2^r \cdot \ell$-bit long prefix of $C''(x)$ equals $\big(E(C(x)_1), \ldots, E(C(x)_N)\big)^{2^r\ell/(Nn)}$. Moreover, the tester of C'' makes nearly-uniform queries.*

As a corollary of Theorem 6.1, we obtain that any good linear code can be augmented to a linear strong-LTC with *polynomial* length, such that the prefix of the new code is closely related to that of the original code (but is *not* equal to the original code). This is done by viewing the good linear code as a trivial strong-LTC over a sufficiently large alphabet.

▶ **Corollary 6.2** (our starting point). *Let $C : \{0,1\}^k \to \{0,1\}^{ck}$ be a good linear code with constant relative distance, where $c \in \mathbb{N}$ is a constant. Then, for some $M, m = \mathsf{poly}(k)$, there exists a linear* strong-LTC *$C' : \{0,1\}^k \to \{0,1\}^{2M}$ and a linear code $E : \{0,1\}^k \to \{0,1\}^m$, which has constant relative distance, such that the M-bit long prefix of $C'(x)$ equals $\big(E(C(x)[1]), \ldots, E(C(x)[c])\big)^{M/(c\cdot m)}$, where $C(x)[i]$ is the i^{th} block of length k in $C(x)$. Furthermore, the (strong) tester of C' makes nearly-uniform queries.*

We remark that Theorem 6.1 and Corollary 6.2 are straightforward generalization of [15, Theorem B.2] and [15, Corollary B.3] (respectively), and we defer their proofs to Appendix A.

The Plan

Let $C : \{0,1\}^k \to \{0,1\}^{ck}$ be a good linear code, where $c \in \mathbb{N}$ is a constant. We construct a strong-LTC C' such that a constant fraction of each codeword $C'(x)$ contains copies of $C(x)$. This, in turn, implies a scPCPP for C (see Proposition 6.5). Note that by applying Corollary 6.2 to C we obtain a strong-LTC C' such that a constant fraction of each codeword $C'(x)$ contains copies of $\big(E(C(x)[1]), \ldots, E(C(x)[c])\big)$, but not of $C(x)$. This does not seem to suffice for obtaining a scPCPP, and so we use a different approach.

We start by using Corollary 6.2 to obtain a family of linear strong-LTCs, denoted by $\big\{C_i : \{0,1\}^k \to \{0,1\}^n\big\}_{i\in[ck]}$, where $n = \mathsf{poly}(k)$, with constant relative distance such that the prefix of each codeword $C_i(x)$ contains a linear number of copies of the i^{th}-bit of $C(x)$ (as well as other structural features that will be useful for us). This is done via the next lemma, which uses techniques from [15].

▶ **Lemma 6.3** (obtaining auxiliary codes C_i). *Let $C : \{0,1\}^k \to \{0,1\}^{ck}$ be a good linear code, where $c \in \mathbb{N}$ is a constant. There exist a constant $\alpha \in (0,1)$, a polynomial value $n = \mathsf{poly}(k)$,*

and a linear code $\hat{C} : \{0,1\}^k \to \{0,1\}^{cn}$ *with constant relative distance, which satisfy the following: For every* $i \in [ck]$*, there exists a function* $\pi_i : \{0,1\}^k \to \{0,1\}^{(c+1)n}$ *such that the code* $C_i : \{0,1\}^k \to \{0,1\}^{\alpha n+cn+(c+1)n}$*, given by*

$$C_i(x) = \big((C(x)_i)^{\alpha n}, \hat{C}(x), \pi_i(x)\big),$$

is a linear strong-LTC *with constant relative distance. Moreover, for every* $i \in [ck]$ *the (strong) tester of* C_i *makes nearly-uniform queries.*

We stress that the code $\hat{C}$ (which is common to all C_i's) is independent of i and constitutes a constant fraction of the length of each C_i.

Proof of Lemma 6.3. For every $j \in [c]$, we denote by $C(x)[j]$ the j^{th} block of length k of $C(x)$. For every $i \in [ck]$, consider the code $C'_i : \{0,1\}^k \to \{0,1\}^{(c+1)k}$ given by

$$C'_i(x) \triangleq \big((C(x)_i)^k, C(x)\big) = \big((C(x)_i)^k, C(x)[1], \ldots, C(x)[c]\big).$$

Note that C'_i is a good linear code.

For every $i \in [ck]$, we apply Corollary 6.2 to C'_i and obtain a linear strong-LTC $C''_i : \{0,1\}^k \to \{0,1\}^{2(c+1)\cdot n}$ with constant relative distance, which is (up to a permutation of its bit locations) of the form

$$C''_i(x) = \left(\left(E\big((C(x)_i)^k\big)\right)^t, \left(E\big(C(x)[1]\big)\right)^t, \ldots, \left(E\big(C(x)[c]\big)\right)^t, \pi_i(x)\right)$$

where $m, n = \mathsf{poly}(k)$, the function $E : \{0,1\}^k \to \{0,1\}^m$ is a linear code with constant relative distance, $t = n/m$, and $\pi_i(x) \in \{0,1\}^{(c+1)n}$ is some string. Moreover, the (strong) tester of C''_i makes nearly-uniform queries.

Denote by $\hat{C} : \{0,1\}^k \to \{0,1\}^{cn}$ the linear code (with constant relative distance) that is given by $\hat{C}(x) = \left(\left(E\big(C(x)[1]\big)\right)^t, \ldots, \left(E\big(C(x)[c]\big)\right)^t\right)$. Since E is a linear code with constant relative distance, then $E(0^k) = 0^m$ and $\Delta\big(E\left(1^k\right), 0^m\big) \geq \alpha m$ for some constant $\alpha \in (0,1)$. Now, for every $i \in [ck]$, consider the code $C_i : \{0,1\}^k \to \{0,1\}^{\alpha n+cn+(c+1)n}$, given by $C_i(x) = \big((C(x)_i)^{\alpha n}, \hat{C}(x), \pi_i(x)\big)$, which is obtained from C''_i by simply removing coordinates on which $E(0^k)$ and $E(1^k)$ agree, in each of the t copies in the first part (i.e., $E\big(C(x)_i\big)^k$).

Note that C_i has constant relative distance. Furthermore, since C''_i is linear and since we only removed coordinates on which the value is 0, the code C_i is also a linear code. Finally, by emulating the execution of the tester of C''_i on an (alleged) codeword of C_i (which can be done by returning 0 whenever a coordinate that was omitted is being queried), we obtain that $C_i(x)$, which is of the required form of the hypothesis, is a strong-LTC with a (strong) tester that makes nearly-uniform queries. ◀

In the actual proof of Theorem 3.1, we will construct a code C' that encodes a message x by concatenating the encodings of x by all of the strong-LTCs in $\{C_i : \{0,1\}^k \to \{0,1\}^n\}_{i\in[ck]}$ (i.e., $C'(x) \triangleq \big(C_1(x), \ldots, C_{ck}(x)\big)$). Thus, we will obtain a strong-LTC that (up to a permutation of the bit locations) contains copies of the entire codeword $C(x)$ in its prefix. We remark that, in general, the concatenation of strong-LTCs is *not* a strong-LTC. However, the structure of the aforementioned family of codes (specifically, the fact that all codes in the family contains a *common* sub-code) implies that the concatenation of codes in $\{C_i : \{0,1\}^k \to \{0,1\}^n\}_{i\in[ck]}$ yields a strong-LTC. The next proposition shows a sufficient condition for obtaining strong-LTCs via concatenation of strong-LTCs.

▶ **Proposition 6.4** (concatenating multiple encodings of strong-LTCs with a common sub-code)**.** *Let $C_1, \ldots, C_t : \{0,1\}^k \to \{0,1\}^n$ be* strong-LTC*s with constant relative distance. Let $I \subseteq [n]$ such that $|I| = \Omega(n)$, and let $\widehat{C} : \{0,1\}^k \to \{0,1\}^{|I|}$ be a code with constant relative distance. If $\widehat{C}(x) = C_1(x)|_I = C_2(x)|_I = \ldots = C_t(x)|_I$ for every $x \in \{0,1\}^k$, where $C_i(x)|_I$ denotes the restriction of $C_i(x)$ to I, then the code $C'(x) \triangleq \big(C_1(x), \ldots, C_t(x)\big)$ is a* strong-LTC *with constant relative distance. Moreover, if the (strong) testers of $C_1, \ldots, C_t$ make nearly-uniform queries, then the (strong) tester of C' also makes nearly-uniform queries.*

Proposition 6.4 follows by using a tester that (1) emulates the strong-LTC tester of a randomly selected concatenated code C_i (to ascertain that each concatenated codeword is valid), and (2) tests the consistency of the common code $\hat{C}$ in two randomly selected concatenated codes (to assure that all of the concatenated codewords encode the same message). The analysis is quite straightforward and is deferred to Appendix B.

The last tool we shall need in order to prove Theorem 3.1 is the following proposition, which allows us to transform strong-LTCs to scPCPPs for prefixes of the strong-LTCs' codewords.

▶ **Proposition 6.5** (from strong-LTCs to scPCPPs for related codewords)**.** *Let $C : \{0,1\}^k \to \{0,1\}^n$ be a linear code, and let $C' : \{0,1\}^k \to \{0,1\}^{n'}$ be a linear* strong-LTC*. If there exists $I \subseteq [n']$ where $|I| = \Omega(n')$ and $n' - |I| = \Omega(n')$ such that $C'(x)|_I = \big(C(x)\big)^{|I|/n}$, then there exists a* scPCPP *for C (i.e., for the set of codewords $\{C(x)\}_{x\in\{0,1\}^k}$) with proof length $O(n')$. Moreover, the canonical* scPCPP *proofs are linear, and if the (strong) tester of C' makes nearly-uniform queries, then the verifier of the* scPCPP *for C also makes nearly-uniform queries.*

Proof. Let C, C', and I be as in the hypothesis. Assume, without loss of generality, that $I = \{1, \ldots, |I|\}$. Denote the (strong) tester of C' by T. We use T in a black-box manner in order to construct a scPCPP for the set $\{C(x)\}_{x\in\{0,1\}^k}$.

Given a codeword $C(x)$, the canonical scPCPP proof for $C(x)$ is given by $\pi(x) \triangleq C'(x)|_{[n']\setminus I}$, where $C'(x)|_{[n']\setminus I}$ is the restriction of $C(x)$ to the coordinates outside of I. Let V be the scPCPP verifier that gets oracle access to an alleged codeword $w \in \{0,1\}^n$ and oracle access to a proof oracle p of length $n' - |I|$. Let $t = |I|/n$. The verifier V emulates the execution of T on (w^t, p) as follows: Each query that T makes to the first part (which are allegedly $C(x)^t$) is simulated by a corresponding query to the input oracle w,[19] and each query that T makes to the other coordinates (which is allegedly $\pi(x)$) is simulated by a corresponding query to the proof oracle. The verifier V accepts if and only if the emulated run of T on (w^t, p) accepted. Note that if T makes nearly-uniform queries, then V also makes nearly-uniform queries.

The *completeness* of V is immediate: If w is a codeword $C(x)$ and $p = \pi(x)$, then $\big(w^t, p\big)$ is a codeword of C'. We conclude the proof by showing the *soundness* of V. Note that V gets as input a pair of an alleged codeword w and an alleged canonical proof p. Suppose that $\delta_{\mathsf{PCPP}}(w,p) \triangleq \min_{x\in\{0,1\}^n}\big\{\max\big(\delta(x,w)\,;\,\delta(\pi_{\mathsf{canonical}}(x),p)\big)\big\} > 0$.

For every $x \in \{0,1\}^n$, either the alleged proof p is $\delta_{\mathsf{PCPP}}(w,p)$-far from $\pi_{\mathsf{canonical}}(x)$, or the alleged codeword is $\delta_{\mathsf{PCPP}}(w,p)$-far from $C(x)$. In the former case, since $|p| = n' - |I| = \Omega(n')$, it holds that $\delta\big((w^t,p),(x^t,\pi_{\mathsf{canonical}}(x))\big) = \Omega\big(\delta_{\mathsf{PCPP}}(w,p)\big)$. In the latter case, since $\delta_{C^t}(w^t) = \delta_C(w)$ and $|w^t| = \Omega(n')$, it holds that $\delta\big((w^t,p),(x^t,\pi_{\mathsf{canonical}}(x))\big) =$

[19] Note that the tester expects t copies of $C(x)$, while the input oracle consists of a single copy. Hence, the emulation is done simply by directing the query of the i^{th} bit of the j^{th} copy to the i^{th} bit of the input oracle, for every i, j.

$\Omega\big(\delta_{\mathsf{PCPP}}(w,p)\big)$. Therefore $\delta_{C'}\big((w^t,p)\big) = \Omega\big(\delta_{\mathsf{PCPP}}(w,p)\big)$, and thus the tester of C', and subsequently the verifier V, will reject with probability $\mathsf{poly}(\delta_{\mathsf{PCPP}}(w,p))$ as required. ◀

Using Lemmas 6.3 and Propositions 6.4 and 6.5, we proceed with the proof of Theorem 3.1.

Proof of Theorem 3.1. Let $c \in \mathbb{N}$ be a constant and $C : \{0,1\}^k \to \{0,1\}^{ck}$ be a linear code with constant relative distance. We show a scPCPP, with polynomial proof length, for the language of all codewords of C.

First, we apply Lemma 6.3 on C and get that there exists a linear code $\hat{C} : \{0,1\}^k \to \{0,1\}^{cn}$ with constant relative distance and a set of codes

$$\left\{C_i : \{0,1\}^k \to \{0,1\}^{\alpha n+cn+(c+1)n}\right\}_{\{i\in[ck]\}}$$

such that each C_i is a linear code with constant relative distance that is given by

$$C_i(x) = \big((C(x)_i)^{\alpha n}, \hat{C}(x), \pi_i(x)\big),$$

where $\alpha \in (0,1)$, $n = \mathsf{poly}(k)$ and $\pi_i : \{0,1\}^k \to \{0,1\}^{(c+1)n}$. Moreover, each C_i makes nearly-uniform queries.

Next, we consider the code $C'(x) \triangleq \big(C_1(x), \ldots, C_{ck}(x)\big)$. Observe that, up to a permutation of the indices, C' has the form

$$C'(x) = \big(C(x)^{\alpha n}, \hat{C}(x)^{ck}, \pi(x)\big),$$

where $\pi(x) = \pi_1(x), \ldots, \pi_{ck}(x)$. Note that $\left|\hat{C}(x)^{ck}\right| = ck \cdot cn$, which is a constant fraction of $|C'(x)|$. By Proposition 6.4, the code C' is a strong-LTC with constant relative distance that makes nearly-uniform queries.

Finally, the theorem follows by applying Proposition 6.5 to the code C' with $I = [\alpha n \cdot ck]$, where the code C is repeated $\alpha n = |I|/(ck)$ times. (Indeed, we use the fact that $|I|$ is a constant fraction of $|C'(x)|$.) Note that the scPCPP proof we obtain (namely, $\big(\hat{C}(x)^{ck}, \pi(x)\big)$) is of length $\mathsf{poly}(k)$. ◀

6.2 scPCPPs for Half-Spaces of Good Codes

We start by recalling the statement of Theorem 3.2.

▶ **Theorem 3.2** (restated). *Let $C : \{0,1\}^k \to \{0,1\}^n$ be a linear code with constant relative distance and linear length. Let $i \in [k]$ be a location in a message and $b \in \{0,1\}$ a bit. Then, there exists a* scPCPP *for $C_{i,b}$, where $C_{i,b}$ is the set of all codewords w of C such that the i^{th}-bit of w equals b (i.e., $w_i = b$). Furthermore, the proof length of the* scPCPP *is* $\mathsf{poly}(n)$*, the* scPCPP *verifier makes nearly-uniform queries, and the* scPCPP *proofs are linear (over* $\mathsf{GF}(2)$*).*

Theorem 3.2 is obtained by using Theorem 3.1 in a black-box manner. Specifically, note that in case $b = 0$, the code $C_{i,0}$ is linear, and thus we can apply Theorem 3.1 directly. On the other hand, in case $b = 1$, the code $C_{i,1}$ is *not* linear, but we can "shift" it (by a fixed codeword of $C_{i,1}$) and apply Theorem 3.1.

Proof of Theorem 3.2. In light of the above, we focus on the case in which $b = 1$. Assume, without loss of generality, that there exists a codeword $c^{(i)}$ of C such that that the i^{th}-bit of $c^{(i)}$ is 1 (otherwise, we can always reject). Consider a verifier, $V_{i,1}$, that gets oracle access to an input string w and a proof π, and proceeds as follows. The verifier $V_{i,1}$ emulates the

execution of $V_{i,0}$ (obtained via Theorem 3.1) on input oracle $w + c^{(i)}$ (where the summation is point-wise over $\mathsf{GF}(2)$) and its proof oracle π (which should be the canonical proof for $w + c^{(i)} \in C_{i,0}$). Note that the verifier $V_{i,0}$ makes nearly-uniform queries, and so $V_{i,1}$ also makes nearly-uniform queries. We show that $V_{i,1}$ is a scPCPP for $C_{i,1}$

The *completeness* is immediate: Recall that if w is a codeword of $C_{i,1}$, then $w = C(x)$ such that $w_i = 1$. By the linearity of C, $w + c^{(i)}$ is a codeword of C such that its i^{th} bit is 0 (i.e., $\left(w + c^{(i)}\right)_i = 0$). Therefore, we actually invoke $V_{i,0}$ on a codeword of $C_{i,0}$. For the *soundness* condition, assume that $\delta_{C_{i,1}}(w) > 0$. Observe that

$$\delta_{C_{i,0}}\left(w + c^{(i)}\right) = \min_{w' \in C_{i,0}} \delta\left(w', w + c^{(i)}\right) = \min_{w' \in C_{i,0}} \delta\left(w' + c^{(i)}, w\right) = \delta_{C_{i,1}}(w).$$

Therefore, the verifier $V_{i,1}$ will reject the input $w + c^{(i)}$ (given the corresponding canonical proof) with probability at least $\mathsf{poly}\left(\delta_{C_{i,1}}(w)\right)$, as required. ◀

7 Application to Property Testing

In this section we give an application of our main result (Theorem 1.1) to the area of *property testing*. Specifically, we improve on the best known separation result, due to Gur and Rothblum [15], between the complexity of *decision* versus *verification* in the property testing model. Details follow.

The study of property testing, initiated by Rubinfeld and Sudan [20] and Goldreich, Goldwasser and Ron [11], considers highly-efficient randomized algorithms that solve approximate decision problems, while only inspecting a small fraction of the input. Such algorithms, commonly referred to as *testers*, are given oracle access to some object, and are required to determine whether the object has some predetermined property or is far (say, in Hamming distance) from every object that has the property.

Remarkably, it turns out that many natural properties can be tested by making relatively few queries to the object. However, there are also many natural properties that *no* tester can test efficiently. In fact, "almost all" properties require a very large query complexity to be tested. Motivated by this limitation, Gur and Rothblum [15] initiated the study of $\mathcal{MAP}$ proofs of proximity (hereafter $\mathcal{MAP}$s), which can be viewed as the NP proof-system analogue of property testing.

Loosely speaking, an $\mathcal{MAP}$ is a probabilistic proof system that augments the property testing framework by allowing the tester full and free access to an (alleged) proof. That is, such a proof-aided tester for a property Π is given *oracle* access to an input x and *free* access to a proof string w, and should distinguish between the case that $x \in \Pi$ and the case that x is far from Π, while only making a sublinear number of queries. More precisely, given a *proximity parameter* $\varepsilon > 0$, we require that for inputs $x \in \Pi$, there exist a proof that the tester accepts with high probability, and for inputs x that are ε-far from Π no proof will make the tester accept, except with some small probability of error. For formal definitions we refer to [15, Section 2].

As observed by [15], given an $\mathcal{MAP}$ proof of length that is linear in the size of the object (specifically, a proof that fully describes the object), every property can be tested by only making $O(1/\varepsilon)$ queries to the object, simply by verifying the proof's consistency with the object. Hence, it is natural to measure the complexity of an $\mathcal{MAP}$ by both the length of the proof and the number of queries made in order to decide whether $x \in \Pi$ or ε-far from it. We note that a property tester can be viewed as an $\mathcal{MAP}$ that uses a proof of length 0.

Gur and Rothblum [15] showed that the task of separating the power of property testers and $\mathcal{MAP}$s can be reduced to the task of designing a code that is both locally testable and

locally decodable. Furthermore, they noticed that for such a separation, *relaxed decodability* suffices. Unable to construct a code as in Theorem 1.1, Gur and Rothblum used several weaker codes to obtain partial separation results. Specifically, they proved the following theorem.

▶ **Theorem 7.1** (Theorems 3.1, 3.2 and 3.3 in [15]). *In all items, n denotes the length of the main input being tested.*

1. *For every constant $\alpha > 0$, there exists a property Π_α that has an $\mathcal{MAP}$ that uses a proof of length $O(\log n)$ and makes $\mathsf{poly}(1/\varepsilon)$ queries for every $\varepsilon > 1/\mathsf{polylog}(n)$, but for which every* property tester *must make $\Omega(n^{1-\alpha})$ queries.*
2. *For every constant $\alpha > 0$, there exists a property Π_α that has an $\mathcal{MAP}$ that uses a proof of length $O(\log n)$ and makes $\mathsf{poly}(\log n, 1/\varepsilon)$ queries, but for which every* property tester *must make $\Omega(n^{1-\alpha})$ queries.*
3. *There exists a universal constant $c \in (0,1)$ and a property Π that has an $\mathcal{MAP}$ that uses a proof of length $O(\log n)$ and makes $\mathsf{poly}(1/\varepsilon)$ queries (without limitation on ε), but for which every* property tester *must make n^c queries.*

Furthermore, each of the above $\mathcal{MAP}$s has one-sided error.

Note that each of these separation results has a drawback: The first separation works only for sufficiently large values of the proximity parameter, the second separation has non-constant query complexity for the $\mathcal{MAP}$s, and the third separation does not require property testers to make nearly-linear number of queries.

Plugging in the code C' from Theorem 1.1 into the framework developed by [15, Lemmas 3.4 and 3.5], we achieve the best of all the aforementioned results; that is, a separation for all values of the proximity parameter, with constant query complexity for the $\mathcal{MAP}$s, and nearly-linear query complexity for testers. Formally, we obtain the following separation result between $\mathcal{MAP}$s and property testers.

▶ **Theorem 1.3** (restated). *For every constant $\alpha > 0$, there a property Π_α that has an $\mathcal{MAP}$ that uses a proof of length $O(\log n)$ and makes $\mathsf{poly}(1/\varepsilon)$ queries (without limitation on ε), but for which every* property tester *must make $n^{1-\alpha}$ queries. Furthermore, the $\mathcal{MAP}$ has one-sided error.*

Acknowledgments. We would like to thank Or Meir for a helpful discussion regarding the robustness of tensor codes and its relation to local testability, and Michael Ben-Or for raising the issue of tolerant testing. The third author would like to thank his advisor Moni Naor for his support and encouragement.

References

1 Eli Ben-Sasson, Oded Goldreich, Prahladh Harsha, Madhu Sudan, and Salil P. Vadhan. Robust PCPs of proximity, shorter PCPs, and applications to coding. *SIAM Journal on Computing*, 36(4):889–974, 2006.

2 Eli Ben-Sasson and Madhu Sudan. Robust locally testable codes and products of codes. *Random Structures & Algorithms*, 28(4):387–402, 2006.

3 Eli Ben-Sasson and Michael Viderman. Towards lower bounds on locally testable codes via density arguments. *Computational Complexity*, 21(2):267–309, 2012.

4 Benny Chor, Oded Goldreich, Eyal Kushilevitz, and Madhu Sudan. Private information retrieval. *Journal of the ACM*, 45(6):965–981, 1998.

5 Irit Dinur and Tali Kaufman. Dense locally testable codes cannot have constant rate and distance. In *APPROX-RANDOM*, pages 507–518, 2011.

6 Irit Dinur and Omer Reingold. Assignment testers: Towards a combinatorial proof of the PCP theorem. *SIAM Journal on Computing*, 36(4):975–1024, 2006.

7 Klim Efremenko. 3-query locally decodable codes of subexponential length. *SIAM Journal on Computing*, 41(6):1694–1703, 2012.

8 Katalin Friedl and Madhu Sudan. Some improvements to total degree tests. In *ISTCS*, pages 190–198, 1995.

9 William I. Gasarch. A survey on private information retrieval (column: Computational complexity). *Bulletin of the EATCS*, 82:72–107, 2004.

10 Oded Goldreich. Short locally testable codes and proofs: A survey in two parts. In *Property Testing*, pages 65–104, 2010.

11 Oded Goldreich, Shafi Goldwasser, and Dana Ron. Property testing and its connection to learning and approximation. *Journal of the ACM*, 45(4):653–750, 1998.

12 Oded Goldreich and Dana Ron. On proximity oblivious testing. In *STOC*, pages 141–150, 2009.

13 Oded Goldreich and Madhu Sudan. Locally testable codes and PCPs of almost-linear length. *Journal of the ACM*, 53(4):558–655, 2006.

14 Alan Guo, Swastik Kopparty, and Madhu Sudan. New affine-invariant codes from lifting. In *ITCS*, pages 529–540. ACM, 2013.

15 Tom Gur and Ron D. Rothblum. Non-interactive proofs of proximity. In *ITCS*, pages 133–142. ACM, 2015.

16 Venkatesan Guruswami and Atri Rudra. Tolerant locally testable codes. In *Approximation, Randomization and Combinatorial Optimization. Algorithms and Techniques*, pages 306–317. Springer, 2005.

17 Jonathan Katz and Luca Trevisan. On the efficiency of local decoding procedures for error-correcting codes. In *STOC*, pages 80–86, 2000.

18 Tali Kaufman and Michael Viderman. Locally testable vs. locally decodable codes. In *APPROX-RANDOM*, pages 670–682, 2010.

19 Michal Parnas, Dana Ron, and Ronitt Rubinfeld. Tolerant property testing and distance approximation. *Journal of Computer and System Sciences*, 72(6):1012–1042, 2006.

20 Ronitt Rubinfeld and Madhu Sudan. Robust characterizations of polynomials with applications to program testing. *SIAM Journal on Computing*, 25(2):252–271, 1996.

21 Luca Trevisan. Some applications of coding theory in computational complexity. *Electronic Colloquium on Computational Complexity (ECCC)*, 2004.

22 Michael Viderman. A combination of testability and decodability by tensor products. In *APPROX-RANDOM*, pages 651–662, 2012.

23 Michael Viderman. Strong LTCs with inverse poly-log rate and constant soundness. *Electronic Colloquium on Computational Complexity (ECCC)*, 20:22, 2013.

24 Sergey Yekhanin. Towards 3-query locally decodable codes of subexponential length. *Journal of the ACM*, 55(1):1, 2008.

25 Sergey Yekhanin. Locally decodable codes. *Foundations and Trends in Theoretical Computer Science*, 6(3):139–255, 2012.

A Obtaining Strong LTCs from LIPS

In this appendix, we provide tools that allow us to use the *linear inner proof systems* (hereafter, LIPS), constructed by Goldreich and Sudan [13], to obtain families of strong-LTCs with several features that we take advantage of in Appendix 6. Specifically, we prove Theorem 6.1 and Corollary 6.2. Throughout this section, denote $\mathbb{F} = \mathsf{GF}(2)$. Recall the statement of Theorem 6.1.

▶ **Theorem 6.1** (restated). *Let $\Sigma = \mathbb{F}^b$. For infinitely many k, there exists $n = \mathsf{poly}(k)$ and a linear code $E : \Sigma \to \mathbb{F}^n$ such that the following holds. Suppose that $C : \Sigma^K \to \Sigma^N$ is a* strong-LTC *that is linear over $\mathbb{F}$ and has a (non-adaptive) tester that uses r random bits and makes nearly-uniform queries. Then, there exists $\ell = \mathsf{poly}(k)$ such that ℓ is a multiple of n, and a linear* strong-LTC *$C'' : \mathbb{F}^{bk} \to \mathbb{F}^{2^{r+1}\cdot\ell}$ such that the $2^r \cdot \ell$-bit long prefix of $C''(x)$ equals $\big(E(C(x)_1), \ldots, E(C(x)_N)\big)^{2^r \ell/Nn}$. Moreover, the tester of C'' makes nearly-uniform queries.*

Proof. We follow the proof of [13, Theorem 5.20], while using the code C of the theorem's hypothesis instead of the third ingredient in that proof. In addition, following [13, Proposition 5.21], we use composition theorems (i.e., [13, Theorem 5.15] and [13, Theorem 5.17]) that preserve the nearly-uniform distribution of the queries the verifiers (or tester) make, thus ascertaining that $C''(x)$ has a tester that queries each location with probability $\Theta(1/N)$. We note that in our settings, the overhead of replacing the "vanilla" composition theorems (which are used in [13, Theorem 5.20]) with the composition theorems that preserve the nearly-uniform queries is insignificant. Details follow.

In the following description, all references refer to [13]. Recall some basics regarding the terminology used in [13]. By Definitions 5.8 and 5.9, a $\big(\mathbb{F}, (q,b) \to (p,a), \delta, \gamma\big)$-LIPS refers to input oracles $X_1, ..., X_q : [n] \to \mathbb{F}^a$ and a proof oracle $X_{q+1} : [\ell] \to \mathbb{F}^a$, where the input oracles provide an n-long encoding (over $\mathbb{F}^a$) of a single symbol in the (much) bigger alphabet $\mathbb{F}^b$ (i.e., this encoding is denoted $E : \mathbb{F}^b \to (\mathbb{F}^a)^n$). (In addition δ is the relative distance of the encoding used, and γ is the detection ratio in strong soundness. In the following, both parameters will be small constants.)

The proof of Theorem 5.20 starts with an overview (page 79), and then lists three ingredients (page 80) that will be used: (1) The Hadamard based $\big(\mathbb{F}, (p_H, k_H) \to (p_H + 5, 1), 1/2, 1/8\big)$-LIPS (for any choice of p_H and k_H) of Proposition 5.18, (2) The Reed-Muller based $\big(\mathbb{F}, (p_{RM}, k_{RM}) \to (p_{RM} + 4, \mathsf{poly}(\log p_{RM} k_{RM})), 1/2, \Omega(1)\big)$-LIPS (for any choice of p_{RM} and k_{RM}) of Proposition 5.18, and (3) a specific strong-LTC (namely, the strong-LTC in Part 1 of Theorem 2.4). We shall use the very same first two ingredients,[20] but use the code C in place of the third. Assume, without loss of generality, that the randomness complexity r of the strong (tester) of C satisfies that 2^r is a multiple of N. (We remark that all three ingredients have verifiers or testers that make nearly-uniform queries, and that we compose these ingredients via the composition theorems that preserve this distribution of queries.) Specifically, the second paragraph following the ingredients-list asserts that for any desired p'' and k'', an $\big(\mathbb{F}, (p'', k'') \to (p'' + 13, 1), \Omega(1), \Omega(1/p'')^2\big)$-LIPS with randomness $O(p'' \log k'')$, input length $\mathsf{poly}(p''k'')$, and proof length that are $\mathsf{poly}(p''k'')$. We shall use $p'' = O(1)$ and $k'' = b$, where the $O(1)$ stands for the query complexity of the codeword tester for C. Thus the above simplifies to asserting an $\big(\mathbb{F}, (O(1), b) \to (O(1), 1), \Omega(1), \Omega(1)\big)$-LIPS

[20] We remark that while these two LIPSs are presented in [13] as if they are non-uniform, it can be verified that they can be presented in uniform terms (i.e., computable by Turing machines rather than by circuits).

with randomness $O(\log b)$ and input/proof lengths (i.e., n and ℓ) that are $\mathsf{poly}(b)$. Without loss of generality, we may assume that ℓ is a multiple of n.

Next, we wish to compose C with the above LIPS via Theorem 5.15 (instead of via Theorem 5.13, which does not preserve the nearly-uniform distribution of the queries). It follows that in Item 1 of Theorem 5.15 we use K, N and r as provided by the hypothesis and $q = O(1)$. For Item 2, we use b as provided by the hypothesis, ($q = O(1)$ as above), $p = O(1)$ and $a = 1$, and $n, \ell = \mathsf{poly}(b)$ (all fitting the LIPS above). So we have $\Gamma = F$, and get a strong-LTC mapping $\mathbb{F}^{bK}$ to $\mathbb{F}^{2^{r+1}\cdot\ell}$, which makes nearly-uniform queries. In particular, for $t = 2^r\ell/Nn$ (i.e., $tNn = 2^r\ell$), as shown on top of page 56 (see Equation (32)), the first half of the codewords of the resulting code have the form $\big(E(C(x)_1)), ..., E(C(x)_N)\big)^t$, where $x \in \mathbb{F}^{bK}$ is viewed as an element of Σ^K. The theorem follows. ◀

Next, recall the statement of Corollary 6.2.

▶ **Corollary 6.2** (restated). *Let $C : \{0,1\}^k \to \{0,1\}^{ck}$ be a good linear code with constant relative distance, where $c \in \mathbb{N}$ is a constant. Then, for some $M, m = \mathsf{poly}(k)$, there exists a linear* strong-LTC *$C' : \{0,1\}^k \to \{0,1\}^{2M}$ and a linear code $E : \{0,1\}^k \to \{0,1\}^m$, which has constant relative distance, such that the M-bit long prefix of $C'(x)$ equals $\big(E(C(x)[1]), ..., E(C(x)[c])\big)^{M/cm}$, where $C(x)[i]$ is the i^{th} block of length k in $C(x)$. Furthermore, the (strong) tester of C' makes nearly-uniform queries.*

Proof. Let $C : \mathbb{F}^k \to \mathbb{F}^{ck}$ be a good linear code. Viewing C as a mapping from $\Sigma = \mathbb{F}^k$ to Σ^c, note that C is a strong-LTC, which is (trivially) checked by reading all c symbols (and hence, by definition, it makes uniform queries). The claim follows by instantiating Theorem 6.1 using the code C and taking $b = k$, $K = 1$, $N = c = O(1)$, and $r = 0$. ◀

B Concatenating Multiple Encodings of Strong LTCs

In this appendix, we show a sufficient condition for obtaining strong-LTCs via concatenation of strong-LTCs. Recall the statement of Proposition 6.4.

▶ **Proposition 6.4** (restated). *Let $C_1, \ldots, C_t : \{0,1\}^k \to \{0,1\}^n$ be* strong-LTC*s with constant relative distance. Let $I \subseteq [n]$ such that $|I| = \Omega(n)$, and let $\widehat{C} : \{0,1\}^k \to \{0,1\}^{|I|}$ be a code with constant relative distance. If $\widehat{C}(x) = C_1(x)|_I = C_2(x)|_I = \ldots = C_t(x)|_I$ for every $x \in \{0,1\}^k$, where $C_i(x)|_I$ denotes the restriction of $C_i(x)$ to I, then the code $C'(x) \triangleq \big(C_1(x), \ldots, C_t(x)\big)$ is a* strong-LTC *with constant relative distance. Moreover, if the (strong) testers of $C_1, \ldots, C_t$ make nearly-uniform queries, then the (strong) tester of C' also makes nearly-uniform queries.*

Proof. Let $|I| = \alpha \cdot n$ for constant $0 \leq \alpha \leq 1$. Assume, without loss of generality, that $I = \{1, \ldots, \alpha \cdot n\}$. For every $i \in [t]$, we refer to an alleged (n-bit) codeword $C_i(x)$ as the pair of strings $(y_i, z_i) \in \{0,1\}^{\alpha \cdot n} \times \{0,1\}^{(1-\alpha)\cdot n}$, so that y_i is the common codeword $\widehat{C}(x)$ and z_i is the rest of the codeword.

We show a tester that, given oracle access to a binary string $w = \big((y_1, z_1), \ldots, (y_t, z_t)\big)$, where $(y_i, z_i) \in \{0,1\}^n$ for every $i \in [t]$, accepts every codeword of C' and rejects non-codewords of C' with probability that is polynomial in their relative distance from C'. The strong-LTC procedure for C' is described in Figure 3.

Note that Step 1 of the tester T invokes the tester of a uniformly selected inner code (C_i), and so, if the testers of $C_1, \ldots, C_t$ make nearly-uniform queries, then Step 1 of T also makes nearly-uniform queries. As for Step 2 of T (which queries a uniformly selected bit in two uniformly selected y_i's), note that by adding two dummy queries to the second part of

The strong-LTC Procedure for C'

Input: a string $\big((y_1,z_1),\ldots,(y_t,z_t)\big) \in \{0,1\}^{n\cdot t}$.

1. **The inner strong-LTC test:** Select at random $i \in [t]$, and run the strong-LTC tester of C_i on (y_i,z_i).
2. **The common codeword consistency test:** Select at random $i_1,i_2 \in [t]$ and $j \in [n]$, and reject if the j^{th} bit of y_{i_1} and y_{i_2} differs.

Figure 3 Strong local tester for C'.

each inner code (i.e., query a uniformly selected bit in two uniformly selected z_i's) we ensure that the first test also makes nearly-uniform queries.

The *completeness* of the tester is straightforward. If $\big((y_1,z_1),\ldots,(y_t,z_t)\big)$ is equal to $C'(x)$ for some $x \in \{0,1\}^k$, then: (1) for every $i_1,i_2 \in [t]$ it holds that $y_{i_1} = y_{i_2}$, and (2) for every $i \in [t]$ it holds that (y_i,z_i) is equal to $C_i(x)$. Thus the tester accepts.

Next, we show the *soundness* of the tester. Let $w = \big((y_1,z_1),\ldots,(y_t,z_t)\big)$ be $\delta_{C'}(w)$-far from the code C', let $u \in \{0,1\}^n$ be a string that minimizes the value of $\Delta\big((y_1,\ldots,y_t),u^t\big)$, and let $\gamma = \delta(\widehat{C})/36$. Suppose that $(y_1,\ldots,y_t)$ is $\gamma\cdot\delta_{C'}(w)$-far from u^t. In this case, the "common codeword consistency test" rejects with probability

$$\mathop{\mathbb{E}}_{i_1,i_2\in_R[t]}\left[\frac{\Delta(y_{i_1},y_{i_2})}{n}\right] \geq \mathop{\mathbb{E}}_{i_1\in_R[t]}\left[\frac{\Delta(y_{i_1},u)}{n}\right] = \frac{\Delta\big((y_1,\ldots,y_t),u^t\big)}{n\cdot t} = \gamma\cdot\delta_{C'}(w).$$

Thus, in the sequel, we assume that $(y_1,\ldots,y_t)$ is $\gamma\cdot\delta_{C'}(w)$-close to u^t.

Suppose that u is $3\gamma\cdot\delta_{C'}(w)$-far from $\widehat{C}$. Since $(y_1,\ldots,y_t)$ is $\gamma\cdot\delta_{C'}(w)$-close to u^t, at least half of the y_i's must be $2\gamma\cdot\delta_{C'}(w)$-close to u, so these y_i's are $\gamma\cdot\delta_{C'}(w)$-far from $\hat{C}$. Thus, in the invocation of the strong-LTC test of a random C_i, with probability $1/2$, the test is invoked on a string (y_i,z_i) such that y_i is $\gamma\cdot\delta_{C'}(w)$-far from the codewords of $\widehat{C}$. Since $|I| = |y_i|$ the tester will reject with probability $\Omega(\delta_{C'}(w))$. Hence, in the sequel, we assume that u is $3\gamma\cdot\delta_{C'}(w)$-close to a codeword of $\widehat{C}$. Since we also assume that $(y_1,\ldots,y_t)$ is $\gamma\cdot\delta_{C'}(w)$-close to u^t, then by the triangle inequality, the string $(y_1,\ldots,y_t)$ is $4\gamma\cdot\delta_{C'}(w)$-close to a (unique, since $4\gamma < \delta(\widehat{C})/2$) codeword $\widehat{C}^t(x)$. Furthermore, by an averaging argument, at most $\delta_{C'}(w)/8$ fraction of the y_i's are $\delta(\widehat{C})/2$-far from $\widehat{C}(x)$.

Since $|\hat{C}(x)|^t = \alpha\cdot|C'(x)|$ for a constant $\alpha \in (0,1)$, and since $(y_1,\ldots,y_t)$ is $4\gamma\cdot\delta_{C'}(w)$-close to $\widehat{C}^t(x)$, then $(z_1,\ldots,z_t)$ is $\delta_{C'}(w)/2$-far from any $(\hat{z}_1,\ldots,\hat{z}_t) \in \{0,1\}^{(n-|I|)t}$ such that $\Big(\widehat{C}^t(x),(\hat{z}_1,\ldots,\hat{z}_t)\Big)$ is a codeword of C'. Thus, at least $\delta_{C'}(w)/4$ fraction of the z_i's are $\delta_{C'}(w)/4$-far from their corresponding $\hat{z}_i$'s. Hence, at least $\delta_{C'}(w)/8$ fraction of the (y_i,z_i) pairs satisfy (1) y_i is $\delta(\widehat{C})/2$-close to $\widehat{C}(x)$, and (2) z_i is $\delta_{C'}(w)/4$-far from $\hat{z}_i(x)$. Therefore, if we invoke the verifier of C_i on such (y_i,z_i), it will reject with probability $\Omega(\delta_{C'}(w))$. Therefore, the tester T rejects with probability $\mathsf{poly}\big(\delta_{C'}(w)\big)$, as required. ◀

C Robustness of Tensor Codes

In this section we prove Theorem 2.6, which is implicit in [22]. Specifically, in [22, Theorem A.5] it is shown that for $d \geq 3$, if a codeword w of a d-dimensional tensor code $C^{\otimes d}$ is corrupted, then the corruption in a random hyperplane (i.e., a $d-1$-dimensional subplane) of w is proportional to the corruption in the entire (d-dimensional) tensor w. By applying this theorem recursively we obtain that for *constant* values of $d \geq 3$, the corruption in a random

2-dimensional plane of a corrupted codeword of $C^{\otimes d}$ is proportional to the corruption in the entire codeword. Formally, we show the following.

▶ **Theorem 2.6** (restated). *Let C be a linear binary code and $d \geq 3$ an integer. Then, there exists a constant $c_{\mathsf{robust}} \in (0,1)$ such that for every tensor $w \in \{0,1\}^{n^d}$ it holds that*

$$\mathop{\mathbb{E}}_{\mathsf{p} \in_R \mathsf{Planes}} \left[\delta\left(w|_{\mathsf{p}}, C^{\otimes 2}\right)\right] > c_{\mathsf{robust}} \cdot \delta_{C^{\otimes d}}(w).$$

We start by recalling the definition of robustness. Informally, we say that a tester is robust if for every word that is far from the code, the tester's view is far in expectation from any consistent view. This notion was defined for LTCs following an analogous definition for PCPs [1].

▶ **Definition C.1** (Robustness). Given a tester T for a code $C : \{0,1\}^k \to \{0,1\}^n$, for every word $w \in \{0,1\}^k$ we define

$$\rho^T(w) = \mathop{\mathbb{E}}_{I} \left[\delta(w|_I, C|_I)\right],$$

where $w|_I$ denotes the local view of the tester after querying on coordinates given by I. We say that the tester T has robustness ρ_C^T on the code C if for every $w \in \{0,1\}^k$ it holds that $\rho^T(w) \geq \rho_C^T \cdot \delta_C(w)$.

Next, we consider the "hyperplane tester for tensor codes" of Ben-Sasson and Sudan [2]. Towards this end, we first provide a notation for hyperplanes. For every $j \in [d]$, and $b \in [n]$, we say that τ is a (j,b)-hyperplane in $\{0,1\}^{n^d}$ if

$$\tau = \{(i_1, \ldots, i_{j-1}, b, i_{j+1}, \ldots, i_d) \ : \text{ for all } t \in [d] \setminus \{\, j \,\} \text{ we have } i_t \in [n]\}\,.$$

We denote by $\mathsf{Hyperplanes} = \{(j,b)\text{-hyperplane}\}_{\{j \in [d], b \in [n]\}}$ the set of all hyperplanes in $\{0,1\}^{n^d}$, and denote the restriction of a tensor $w \in \{0,1\}^{n^d}$ to a hyperplane $\tau \in \mathsf{Hyperplanes}$ by $w|_\tau \in \{0,1\}^{n^{d-1}}$.

▶ **Definition C.2** (Hyperplane Tester for Tensor Codes). Let C be a linear code, $d \geq 3$ an integer, and $w \in \{0,1\}^{n^d}$. The hyperplane tester for $C^{\otimes d}$ selects uniformly at random $\tau \in \mathsf{Hyperplanes}$, obtains $w|_\tau$ by querying all points on τ, and accepts if and only if $w|_\tau \in C^{\otimes d-1}$.

▶ **Theorem C.3** ([22, Theorem A.5]). *Let C be a linear code and $d \geq 3$. Let T be the hyperplane tester for $C^{\otimes d}$. Then, $\rho^T_{C^{\otimes d}} \geq \frac{\delta(C)^d}{2d^2}$.*

We show that Theorem 2.6 follows by iterative applications of Theorem C.3.

Proof of Theorem 2.6. Let C be a linear code and $d \geq 3$ a constant integer. Let $w \in \{0,1\}^{n^d}$ be a tensor. For every $3 \leq t \leq d$, let T_t be the hyperplane tester for $C^{\otimes t}$. Note that for every $3 \leq t \leq d$, the tester T_t queries a hyperplane that is allegedly a codeword of $C^{\otimes t-1}$; hence T_{t-1} can be composed with T_t; that is, we can run T_t on input w, during which T_t generates a local view $w|_I$ to be queried, and so, we can run T_{t-1} on the local view $w|_I$. (Note that the composed tester $T_3 \circ \ldots \circ T_d$ queries the restriction of the input w to a uniformly selected plane $\mathsf{p} \in \mathsf{Planes}$.) The robustness of the composed tester will hence be

$$\rho^{T_3 \circ \ldots \circ T_d}_{C^{\otimes d}} \geq \rho^{T_d}_{C^{\otimes d}} \cdot \rho^{T_{d-1}}_{C^{\otimes d-1}} \cdot \ldots \cdot \rho^{T_3}_{C^{\otimes 3}}.$$

By Theorem C.3, for every $t \geq 3$ we have $\rho^{T_t}_{C^{\otimes t}} \geq \frac{\delta(C)^t}{2t^2}$. Thus, for constant $d \geq 3$ it holds that $c_{\mathsf{robust}} \triangleq \rho^{T_3 \circ \ldots \circ T_d}_{C^{\otimes d}}$ is a positive constant that depends only on $\delta(C)$ and d. ◀

D Average Smoothness and Error Reduction for Relaxed LDCs

In this appendix, following [1, Section 4.2], we show that the modified definition of relaxed-LDCs (see Definition 4.2) implies the standard definition of relaxed-LDCs (see Definition 2.2). Towards this end we need to show the following: (1) The soundness can be increased from $\Omega(1)$ (as in Condition 2 of Definition 4.2) to 2/3 (as in Condition 2 of Definition 2.2), and (2) the *average smoothness* (i.e., Condition 3 of Definition 4.2) can be replaced with the *success rate* condition (i.e., Condition 3 of Definition 2.2). Both claims were shown in [1]; we provide their proofs (adapted to our settings) for completeness.

We start by showing how to perform error-reduction for relaxed-LDC with soundness $\Omega(1)$. Recall that the decoder is required to successfully decode each valid codeword, and in addition, given a somewhat corrupted codeword the decoder is required to either decode successfully or abort with probability $\Omega(1)$. On the face of it, it may seem that standard error reduction cannot be applied (since we start with a large error probability). However, the error reduction can be simply performed by repeating the execution of the decoder, outputting a bit only if all invocations returned this bit, and aborting otherwise. We remark that the above may cause an increase in the number of indices on which the decoder aborts (with probability at least 2/3). However, in the modified definition (i.e., Definition 4.2) there is no restriction on the success rate.

▶ **Proposition D.1.** *Let $C : \{0,1\}^k \to \{0,1\}^n$ be a modified* relaxed-LDC*, according to Definition 4.2. Then, C has a modified* relaxed-LDC *decoder that also satisfies Condition* 2 *of Definition 2.2.*

Proof. Let C be a modified relaxed-LDC. Denote its decoder by D. There exists a constant $p > 0$ such that for every string w that is sufficiently close to a codeword of C it holds that $\Pr_D[D^w(i) = \{x_i, \perp\}] \geq p$. Consider a decoder D' that operates follows: D' executes the original decoder D (with fresh randomness) for r times, where r is a constant to be determined later. If all of the executions are consistent, i.e., there exists an $a \in \{0,1,\perp\}$ such that in every execution $D^w(i) = a$, then D' output a; otherwise, D' output $\perp$. (We remark that the distribution of queries of D' is identical to that of D, and thus D' also satisfies the *average smoothness* condition.)

Note that the new decoder D' satisfies Condition 1 of Definition 2.2 (the *completeness* condition). Moreover, D' satisfies Condition 2 of Definition 2.2: Indeed, given w that is sufficiently close to $C(x)$, the probability that D' errs is at most $p' = (1-p)^r$. Hence, by fixing $r = 2/p$ we get that $\Pr_{D'}[D'^w(i) = \{x_i, \perp\}] \geq 1 - p' \geq 2/3$, as needed. ◀

Finally, we show that the *average smoothness* condition (i.e., Condition 3 of Definition 4.2) can be replaced by the *success rate* condition (i.e., Condition 3 of Definition 2.2, which limits the number of indices upon which the decoder aborts (with probability at least 2/3)). The key idea is that a decoder that satisfies the completeness and soundness conditions (i.e., Conditions 1 and 2 of Definition 2.2) only aborts if the local view of the codeword that it queries contains a corrupted point. By the *average smoothness*, on average the decoder will only query a corrupted point with low probability. Thus, by an averaging argument, we can deduce that there is a small number of indices upon which the decoder might abort.

▶ **Proposition D.2.** *Let $C : \{0,1\}^k \to \{0,1\}^n$ be a linear code, and let D be a constant-query decoder for C that satisfies Conditions* 1 *and* 2 *of Definition 2.2 as well as Condition* 3 *of Definition 4.2 (i.e., average smoothness). Then, C satisfies all three conditions of Definition 2.2.*

Proof. Let the code C and the decoder D be as in the hypothesis of the proposition. Denote the (constant) query complexity of D by q. According to Condition 1, for any $x \in \{0,1\}^k$ and every $i \in [k]$, it holds that $\Pr\left[D^{C(x)}(i) = x_i\right] = 1$. Considering any w that is δ-close to $C(x)$ (where $\delta \leq \delta_{\mathsf{radius}}$), the probability that given a *uniformly distributed* index $i \in [k]$ the decoder D queries a location on which w and $C(x)$ disagree is at most $q \cdot (2/n) \cdot \delta n = 2q\delta$. This is due to the fact that, for a uniformly distributed i, no position is queried with probability greater than $2/n$.

Let p_i^w denote the probability that on input i the decoder D queries a location on which w and $C(x)$ disagree. We have just established that $(1/k) \cdot \sum_{i=1}^{k} p_i^w \leq 2q\delta$. By an averaging argument, for $I_w \triangleq \{i \in [k] : p_i^w \leq 1/3\}$, it holds that $|I_w| \geq (1 - 6q\delta) \cdot k$. Observe that for any $i \in I_w$, it holds that $\Pr[D^w(i) = x_i] \geq 1 - 1/3 = 2/3$, as required. ◀

E Proof of Claim 5.6

In this section we provide the proof of Claim 5.6. The proof is similar to the proof of Claim 5.5. However, note that Claims 5.5 and 5.6 deal with different objects: While Claim 5.5 deals with the planes of the tensor code and the *plane* scPCPPs, Claim 5.6 deals with the lines of the tensor and the *point-line* scPCPPs. In particular, every plane in the tensor code is coupled with a *unique* plane scPCPP proof, whereas every line in the tensor code is coupled with n *different* point-line scPCPPs, one for each point on the line. We begin by restating Claim 5.6. Recall that $\gamma = \delta(C)/(24d)$.

▶ **Claim 5.6** (restated). *Assuming $\bar{c}$ is $\gamma \cdot \delta_{C'}(w)$-close to being a codeword of C^{t_1}, if $\delta_{\bar{p}^{\mathsf{lines}}} > \delta_{C'}(w)$, then $\Pr_T[T^w = 0] \geq \mathsf{poly}\big(\delta_{C'}(w)\big)$.*

Proof. By the lemma's hypothesis, $\bar{c}$ is $\delta_{\bar{c}}$-close to $C(x)^{t_1}$, where $\delta_{\bar{c}} \leq \gamma \cdot \delta_{C'}(w)$. By an averaging argument, with probability at least $2/3$ the random copy $\mathbf{c}$ is $3\delta_{\bar{c}}$-close to $C(x)$. We say that a point $\bar{i} \in [n]^d$ in $\mathbf{c}$ is corrupted if $\mathbf{c}_{\bar{i}} \neq C'(x)_{\bar{i}}$ and so, there are at most $3\delta_{\bar{c}} n^d$ corrupted points in $\mathbf{c}$. Since there are $d \cdot n^{d-1}$ axis-parallel lines in $\mathbf{c}$, then on average, the number of corrupted points in a random axis-parallel line is at most $\frac{3\delta_{\bar{c}} n^d}{d \cdot n^{d-1}} \leq 3\delta_{\bar{c}} n$. Thus, by an averaging argument, we obtain that at most $\frac{\delta_{\bar{p}}}{4}$ fraction of the axis-parallel lines in $\mathbf{c}$ contain at least $\frac{4}{\delta_{\bar{p}}} \cdot 3\delta_{\bar{c}} n$ corrupted points.

Recall that every axis-parallel line ℓ has n corresponding *point-line* scPCPP proofs (one for each point on ℓ). For every line ℓ we view these n proofs as one concatenated proof for the line ℓ. By an averaging argument, with probability at least $\delta_{\bar{p}} \triangleq \delta_{\bar{p}^{\mathsf{lines}}}/2$ the random copy $\bar{p}$ in $\bar{p}^{\mathsf{lines}}$ is $\delta_{\bar{p}}$-far from its corresponding set of canonical proofs, $\pi_{\mathsf{lines}}(x)$. Assume from now on that $\bar{p}$ is $\delta_{\bar{p}}$-far from $\pi_{\mathsf{lines}}(x)$. By another averaging argument, at least a $\delta_{\bar{p}}/2$ fraction of the concatenated line proofs (i.e., proofs which consists of n point-line scPCPP proofs) are $\delta_{\bar{p}}/2$-far from their corresponding (concatenated) canonical line proofs.

By combining the conclusions of the last two paragraphs, we deduce that $\Omega\big(\delta_{C'}(w)\big)$-fraction of the axis-parallel lines ℓ in $\mathbf{c}$ are both $\delta(C_0)/2$-close to the restriction of the tensor codeword $C(x)$ to ℓ, and their corresponding (concatenated) proofs are $\Omega\big(\delta_{C'}(w)\big)$-corrupted; that is, there is a subset of lines, denoted BAD, which consists of at least $\frac{\delta_{\bar{p}}}{4}$ fraction of all the lines in $\mathbf{c}$ that are $\delta(C_0)/2$-close to $C(x)|_\ell$ (recall that $\delta_{\bar{c}} \leq \gamma \cdot \delta_{C'}(w)$ and $\delta_{\bar{p}} > \delta_{C'}(w)$, therefore $\frac{12 \cdot \delta_{\bar{c}}}{\delta_{\bar{p}}} < \delta(C_0)/2$), and in addition satisfy the following: For every $\ell \in \mathrm{BAD}$, the n (alleged) *point-line* scPCPP proofs that correspond to ℓ are $\delta_{\bar{p}}/2$-far from their (correct) canonical proofs in $\pi_{\mathsf{lines}}(x)$. By an averaging argument, for every $\ell \in \mathrm{BAD}$ it holds that $\delta_{\bar{p}}/4$ fraction of the point-line PCPP proofs that correspond to the line ℓ (recall that there are n such proofs) are $\delta_{\bar{p}}/4$-far from their canonical proof in $\pi_{\mathsf{lines}}(x)$.

Recall that the tester chooses a line $\ell = \ell_{j,\bar{\imath}}$ by sampling uniformly at random a point $\bar{\imath} \in [n]^d$ and a direction $j \in [d]$. Notice that for if $\ell \in \text{BAD}$, then with probability $\delta_{\bar{p}}/4$, in order for input $\mathbf{c}|_\ell$ and the proof $p^{\ell_{j,\bar{\imath}}}$ (that refers to the same line as ℓ) to be a valid claim for the input-proof language that $V^{\mathsf{line}}(i_j, \mathbf{c}_{\bar{\imath}})$ verifies, one must make at least one of the following changes: (1) change a fraction of at least $\frac{\delta_{\bar{p}}}{4}$ of the proof $p^{\ell_{j,\bar{\imath}}}$ such that it matches $\pi_{\mathsf{line}}\left(C(x)|_{\ell_{j,\bar{\imath}}}, i_j\right)$, or (2) change a fraction of at least $\delta(C_0)/2$ of $\mathbf{c}|_\ell$ (since $p^{\ell_{j,\bar{\imath}}}$ might be a valid proof for input $C_0(y) \neq \mathbf{c}|_\ell$). Thus, for every $\ell_{j,\bar{\imath}} \in \text{BAD}$, the probability that $V^{\mathsf{line}}(i_j, \mathbf{c}_{\bar{\imath}})$ rejects input $c|_{\ell_{j,\bar{\imath}}}$ and proof $p^{\ell_{j,\bar{\imath}}}$ is at least polynomial in $\delta_{C'}(w)$.

Putting it all together, with probability $2/3$ we hit a random copy $\mathbf{c}$ of the tensor code that is $3\delta_{\bar{c}}$-close to $C(x)$. Furthermore, with probability at least $\delta_{\bar{p}}$ we hit a random copy $\bar{p}$ that is $\delta_{\bar{p}}$-corrupted, and subsequently, with probability $\delta_{\bar{p}}/2$ we hit a set of n line scPCPP proofs that are $\delta_{\bar{p}}/2$-corrupted. Moreover, with probability at least $\delta_{\bar{p}}/4$ we hit a point-line scPCPP proof that is $\delta_{\bar{p}}/4$ corrupted. Finally, assuming the foregoing, the corresponding scPCPP verifier rejects with probability $\mathsf{poly}\left(\delta_{C'}(w)\right)$. Therefore,

$$\Pr_T[T^w = 0] \geq \frac{2}{3} \cdot \delta_{\bar{p}} \cdot \frac{\delta_{\bar{p}}}{2} \cdot \frac{\delta_{\bar{p}}}{4} \cdot \mathsf{poly}\left(\delta_{C'}(w)\right) \geq \mathsf{poly}\big(\delta_{C'}(w)\big).$$

◀