

Completeness Amplification of PCPs and Its Applications

Prasant Gopal*

Dana Moshkovitz[†]

August 23, 2012

Abstract

We construct the first PCP verifier that performs linear tests and has polynomially small completeness error simultaneously with logarithmic randomness, constant number of queries, constant alphabet and constant soundness error. Prior to our work, the smallest completeness error that was known in such circumstances was an arbitrarily small constant.

We use the verifier we construct to explore several problems:

1. *Threshold behavior:* We study MAX-3LIN and MAX-3SAT around their approximability thresholds. Specifically, we show that MAX-3LIN and MAX-3SAT become NP-hard to approximate at $1/2 + 1/(\log n)^\alpha$ and $7/8 + 1/(\log n)^\alpha$, respectively, for some $\alpha > 0$. Moreover, we identify the bottleneck for achieving an optimal (up to the constant α) hardness of $1/2 + 1/n^\alpha$ and $7/8 + 1/n^\alpha$. We show that further improvements in low error projection games could replace the $1/(\log n)^\alpha$ in our results with $1/n^\alpha$.
2. *Polynomial hardness factors:* We show that the NEAREST-CODEWORD-PROBLEM for binary linear codes, as well as the MIN-3LIN-DELETION problem are NP-hard up to a polynomial approximation factor.

*prasant@csail.mit.edu, CSAIL, MIT

[†]dmoshkov@csail.mit.edu, CSAIL, MIT

1 Introduction

The Probabilistically Checkable Proofs (PCP) Theorem shows how to efficiently transform any mathematical proof to a format that can be probabilistically verified by reading only a constant number of locations. A PCP verifier has a number of parameters including the number of queries, the randomness, the alphabet, and the error parameters. There can be two types of errors: the verifier may reject given a correct statement and proof (completeness error), or it may accept given an incorrect statement (soundness error). If the verifier never rejects when given a correct statement and proof, we say that it has *perfect completeness*.

The basic PCP theorem was established through the works of Arora and Safra [AS98]; and Arora et al. [ALM⁺98]. They relied on a series of previous works including those of Feige et al. [FGL⁺96]; Lund et al. [LFKN90]; Babai et al. [BFL91, BFLS91]; Rubinfeld and Sudan [RS96]. The PCP Theorem states that every language in NP has a verifier that on input size n , uses $O(\log n)$ random bits, makes two queries to a proof over constant sized-alphabet; where the answer to the first query determines at most one satisfying answer to the second query (this is known as the “projection property”). The verifier always accepts a correct proof, and rejects any incorrect statement with a constant probability. The PCP theorem is the basis of essentially all hardness of approximation results known today. For more information about it, see [Din10].

Decreasing the error in PCP is the subject of considerable research. This is motivated both by the intrinsic interest in trustworthy verifiers, and by the use of PCP for hardness of approximation, in which PCPs with low error play a crucial role. Since there are PCP constructions with perfect completeness, most of the work concentrated on decreasing the soundness error [Raz98, RS97, AS03, DFK⁺99, MR10, IKW09, DM11].

For certain verifiers, one cannot achieve perfect completeness. One example is unique verifiers, where the verifier makes two queries, and the answer to any one query determines uniquely a satisfying answer to the other query. Another example is linear verifiers, where the alphabet is a finite field, and the verifier performs linear tests over the field. Both types of verifiers were discovered to be very useful for hardness of approximation. The first gives rise to the host of UGC-based (UGC stands for Unique Games Conjecture [Kho02]; see the survey [Kho10]) hardness results we know today. The second allows using the Hadamard code in lieu of the long code [Kho01].

In this paper, we construct the first linear verifier with polynomially small completeness error:

Theorem 1.1. *There is a constant $\alpha > 0$, such that every NP language L has a linear PCP verifier with the projection property, such that on input size n , the verifier uses $O(\log n)$ random bits, queries a proof over a constant-size alphabet, has completeness error $1/n^\alpha$ and has soundness error 0.9.*

Given this theorem, it is possible to apply a soundness amplification technique and get a linear verifier with low soundness error in addition to low completeness error. For instance, using the best soundness amplification technique known today for verifiers with the projection property [MR10] (which can be shown to preserve linearity), we deduce a statement similar to Theorem 1.1 but with lower, poly-logarithmically small, soundness error:

Corollary 1.1.1. *There is a constant $\alpha > 0$, such that every NP language L has a linear PCP verifier with the projection property, such that on input size n , the verifier uses $O(\log n)$ random bits, queries a proof over $\text{poly}(n)$ -size alphabet, has completeness error $1/n^\alpha$ and has soundness error $1/(\log n)^\alpha$.*

We note that the alphabet of the proof must grow to allow lower soundness error, and this is the reason for the difference in the alphabet size between Theorem 1.1 and Corollary 1.1.1.

Ideally, one could hope for polynomially-small soundness error in Corollary 1.1.1. The logarithmically small soundness error is due to the technique of [MR10]. We believe (see [Mos12]) that the soundness error of PCPs with the projection property could be polynomially small. Specifically, the soundness error in Corollary 1.1.1 could be replaced by $1/n^\alpha$, if the following conjecture holds:

Conjecture 1.2 (Soundness amplification for linear projection). *There exists $\beta > 0$, such that for every n and $\varepsilon = \varepsilon(n) \geq 1/n^\beta$, a linear PCP verifier with the projection property that uses r random bits, queries a proof over alphabet Σ , has completeness error δ , and has soundness error 0.9 can be efficiently transformed into a linear PCP verifier with the projection property that uses $(1 + o(1))r + O(\log(1/\varepsilon))$ random bits, queries a proof over alphabet Σ' , $|\Sigma'| \leq \text{poly}(1/\varepsilon)$, has completeness error $O(\log(1/\varepsilon)) \cdot \delta$, and has soundness error ε .*

1.1 Application to The Threshold Behavior of Approximation

The work of Håstad [Hås01], relying on previous works by Raz [Raz98] and Bellare, Goldreich and Sudan [BGS98], showed that many optimization problems Π undergo a phase transition in their approximability: up to some approximation factor α_Π , they can be efficiently approximated, while for any $\varepsilon > 0$, approximation better than $\alpha_\Pi + \varepsilon$ (or $\alpha_\Pi - \varepsilon$ for minimization problems) is NP-hard.

A subsequent work by Moshkovitz and Raz [MR10] strengthened this result, establishing that for the problems Håstad considered: (1) The threshold is sharp, and the phase transition occurs in a window of width $\varepsilon = o(1)$; (2) Beyond the threshold window, the problems are not only NP-hard, but in fact have a similar time lower bound as (exact) SAT, which is conjectured to require exponential time (Previously, if SAT required time T , one could only show lower bounds of the form $T^{\Omega(1/\log(1/\varepsilon))}$ for Π).

The work of Moshkovitz and Raz did not concentrate on estimating the width ε of the threshold window, only on showing that it goes down to 0 with n . A natural question is to understand what the width of the threshold window is. This question is further motivated by the recent realization that many optimization problems have non-trivial, sub-exponential time approximation algorithms around their approximation threshold. Unique games [ABS10] is one example, and SET-COVER [CKW09] and MAX-CLIQUE are other examples. Understanding for which approximation factors we can expect such algorithms is important.

A different analogy can be made to the research of random graphs, which is another area where phase transitions and sharp thresholds arise. Recent exploration of the threshold window there (see, e.g., [BBC⁺01], and many others) revealed surprisingly interesting behavior. We wonder whether similarly interesting behavior could be obtained for approximation problems.

For convenience, we concentrate on two of the problems that were considered by Håstad, MAX-3LIN and MAX-3SAT, and use Corollary 1.1.1 to study their approximability around their thresholds, $1/2$ and $7/8$, respectively. For both problems, a simple random assignment algorithm gives an approximation up to the threshold. Moreover, for any constant larger than the threshold, Håstad proved that the problems become NP-hard to approximate [Hås01]. Moshkovitz and Raz refined the result to show that for an additive $1/(\log \log n)^\alpha$ beyond the threshold ($\alpha > 0$ is some constant), the problem has a nearly-exponential time lower bound, assuming SAT requires exponential time [MR10]. For practical values of n , the term $1/(\log \log n)^\alpha$ is quite large.

We prove that the window of efficient approximability is in fact much narrower than $1/(\log \log n)^\alpha$:

Theorem 1.3. *There is a constant $\alpha > 0$ for which it is NP-hard to approximate MAX-3LIN to within a factor better than $1/2 + 1/(\log n)^\alpha$, and MAX-3SAT to within a factor better than $7/8 + 1/(\log n)^\alpha$.*

Moreover, if Conjecture 1.2 holds, then our methods yield hardness for MAX-3LIN and MAX-3SAT up to $1/2 + 1/n^\alpha$ and $1/2 + 1/n^\alpha$ for some constant $\alpha > 0$, respectively. Up to the constant α , this matches the polynomial-time approximation of $1/2 + 1/\sqrt{n}$ achieved by Håstad for MAX-3LIN [Hås00].

1.2 Application to Min-3Lin-Deletion and Nearest-Codeword-Problem

MIN-3LIN-DELETION is the problem in which, given a system of linear equations over $GF(2)$, where each equation depends on three variables, the task is to find the minimum number of equations that their removal leaves a fully satisfiable system.

NEAREST-CODEWORD-PROBLEM is the problem in which, given the generator matrix of a binary linear code, and a purported codeword, the task is to find the Hamming distance of the purported codeword from the code.

What both problems have in common is that their approximation factor is dominated by the completeness parameter, rather than the soundness parameter. Specifically, if we construct MIN-3LIN-DELETION instances in which there is an α fraction of equations that could be removed to make the system satisfiable, while we argue that it NP-hard to find even half of the equations whose removal would suffice, then we prove the hardness of approximation of the problem up to a factor $1/2\alpha$. Similarly, if we construct NEAREST-CODEWORD-PROBLEM instances in which the purported codeword is α -close to the code, while we argue that it NP-hard to find an actual codeword that is $2/3$ -close to the purported codeword, then we prove the hardness of approximation of the problem up to a factor $2/3\alpha$. In both cases, the factor is essentially determined by α .

Hence, our completeness amplification technique in Corollary 1.1.1 yields the first polynomially large hardness of approximation factors for MIN-3LIN-DELETION and NEAREST-CODEWORD-PROBLEM. This is optimal for these problems, up to the constant in the exponent.

Theorem 1.4. *There is a constant $\beta > 0$ for which it is NP-hard to approximate MIN-3LIN-DELETION and NEAREST-CODEWORD-PROBLEM on instances of size n to within a factor better than $1/n^\beta$.*

Similar results can be achieved for any other problem in which the completeness error dominates the approximation factor.

Previously, Arora *et al.* [ABSS93] established NP-hardness of approximation for NEAREST-CODEWORD to within any constant, and hardness of $2^{\log^{(1-\epsilon)} n}$ for any $\epsilon > 0$ under $\text{NP} \not\subseteq \text{DTIME}(2^{\text{polylog}})$. This was improved to inapproximability to within $n^{1/O(\log \log n)}$ under $\text{P} \neq \text{NP}$ by Dinur *et al.* [DKRS03]. On the algorithmic front, Berman and Karpinski [BK02] gave a randomized algorithm for $\epsilon \cdot n / \log n$ approximation, for any fixed $\epsilon > 0$ and $\epsilon \cdot n$ approximation in deterministic time.

For MIN-3LIN-DELETION, the best hardness of approximation was to within polylogarithmic factors under the assumption that NP does not have quasi-polynomial time algorithms [KP06]. We remark that in [KP06], this result is used to establish improved hardness results for MAX-CLIQUE and CHROMATIC-NUMBER, but our improvement for MIN-3LIN-DELETION does not yield an improvement for the latter.

1.3 The Completeness Amplification of Khot and Ponnuswami

The only previous work on completeness amplification that the authors are aware of is by Khot and Ponnuswami [KP06]. They construct linear PCP verifiers with completeness error $2^{-\Omega(\sqrt{\log n})}$ and large blow-up $2^{(\log n)^{O(1)}}$ (which corresponds to poly-logarithmic randomness), and apply their

construction to yield stronger hardness of approximation results for MIN-3LIN-DELETION, MAX-CLIQUE and CHROMATIC-NUMBER.

Next we describe the technique of Khot and Ponnuswami, which is the basis of our results. Suppose that we have a PCP reduction that given a SAT instance φ , produces a system \mathcal{L} of linear equations (which is equivalent to a linear verifier). If φ is satisfiable, then there is an assignment to the variables of the linear system that satisfies all the equations, except perhaps of a δ fraction. On the other hand, if φ is not satisfiable, then any assignment to the variables of the linear system satisfies at most s fraction of the equations. Now, suppose that we want to decrease δ , the completeness error. A natural way to do that would be to consider new equations over the same variables. Each new equation takes two equations from \mathcal{L} , say, $e_1 = 0$ and $e_2 = 0$, and is defined as $e_1 \cdot e_2 = 0$. Since the new equation is not satisfied only if both $e_1 = 0$ and $e_2 = 0$ are not satisfied, the new system has parameters δ^2 and $(1 - s)^2$, instead of δ and $1 - s$, respectively. This simple transformation has one major caveat: it produces quadratic equations instead of linear equations. What Khot and Ponnuswami suggest is to linearize the system, by replacing products of the form $x_i x_j$ with a new variable $x_{i,j}$. This is equivalent to *tensoring* the system.

Khot and Ponnuswami argue that tensoring has a similar effect on the parameters of the system as taking quadratic products. They move on to handle two side-effects: (1) The soundness error increases; (2) The number of variables each equation depends on increases. They solve both problems using standard PCP techniques: (1) is solved by soundness amplification and (2) is solved by composition. These operations are applied iteratively to achieve a low completeness error.

What Khot and Ponnuswami do not solve is the effect tensoring has on the size of the system: the number of variables squares. This is the reason that their reduction has a large blow-up.

A similar blow-up occurs with the parallel repetition theorem of Raz [Raz98]. The purpose of the latter is to decrease the soundness error (and not the completeness error), while keeping each equation dependent on two variables (and not increasing the number of variables per equation). Even though parallel repetition requires a much more challenging analysis, the two constructions are similar in that both rely on tensoring of the original system. Similarly, both construction inherently raise the number of variables to a power that depends inversely on the desired error, and thus both incur a large blow-up.

1.4 Our Completeness Amplification

In this work we develop a completeness amplification technique that does not involve large blow-ups. Our idea is to apply the completeness amplification technique of Khot and Ponnuswami, but only on constant-size instances. This allows us to iterate their technique a logarithmic number of times, and obtain polynomially small completeness error, without introducing a large blow-up.

We then compose this construction as an inner PCP together with an outer PCP that has perfect completeness, resulting in: (1) The tests are linear as those of the inner PCP; (2) The completeness error is the sum of the outer and inner completeness errors, which in this case equals the inner completeness error; (3) The blow-up is inherited from both the outer and inner constructions, but the blow-up of the outer construction can be made low using existing PCP technology, while the blow-up of the inner can be low if we tune the number of iterations accordingly.

One subtlety is that an inner construction in PCP composition is used, not only to verify that a satisfying assignment exists, but also to decode information about the satisfying assignment. This information is then used to check consistency between different inner assignments. The need in decoding requires us to prove that any assignment that satisfies the maximum number of equations of the tensored system has, as a sub-assignment, an assignment to the original variables that satisfies the maximum number of equations of the original system.

In contrast, attempts to get a decoding version of parallel repetition were shown to fail [DG10] (At the same time, parallel repetition with additional consistency checks does have a decoding version [IKW09]. However, our version of tensoring, which preserves linearity, is analogous to the basic parallel repetition, without consistency checks).

2 Preliminaries

Throughout this section we work with the binary alphabet $\Sigma = \{0, 1\}$, and arithmetic over the alphabet is over the field $GF(2)$.

2.1 Hamming Weight

We start by introducing the notion of Hamming weight and its corresponding metric the Hamming distance. For any $\mathbf{x} \in \Sigma^n$, the hamming weight of \mathbf{x} is the number of nonzero coordinates of \mathbf{x} , also known by $wt(\mathbf{x})$. The weight function is a norm and the corresponding metric $d(\mathbf{x}, \mathbf{y})$ is the weight of difference between \mathbf{x} and \mathbf{y} , that is, $wt(\mathbf{x} + \mathbf{y})$. It is often useful to normalize the Hamming weight (resp. distance) w.r.t. n . From now on, we use normalized Hamming weight, which is $wt(\mathbf{x})/n$. Also, we use bold face small case letters \mathbf{x}, \mathbf{y} to denote column vectors and upper case bold letters like \mathbf{A}, \mathbf{X} to denote matrices. We use the notation $(\cdot)^T$ to denote the transpose of the corresponding vector or matrix.

Claim 2.1. *For every $\mathbf{x}, \mathbf{y}, \mathbf{z}$, $d(\mathbf{x}, \mathbf{y}) = d(\mathbf{x} + \mathbf{y}, \mathbf{y} + \mathbf{z})$.*

Proof. Say that \mathbf{x}, \mathbf{y} agree on a coordinate i , then they continue to remain so even after adding \mathbf{z} to both \mathbf{x}, \mathbf{y} . Similarly, if \mathbf{x}, \mathbf{y} disagree on the coordinate. \square

2.2 Linear Systems of Equations

Let \mathcal{L} , $\mathcal{L} \equiv \mathbf{Ax} + \mathbf{y} = \mathbf{0}$, be a linear system with m equations over n variables \mathbf{x} . We use the notation $UNSAT(\mathcal{L}, \mathbf{x})$ to denote the fraction of unsatisfied equations in \mathcal{L} when its variables are assigned to \mathbf{x} . Note that, this is exactly same as the Hamming weight of $\mathbf{Ax} + \mathbf{y}$. We define the UNSAT value of \mathcal{L} as follows.

Definition 1 ($UNSAT(\mathcal{L})$). *The UNSAT value of \mathcal{L} is defined to be minimum fraction of unsatisfied equations over all possible assignments to the variables.*

Proposition 2.2. *The following holds:*

$$UNSAT(\mathcal{L}) = \min_{\mathbf{x}} d(\mathbf{Ax}, \mathbf{y}) = \min_{\mathbf{x}} d(\mathbf{Ax} + \mathbf{y}, \mathbf{0}) \quad (1)$$

Proof. Recall that, UNSAT value of \mathcal{L} is the minimum Hamming weight of $\mathbf{Ax} + \mathbf{y}$ over all \mathbf{x} . This settles that the first and the last expression are the same. Now, subtract \mathbf{y} from both terms of $d(\mathbf{Ax} + \mathbf{y}, \mathbf{0})$. We now invoke Claim 2.1 to establish that shifts preserve Hamming distances. And this establishes the truth of (1). \square

3 Tensoring

We start by defining the tensor of n -dimensional vector \mathbf{x} over a binary alphabet.

Definition 2 (Tensor). We define the tensor of a vector $\mathbf{x} \in \{0, 1\}^n$ as the $n \times n$ matrix $\mathbf{x}^{\otimes 2}$ that has in its i, j position the entry $x_i \cdot x_j$.

We often treat matrices as vectors, so an $n \times n$ matrix corresponds to a vector of length n^2 .

The tensor $\mathcal{L}^{\otimes 2}$ of the system \mathcal{L} involves taking the products of all possible pairs of (left hand sides of) linear equations from \mathcal{L} , and then linearizing them via replacing terms of the form $x_i \cdot x_j$ with x_{ij} and x_i with x_{ii} (recall that in $GF(2)$ it holds that $x_i^2 = x_i$). The claim that we make about the aforementioned transformation is that the UNSAT values of \mathcal{L} and its tensor are very closely related. In what follows, we make this connection explicit.

Definition 3. Given a linear system \mathcal{L} , $\mathcal{L} \equiv \mathbf{Ax} + \mathbf{y} = \mathbf{0}$, over variables $\mathbf{x} = (x_1, \dots, x_n)$, the tensor of \mathcal{L} is defined in two steps: (1) Consider the quadratic system $(\mathbf{Ax} + \mathbf{y}) \cdot (\mathbf{Ax} + \mathbf{y})^T$. (2) Linearize the quadratic system by replacing every term of the form $x_i \cdot x_j$ for $i, j \in [n]$ by X_{ij} , and every term of the form x_i for $i \in [n]$ with X_{ii} . We denote this linearized system by $\mathcal{L}^{\otimes 2}$. We use the matrix \mathbf{X} to denote the variables X_{ij} of this system.

As a matter of convention, we associate with \mathbf{X} , the variables of the tensored system, its diagonal $\mathbf{x} = (X_{11}, \dots, X_{nn})$, as a candidate assignment to the original system. Note that when $\mathbf{X} = \mathbf{x}^{\otimes 2}$, it indeed holds that \mathbf{X} 's diagonal is \mathbf{x} . Using this convention, we can develop an expression for the system $\mathcal{L}^{\otimes 2}$ as a function of the ingredients of \mathcal{L} , i.e., the matrix \mathbf{A} and the vector \mathbf{y} :

$$\mathbf{AXA}^T + \mathbf{y}(\mathbf{Ax})^T + (\mathbf{Ax} + \mathbf{y}) \cdot \mathbf{y}^T \quad (2)$$

Proposition 3.1. For every linear system \mathcal{L} ,

$$\text{UNSAT}(\mathcal{L}^{\otimes 2}, \mathbf{x}^{\otimes 2}) = \text{UNSAT}(\mathcal{L}, \mathbf{x})^2.$$

Proof. When $\mathbf{X} = \mathbf{x}^{\otimes 2}$, the tensored system $\mathcal{L}^{\otimes 2}$ is the same as the quadratic system from which it was generated. An equation in the quadratic system is not satisfied if and only if both of the equations it consists of were not satisfied. The relation stated in the proposition follows. \square

Lemma 3.2. For every linear system \mathcal{L} over variables \mathbf{X} with diagonal \mathbf{x} ,

$$\text{UNSAT}(\mathcal{L}^{\otimes 2}, \mathbf{X}) \geq \text{UNSAT}(\mathcal{L}, \mathbf{x}) \cdot \text{UNSAT}(\mathcal{L}).$$

In particular, $\text{UNSAT}(\mathcal{L}^{\otimes 2}) \geq \text{UNSAT}(\mathcal{L})^2$.

Proof. Let us use the following notation for the two parts of $\mathcal{L}^{\otimes 2}$ as in Equation (2):

$$\mathbf{P} := \mathbf{AXA}^T + \mathbf{y}(\mathbf{Ax})^T; \quad \mathbf{Q} := (\mathbf{Ax} + \mathbf{y})\mathbf{y}^T.$$

We start by analyzing \mathbf{Q} . For every $i \in [n]$, the i 'th row of \mathbf{Q} is either $\mathbf{0}$, if $(\mathbf{Ax} + \mathbf{y})_i = 0$, or \mathbf{y} , if $(\mathbf{Ax} + \mathbf{y})_i = 1$.

Consider \mathbf{P} . We observe that the rows of \mathbf{AXA}^T are all in the image of \mathbf{A} , and so are the rows of $\mathbf{y}(\mathbf{Ax})^T$. It follows from linearity that all of \mathbf{P} 's rows are in the image of \mathbf{A} .

Thus, for every $i \in [n]$ such that $(\mathbf{Ax} + \mathbf{y})_i = 1$, the i 'th row of $\mathbf{P} + \mathbf{Q}$ is of the form $\mathbf{Ax}_i + \mathbf{y}$ for some \mathbf{x}_i . Overall, for at least $\text{UNSAT}(\mathcal{L}, \mathbf{x})$ of the rows, there are at least $\text{UNSAT}(\mathcal{L})$ non-zeros. \square

3.1 Decoding Tensorized Systems

So far, we were essentially recollecting well known facts. In this work, we would like to use tensoring to construct a PCP. In any PCP theorem, the challenging part is to infer some structure if we accept certain proof. Informally, this corresponds to establishing that if we accept a purported proof, we must be able to decode an assignment that (approximately) achieves the claims made by the purported proof. For our use of tensoring, we would like to state that if an assignment \mathbf{X} maximally satisfies the tensorized system, then necessarily $\mathbf{X} = \mathbf{x}^{\otimes 2}$ when \mathbf{x} is the diagonal of \mathbf{X} . We only focus on symmetric matrices \mathbf{X} . We can enforce this using a technique known as *folding* in the proof verification literature.

3.1.1 Folding

We can restrict our \mathbf{X} 's to symmetric matrices, that is, $X_{ij} = X_{ji}$ for all $i, j \in [n]$. This is done simply by not having separate entries for X_{ij} and X_{ji} , but only one entry, say X_{ij} . The term X_{ji} is replaced by X_{ij} .

3.1.2 Observations

The following observations will be crucial for our analysis:

Proposition 3.3. *For every symmetric matrix \mathbf{M} , if the Hamming weight of \mathbf{M} is k^2 and there is a subset of k rows each with a Hamming weight of k , then it must be that \mathbf{M} has only k non-zero columns. Moreover, the Hamming weight of each of these columns is k .*

Proof. Assume for the sake of contradiction that \mathbf{M} has $k+1$ non-zero columns. From the symmetry of \mathbf{M} , it follows that \mathbf{M} has $k+1$ non-zero rows. Since, \mathbf{M} already has k rows each with Hamming weight k , an additional non-zero row would imply that the Hamming weight of \mathbf{M} is larger than k^2 and this contradicts the hypothesis. Thus, it must be that \mathbf{M} has only k non-zero columns. The artifact that each of these columns has a Hamming weight of k follows from symmetry. \square

Proposition 3.4 (Uniqueness). *For every $\mathbf{x} \in GF_2^n$, every row of $\mathbf{x}^{\otimes 2}$ is either equal to \mathbf{x} or $\mathbf{0}$. Moreover, $\mathbf{x}^{\otimes 2}$ is symmetric.*

Sketch. Follows from the definition of $\mathbf{x}^{\otimes 2}$ and that we are working with GF_2 . \square

Proposition 3.5 (Converse of Uniqueness). *For every symmetric \mathbf{X} , if every row of \mathbf{X} is either \mathbf{x} or $\mathbf{0}$, then $\mathbf{X} = \mathbf{x}^{\otimes 2}$.*

Proof. Say \mathbf{X} has \mathbf{x} over rows R . By symmetry, \mathbf{X} also has them over R columns. Thus, \mathbf{X} has 1 in the (i, j) entry whenever \mathbf{x} has 1 at both i -th and j -th entries and zero elsewhere. And this is exactly the definition of $\mathbf{x}^{\otimes 2}$. \square

For any linear system $\mathcal{L} \equiv \mathbf{A}\mathbf{x} + \mathbf{y} = \mathbf{0}$, we define $d(\mathcal{L})$ as the $\min_{\mathbf{x} \neq \mathbf{0}} \mathbf{A}\mathbf{x}$.¹

Claim 3.6. *For every \mathbf{A} and $\mathbf{x}^{\otimes 2}$, the linear form $\mathbf{A}\mathbf{x}^{\otimes 2}\mathbf{A}^T$ has every row equal to either $\mathbf{A}\mathbf{x}$ or $\mathbf{0}$.*

Sketch. One may rewrite $\mathbf{A}\mathbf{x}^{\otimes 2}\mathbf{A}^T$ as $\mathbf{A}\mathbf{x} \cdot (\mathbf{A}\mathbf{x})^T$. Thus, the product is of the form $(\mathbf{A}\mathbf{x})^{\otimes 2}$ and now the claim follows from Proposition 3.4. \square

¹This is exactly the distance of the linear code having \mathbf{A} as its generator matrix.

Claim 3.7. *For every \mathbf{A} and symmetric \mathbf{X} , if every row of the linear form $\mathbf{A}\mathbf{X}\mathbf{A}^T$ is either $\mathbf{A}\mathbf{x}$ or $\mathbf{0}$ and $d(\mathcal{A}) > 0$, then $\mathbf{X} = \mathbf{x}^{\otimes 2}$.*

Proof. Since $d(\mathcal{A}) > 0$, for any \mathbf{v}_1 different from \mathbf{v}_2 , $\mathbf{A}\mathbf{v}_1 \neq \mathbf{A}\mathbf{v}_2$. Given this observation, for any \mathbf{z} in the image of \mathbf{A} , one can uniquely determine \mathbf{x}_0 such that $\mathbf{A}\mathbf{x}_0 = \mathbf{z}$. Now, the Claim follows via similar arguments as in Proposition 3.5. \square

Claim 3.8. *For every linear system $\mathcal{L}, \mathcal{L} \equiv \mathbf{A}\mathbf{x} + \mathbf{y} = \mathbf{0}$, and any symmetric \mathbf{X} , if every row of $\mathbf{P}, \mathbf{P} = \mathbf{A}\mathbf{X}\mathbf{A}^T + \mathbf{y} \cdot (\mathbf{A}\mathbf{x})^T$, is either $\mathbf{A}\mathbf{x}$ or $\mathbf{0}$ and $d(\mathcal{A}) > 0$, then $\mathbf{X} = \mathbf{x}^{\otimes 2}$.*

Proof. We start by observing that if every row of \mathbf{P} is either $\mathbf{A}\mathbf{x}$ or $\mathbf{0}$, then the sum of \mathbf{P} and $\mathbf{y}(\mathbf{A}\mathbf{x})^T$ also has each of its row either equal to $\mathbf{A}\mathbf{x}$ or $\mathbf{0}$. The last observation follows from the fact that we are working with GF_2 and it is easy to see that the set $\{\mathbf{A}\mathbf{x}, \mathbf{0}\}$ is closed under addition. Thus, $\mathbf{Z}, \mathbf{Z} = \mathbf{P} + \mathbf{y}(\mathbf{A}\mathbf{x})^T$, has every row either equal to $\mathbf{A}\mathbf{x}$ or $\mathbf{0}$. Recall that \mathbf{Z} is essentially the linear form $\mathbf{A}\mathbf{X}\mathbf{A}^T$, for some \mathbf{X} . Since $d(\mathcal{A}) > 0$, we may invoke Claim 3.7 to infer that \mathbf{X} must be equal to $\mathbf{x}^{\otimes 2}$. \square

Proposition 3.9. *For any non-zero \mathbf{v} and every \mathbf{X} , linear system $\mathcal{L}, \mathcal{L} = \mathbf{A}\mathbf{x} + \mathbf{y} = \mathbf{0}$, such that $UNSAT(\mathcal{L}) > 0$ and $d(\mathcal{A}) > 0$, it holds that $\mathbf{A}\mathbf{X}\mathbf{A}^T \neq \mathbf{y} \cdot (\mathbf{A}\mathbf{v})^T$.*

Proof. It follows from linearity that the columns of $\mathbf{A}\mathbf{X}\mathbf{A}^T$ are in the image of \mathbf{A} . Also from hypothesis, since \mathbf{v} is non-zero and $d(\mathcal{A}) > 0$, it follows that $\mathbf{A}\mathbf{v}$ is a non-zero vector. Thus, at least one column of $\mathbf{y} \cdot (\mathbf{A}\mathbf{v})^T$ is equal to \mathbf{y} . We now use Proposition 2.2 over that column to conclude that for any \mathbf{z} , $\mathbf{A}\mathbf{z}$ and \mathbf{y} differ on at least one of the coordinates. Thus, for any \mathbf{v} different from $\mathbf{0}$ and every \mathbf{X} and \mathcal{L} , it holds that $\mathbf{A}\mathbf{X}\mathbf{A}^T \neq \mathbf{y} \cdot (\mathbf{A}\mathbf{v})^T$, unless $d(\mathcal{A}) = 0$. \square

3.1.3 Main Lemma

We now show that if the prover provides us with any assignment \mathbf{X} other than one equal to $\mathbf{x}^{\otimes 2}$, then $UNSAT(\mathcal{L}^{\otimes 2}, \mathbf{X})$ is larger than d^2 .

Lemma 3.10 (Unique Tensor Lemma). *For any linear system $\mathcal{L}, \mathcal{L} \equiv \mathbf{A}\mathbf{x} + \mathbf{y} = \mathbf{0}$, with $UNSAT$ value d and every symmetric \mathbf{X} , if $UNSAT(\mathcal{L}^{\otimes 2}, \mathbf{X}) = d^2$, then $\mathbf{X} = \mathbf{x}^{\otimes 2}$ unless $d(\mathcal{A}) = 0$.*

Proof. We begin by recalling that for every \mathbf{X}

$$UNSAT(\mathcal{L}^{\otimes 2}, \mathbf{X}) = \text{wt}(\mathbf{A}\mathbf{X}\mathbf{A}^T + \mathbf{y}(\mathbf{A}\mathbf{x})^T + (\mathbf{A}\mathbf{x} + \mathbf{y}) \cdot \mathbf{y}^T)$$

As a matter of convention, we denote the first two terms of the RHS by \mathbf{P} and the latter by \mathbf{Q} . Let us denote the indices of the non-zero rows of \mathbf{Q} by \mathbf{S} . Recall Lemma 3.2 to infer that $\text{wt}(\mathbf{A}\mathbf{x} + \mathbf{y})$ must be d . Thus, $|\mathbf{S}| = d$. Also, the Hamming weight of $\mathbf{P} + \mathbf{Q}$ over the rows indexed by \mathbf{S} is at least d^2 . Let $\bar{\mathbf{S}}$ denote the set of indices of all non-zero rows of \mathbf{P} such that the corresponding row of \mathbf{Q} is equal to $\mathbf{0}$.

For the Hamming weight of $\mathbf{P} + \mathbf{Q}$ to be equal to d^2 , it must be that $\bar{\mathbf{S}} = \emptyset$. For otherwise, since the sets \mathbf{S} and $\bar{\mathbf{S}}$ are disjoint, the Hamming weight of $\mathbf{P} + \mathbf{Q}$ is at least the sum total of Hamming weight over rows with indices \mathbf{S} and $\bar{\mathbf{S}}$, which is greater than d^2 unless $\bar{\mathbf{S}} = \emptyset$.² Thus, if

²Because, the non-zero rows of \mathbf{Q} alone contribute to a weight of d^2 in $\mathbf{P} + \mathbf{Q}$ while $\bar{\mathbf{S}}$ deals with only the rows of \mathbf{Q} which are equal to $\mathbf{0}$.

the Hamming weight of $\mathbf{P} + \mathbf{Q}$ is d^2 , then the weight of \mathbf{P} cannot be outside the non-zero rows of \mathbf{Q} .

We now show that if each of the rows in \mathbf{P} indexed by \mathbf{S} ain't the same, then the Hamming weight of $\mathbf{P} + \mathbf{Q}$ is strictly larger than d^2 . In what follows we shall deal with absolute Hamming weight rather than its normalized definition. Say, there are two rows $\mathbf{A}\hat{\mathbf{x}}$ and $\mathbf{A}\tilde{\mathbf{x}}$ different from one another. Since $\mathbf{A}\hat{\mathbf{x}}$ and $\mathbf{A}\tilde{\mathbf{x}}$ are different from one another, they must differ on at least one coordinate. Also, since the indices of these rows are in \mathbf{S} , they are both added to \mathbf{y} . We now observe via Claim 2.1 that $\mathbf{A}\hat{\mathbf{x}} + \mathbf{y}$ and $\mathbf{A}\tilde{\mathbf{x}} + \mathbf{y}$ must also differ on at least one coordinate. Moreover, by Proposition 2.2 any $\mathbf{A}\mathbf{x}_0 + \mathbf{y}$ must have at least dm non-zero coordinates, where m is the dimension of \mathbf{y} . Hence, there are at least $dm + 1$ non-zero coordinates over both $\mathbf{A}\hat{\mathbf{x}} + \mathbf{y}$ and $\mathbf{A}\tilde{\mathbf{x}} + \mathbf{y}$. Therefore, $\mathbf{P} + \mathbf{Q}$ must have at least $(dm + 1)$ non-zero columns. Since $\mathbf{P} + \mathbf{Q}$ is symmetric, by the contrapositive of Proposition 3.3, if $\mathbf{P} + \mathbf{Q}$ has $(dm + 1)$ non-zero columns, then either the Hamming weight of $\mathbf{P} + \mathbf{Q}$ is not $d^2 \cdot m^2$ or $\mathbf{P} + \mathbf{Q}$ does not have $d \cdot m$ rows each with Hamming weight $d \cdot m$. Since, the rows of $\mathbf{P} + \mathbf{Q}$ indexed by \mathbf{S} are in fact $d \cdot m$ rows each with a Hamming weight of $d \cdot m$, it must be that the Hamming weight of $\mathbf{P} + \mathbf{Q}$ is not $d^2 \cdot m^2$. Hence, if \mathbf{P} does not have the same row over the rows indexed by \mathbf{S} , the Hamming weight of $\mathbf{P} + \mathbf{Q}$ is strictly greater than $d^2 \cdot m^2$. In other words, if the (normalized) Hamming weight of $\mathbf{P} + \mathbf{Q}$ is d^2 , then \mathbf{P} has the same row across every row indexed by \mathbf{S} .

In short, we have established that \mathbf{P} must have the same row across all indices in \mathbf{S} and every row outside \mathbf{S} is $\mathbf{0}$. *w.l.o.g.*, assume that one of the following is true: (1) \mathbf{P} has $\mathbf{A}\mathbf{x}$ as its row across all rows indexed by \mathbf{S} . (2) \mathbf{P} has a row different from $\mathbf{A}\mathbf{x}$.

Case 1: Since every row of \mathbf{P} is either $\mathbf{A}\mathbf{x}$ or $\mathbf{0}$ and $d(\mathcal{A}) > 0$, by Claim 3.8, $\mathbf{X} = \mathbf{x}^{\otimes 2}$.

Case 2: Assume for the sake of contradiction that \mathbf{P} has a row $\mathbf{A}\tilde{\mathbf{x}}$ different from $\mathbf{A}\mathbf{x}$. Also since $\mathbf{A}\tilde{\mathbf{x}}$ different from $\mathbf{A}\mathbf{x}$, we may deduce that \mathbf{x} is different from $\tilde{\mathbf{x}}$. Now, $\mathbf{P} + \mathbf{Q}$ can be written as.

$$\begin{aligned}\mathbf{P} + \mathbf{Q} &= (\mathbf{A}\mathbf{x} + \mathbf{y}) \cdot (\mathbf{A}\tilde{\mathbf{x}})^T + (\mathbf{A}\mathbf{x} + \mathbf{y}) \cdot \mathbf{y}^T \\ &= \mathbf{A} \cdot \mathbf{x} \cdot \tilde{\mathbf{x}}^T \cdot \mathbf{A}^T + \mathbf{y} \cdot (\mathbf{A}\tilde{\mathbf{x}})^T + (\mathbf{A}\mathbf{x} + \mathbf{y}) \cdot \mathbf{y}^T\end{aligned}$$

But, the above expression must be exactly equal to

$$\mathbf{A}\mathbf{X}\mathbf{A}^T + \mathbf{y}(\mathbf{A}\mathbf{x})^T + (\mathbf{A}\mathbf{x} + \mathbf{y}) \cdot \mathbf{y}^T$$

Therefore,

$$\begin{aligned}\mathbf{A}\mathbf{X}\mathbf{A}^T + \mathbf{y}(\mathbf{A}\mathbf{x})^T &= \mathbf{A} \cdot \mathbf{x} \cdot \tilde{\mathbf{x}}^T \cdot \mathbf{A}^T + \mathbf{y} \cdot (\mathbf{A}\tilde{\mathbf{x}})^T \\ \mathbf{A} \cdot (\mathbf{X} + \mathbf{x} \cdot \tilde{\mathbf{x}}^T) \cdot \mathbf{A}^T &= \mathbf{y} \cdot (\mathbf{A} \cdot (\mathbf{x} + \tilde{\mathbf{x}}))^T\end{aligned}$$

Since, \mathbf{x} is different from $\tilde{\mathbf{x}}$, it follows that $\mathbf{x} + \tilde{\mathbf{x}}$ is a non-zero vector. Therefore, we can invoke Proposition 3.9 to conclude that there is no $(\mathbf{X} + \mathbf{x} \cdot \tilde{\mathbf{x}}^T)$ for which the above holds. Hence, a contradiction. Therefore, \mathbf{P} cannot have only $\mathbf{A}\tilde{\mathbf{x}}$, for any $\mathbf{A}\tilde{\mathbf{x}}$ different from $\mathbf{A}\mathbf{x}$, in the rows indexed by \mathbf{S} .

This concludes the proof of the Lemma as we have established that if $\text{UNSAT}(\mathcal{L}^{\otimes 2}, \mathbf{X}) = d^2$, then $\mathbf{X} = \mathbf{x}^{\otimes 2}$ unless $d(\mathcal{A}) = 0$. \square

4 Outline of the Construction

We start with a construction of a binary linear PCP based on Håstad’s MAX-3LIN construction [Hås01] (Section 5). Importantly, this PCP is an “assignment tester”, a certain strengthening of PCP that is needed later. We preserve this strong PCP property throughout the operations we perform on this PCP. We perform tensoring to amplify the completeness of the basic construction in Section 6 (getting a “longer long code”). Tensoring increases the number of satisfied equations, and so decreases the completeness error, but increases the soundness error. Hence, in Section 6, we follow tensoring with a randomness-efficient sequential repetition that preserves linearity and the low completeness.

This inner construction is then composed with an outer construction with perfect completeness in Section 7. The outer construction is not linear. However, since the inner construction is based on the long code, the composed construction can still be linear, just like Håstad constructs linear verifiers out of non-linear verifiers in [Hås01].

In the second phase of our construction (ref. Section 8), we leverage the low completeness error to boost the soundness to $1/\mathcal{O}(\log N)^\beta$ for some $\beta > 0$. Specifically, we use techniques from [MR10] to amplify the soundness while preserving the linearity of the verifier. This yields a family of Label-cover instances whose constraints are linear projections. Note that the [MR10] transformation produces a PCP with the projection property (two queries), even when starting with a PCP with many queries.

The applications for MAX-3LIN and MAX-3SAT are presented in Section 9. The linearity of our construction enables us to use Hadamard codes rather than Long codes. To the best of our knowledge, Khot [Kho01] was the first to construct Hadamard based PCPs.

The construction described above is our “simple construction”, and it only yields logarithmically-small completeness error. We proceed to a more complicated construction that yields polynomially-small completeness error. This is needed for our hardness results for MIN-3LIN-DELETION and NEAREST-CODEWORD-PROBLEM, and also to achieve optimal results for MAX-3LIN and MAX-3SAT assuming Conjecture 1.2. For the construction of PCPs with polynomially small completeness error we use an iterative construction, performing tensoring and query reduction. This is done in Section 10. The outline of our construction is also presented in Figure 1.

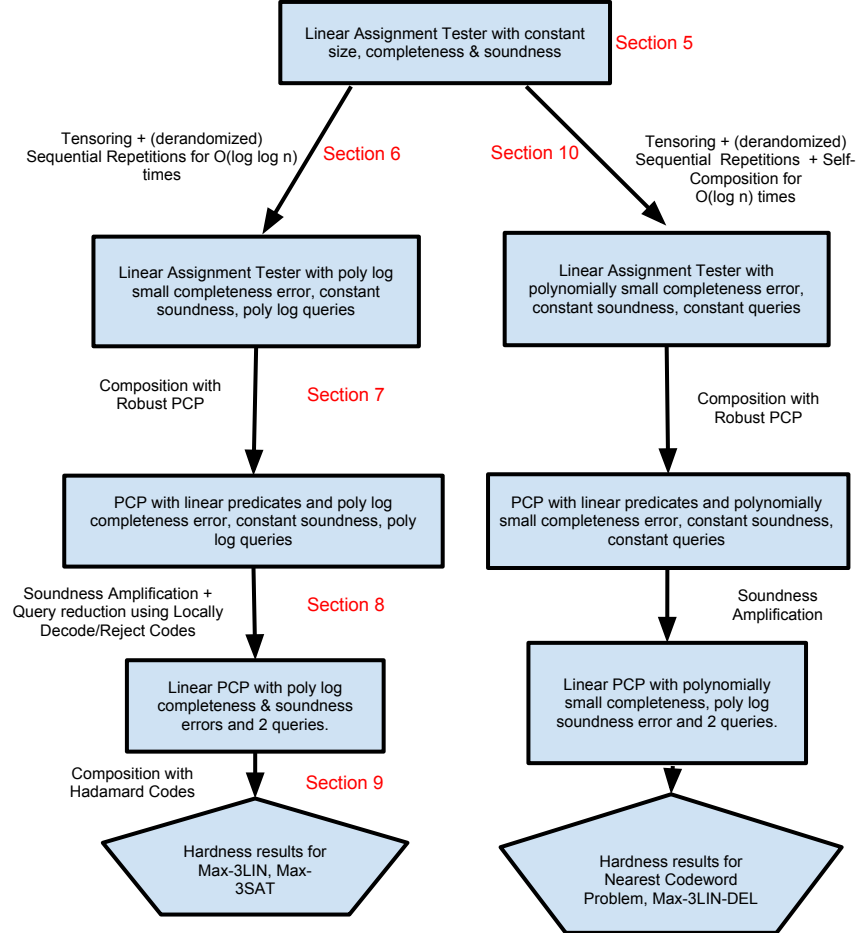


Figure 1: Layout of our Construction.

5 Assignment Testers

In this section, we construct a PCP verifier with a linear tests. In what follows, we work with assignment testers. The notion of assignment testers was pioneered by [DR06, BSGH⁺06, Sze99]. They are (also) sometimes referred to as PCP of Proximity. A related notion was recently explored as Locally Decode and Reject Codes (LDRC) [MR10] (ref. Definition 6) and as dPCP by [DH09]. Informally, an assignment tester not only needs to accept a correct proof almost always, but reject a purported proof which is far from any valid proof. Note that this is a stronger requirement to that of a PCP where one aspires to verify only if there is a valid proof or not. We restrict ourselves to *linear* assignment testers. These are assignment testers where the verifier's predicate is a linear function over its variables.

Definition 4 (Assignment Tester). *An $(1 - \rho, \delta)$ -Assignment Tester is a reduction whose input is a Boolean constraint ψ over a set of Boolean variables X . The output of the reduction is a system of constraints Ψ over variables X and auxiliary variables Y such that for every assignment $\pi : X \rightarrow \{0, 1\}$,*

- **Completeness:** *If π satisfies ψ then there exists an assignment $\tilde{\pi} : Y \rightarrow \{0, 1\}$ such that $\pi \cup \tilde{\pi}$ satisfies at least $1 - \rho$ fraction of the constraints in Ψ .*
- **Soundness:** *For every $\delta' < \delta$, if π is δ' -far from a satisfying assignment for ψ , then for every assignment $\tilde{\pi} : Y \rightarrow \{0, 1\}$, at least $\Omega(\delta')$ fraction of the constraints in Ψ reject $\pi \cup \tilde{\pi}$.*

From now on, it will be useful to use the set $\{+1, -1\} = \{\pm 1\}$ instead of $\{0, 1\}$. We use the following map $b \mapsto (-1)^b$ (that is, $0 \mapsto 1$ and $1 \mapsto -1$). Also note that this maps the addition in $GF(2)$ into multiplication operation over \mathbb{R} .

Standard Definitions. We identify the *long code* of $\mathbf{x} \in \{\pm 1\}^s$ by $\text{LC}(\mathbf{x}) = \{f(\mathbf{x}) \mid f : \{\pm 1\}^s \rightarrow \{\pm 1\}\}$. Informally, we evaluate \mathbf{x} on every Boolean function on s bits. Notice that every Boolean function on s bits may be represented by its truth table. In other words, by specifying its evaluation on all the 2^s inputs. Alternatively, any string of length 2^s may be interpreted as a Boolean function on s bits. We denote 2^s by n . Since, there are 2^n Boolean functions on s bits, $\text{LC}(\mathbf{x})$ is a string on length 2^n . We use the letters f, g to denote Boolean functions. It is easy to check that given a table A , $A \equiv \text{LC}(\mathbf{x} : \{\pm 1\}^s \rightarrow \pm 1, A(f)A(g) = A(fg)$. That is, A is closed under multiplication.

For $\alpha \in [n]$, define

$$\chi_\alpha : \{\pm 1\}^n \rightarrow \{\pm 1\}, \chi_\alpha(f) \triangleq \prod_{i \in \alpha} f(i)$$

It is easy to check that the characters $\{\chi_\alpha\}_{\alpha \subseteq [n]}$ form an orthonormal basis for the space of functions $\{A : \{\pm 1\}^n \rightarrow \mathbb{R}\}$, where inner product is defined by $\langle A, B \rangle = \mathbb{E}_f[A(f)B(f)] = 2^{-n} \sum_f A(f)B(f)$. It follows that any function $A : \{\pm 1\}^n \rightarrow \{\pm 1\}$ can be written as $A = \sum_\alpha \hat{A}_\alpha \cdot \chi_\alpha$, where $\hat{A}_\alpha = \langle A, \chi_\alpha \rangle$.

The Long Code Test. Let $A : \{\pm 1\}^n \rightarrow \{\pm 1\}$. We intend to test if $A : \{\pm 1\}^n \rightarrow \{\pm 1\}$ is in fact the legal encoding of a value $i \in [s]$. In other words, if $A(f) = f(i)$ for all $f \in [2^n]$.

Fix a parameter $\rho \in [0, 1]$. The test picks two uniformly random vectors $f, g \in \{\pm 1\}^n$ and then $h \in \{\pm 1\}^n$ according to the following distribution: for every coordinate $i \in [n]$, with probability $1 - \rho$ we choose $h_i = 1$ and $h_i = -1$ otherwise. It is useful to imagine h as a noise vector. The test accepts iff $A(f)A(g) = A(fgh)$.

Lemma 5.1 (Tester Lemma). *For every $\delta > 0$, table A which is δ -far from any valid long code encoding is rejected with probability at least $\Omega(\delta)$.*

Proof. Assume that A passes the test with probability $1 - \delta$, then $\mathbb{E} [A(f)A(g)A(fgh)] = 1 - 2 \cdot \delta$. Replacing A by its Fourier expansion, we have

$$\begin{aligned} 1 - 2 \cdot \delta &= \mathbb{E}_{f,g,h} \left[\left(\sum_{\alpha} \hat{A}_{\alpha} \chi_{\alpha}(f) \right) \cdot \left(\sum_{\beta} \hat{A}_{\beta} \chi_{\beta}(g) \right) \cdot \left(\sum_{\gamma} \hat{A}_{\gamma} \chi_{\gamma}(fgh) \right) \right] \\ &= \mathbb{E}_{f,g,h} \left[\sum_{\alpha,\beta,\gamma} \hat{A}_{\alpha} \hat{A}_{\beta} \hat{A}_{\gamma} \chi_{\alpha}(f) \chi_{\beta}(g) \chi_{\gamma}(f) \chi_{\gamma}(g) \chi_{\gamma}(h) \right] \\ &= \sum_{\alpha,\beta,\gamma} \hat{A}_{\alpha} \hat{A}_{\beta} \hat{A}_{\gamma} \mathbb{E}_{f,g,h} [\chi_{\alpha}(f) \chi_{\beta}(g) \chi_{\gamma}(f) \chi_{\gamma}(g) \chi_{\gamma}(h)] \end{aligned}$$

Since the fourier basis are orthonormal, the expectation is 0 unless $\alpha = \beta = \gamma$. Therefore,

$$1 - 2\delta = \sum_{\alpha} \hat{A}_{\alpha}^3 \mathbb{E}_h [\chi_{\alpha}(h)]$$

Because $\mathbb{E}_h [\chi_{\alpha}(h)] = \mathbb{E}_h [\prod_{w \in \alpha} h_w] = (1 - 2\rho)^{|\alpha|}$, as each coordinate of h is chosen independently. Hence,

$$\begin{aligned} 1 - 2\delta &= \sum_{\alpha} \hat{A}_{\alpha}^3 (1 - 2\rho)^{|\alpha|} \\ &\leq \sum_{\alpha} \hat{A}_{\alpha}^2 \cdot 1 \cdot (1 - 2\rho)^{|\alpha|} \quad \because \hat{A}_{\alpha} \leq 1 \\ &\leq \sum_{|\alpha|=1} \hat{A}_{\alpha}^2 (1 - 2\rho) + \sum_{|\alpha|>1} \hat{A}_{\alpha}^2 (1 - 2\rho)^{|\alpha|}, \quad \hat{A}_{\emptyset} = 0 \text{ due to folding} \end{aligned}$$

Since A is assumed to have been folded, $\hat{A}_{\alpha} = 0$ if $|\alpha|$ is even. Hence, $(1 - 2\rho)^{|\alpha|} \leq (1 - 2\rho)^3$. Therefore, the above expression can be written as follows.

$$\begin{aligned} 1 - 2\delta &\leq \sum_{|\alpha|=1} \hat{A}_{\alpha}^2 (1 - 2\rho) + \sum_{|\alpha|>1} \hat{A}_{\alpha}^2 (1 - 2\rho)^3 \\ &\leq (1 - \Phi)(1 - 2\rho) + \Phi(1 - 2\rho)^3 \quad \text{Let } \sum_{\alpha, |\alpha|>1} \hat{A}_{\alpha}^2 = \Phi \\ 1 - 2\delta &\leq (1 - 2\rho)(1 - \Phi + \Phi(1 - 2\rho)^2) \\ 1 - 2\delta/1 - 2\rho &\leq (1 - \Phi(1 - (1 - 2\rho)^2)) \\ 1 - 2(\delta - \rho) &\leq 1 - \Phi(1 - (1 - 2\rho)^2) \\ \Phi &\leq 2(\delta - \rho)/(4\rho - 4\rho^2) \end{aligned}$$

Since, ρ is a fixed parameter, we have $\Phi = \mathcal{O}(\delta)$. At which point, we recall a result of Friedgut, Kalai and Naor [FKN02] which (informally) states that if the fourier coefficients are concentrated on the lowest two levels, then the function is close to a constant function or a function that is determined by a single coordinate.

Theorem 5.2 (Theorem B.4, Page 41 in [Din07]). *There is a global constant C' (independent of n) such that the following holds. Let $\Phi > 0$ and let $A : \{1, -1\}^n \rightarrow \{1, -1\}$ be a Boolean function for which $\sum_{\alpha, |\alpha| > 1} |\hat{A}_\alpha|^2 < \Phi$. Then either $|\hat{A}_\emptyset|^2 \geq 1 - C'\Phi$, or for some $i \in [n]$, $|\hat{A}_{\{i\}}|^2 = 1 - C'\Phi$.*

Therefore, by folding, there must be some i such that $\psi(i) = -1$ and χ_i is $\mathcal{O}(\Phi)$. Theorem 5.2 also allows $-\chi_i$ to be $\mathcal{O}(\Phi)$ -close to A , but this causes the test to fail with probability close to $1/4$, which is certainly $\Omega(\delta)$. Hence, we showed that unless A is δ -close to some χ_i for the value of i that satisfies ψ , at least $\Omega(\delta)$ of the tests must be rejected. \square

We are now ready to present the assignment tester needed for our construction. Let ψ be a Boolean constraint over Boolean variables $x_1, x_2 \dots x_s$. We describe an algorithm whose input is ψ and whose output will be a system of linear equations satisfying the requirements of Definition 4. The tester seeks as input a satisfying assignment σ to ψ and the $\text{LC}(\sigma)$, the long code of σ . One may imagine $\text{LC}(\sigma)$ as the set of auxiliary variables used by the tester.

Also, as done in [BGS98], we fold the long code tables over true and the respective constraint ψ . This means that whenever the test needs to read $A[f]$, it reads $A[\psi \wedge f]$ instead. In addition, we fold over true which means for every pair f and $-f$, we let A specify only one and access the other via the identity $A[f] = -A[-f]$. In short, we assume that $A[f] = A[f \wedge \psi]$ and $A[f] = -A[-f]$ for all f . It is well known that after folding $\hat{A}_\alpha = 0$ whenever $|\alpha|$ is even or there exists an i in α for which $\psi(i) = 1$ (recall that 1 corresponds to false). An interested reader may refer to Lemma 22.27 on page 482 of [AB09]. We elaborate on the latter. Set $f' = f + e_i$, where i is an index such that $\psi(i) = 1$. Now, $\chi_\alpha(f) = -\chi_\alpha(f')$ but $A[f] = A[f']$. We now describe the two kinds of constraints we place over these variables. Say, the tester gets two tables σ and A as input.

1. **Long Code Constraints:** The first set of constraints aims at testing if A is indeed a valid long code encoding. It includes 3 variable constraints derived from the *long code* test specified earlier. Specifically, we shall fix ρ to be a non-zero constant and have one constraint per each coin toss of the long code test.
2. **Consistency Constraints:** The second set of constraints include the following: For each choice of $i \in [s]$ and f place a constraint that is satisfied iff $\sigma(x_i) = A(f) \oplus A(f \oplus e_i)$ ³. For the ease of argument, assume that there are an equal number of constraints of each type⁴.

Lemma 5.3 (Assignment Tester). *For any $\rho, \delta > 0$, the constraint system constructed above is a $(1 - \rho, \delta)$ -assignment tester for any Boolean function $\psi : [s] \rightarrow \{0, 1\}$. Moreover, the assignment tester is a linear function on its input.*

Proof. Linearity of the constraints follows from construction. We now analyze the parameters of the tester below parameter by parameter.

- **Completeness:** We set the probability of generating the noise vector h in the long code test to $1 - \rho$. Now, for any good assignment that satisfies the constraint ψ , the second set of constraints are always satisfied. The only case where we reject a good proof is during the long code test and this happens with probability at most ρ .
- **Soundness:** Say, the tester gets σ and π as input and σ is δ -far from a satisfying assignment of ψ , we shall show that the assignment tester rejects with probability at least $\Omega(\delta)$. *w.l.o.g.*,

³ e_i is the vector of dimension 2^s with a -1 in i -th index and 1 elsewhere.

⁴This can be easily achieved by placing multiple copies of the constraint of each type with appropriate multiplicity.

we consider the following cases. (1) π is δ -far from any long code encoding. (2) π is not δ -far from a valid long code encoding.

In the former, it follows from Lemma 5.1 that if π is δ -far from a valid long code table is rejected with probability greater than $\Omega(\delta)$ and by our construction these constraints constitute at least $\Omega(\delta)$ -fraction of the total constraints. Thus, we reject any π that is δ -far from a valid long code encoding with probability at least $\Omega(\delta)$.

In the latter, since π was folded across ψ , we may assume that π encodes a satisfying assignment to ψ . Say π encoded a satisfying assignment σ' . Since, σ is δ -far from a satisfying assignment, it must be that σ is δ -far from σ' . Hence, $\Pr_i[\sigma(x_i) \neq \sigma'(x_i)] \geq \delta$. Since, π is not δ -far from a valid long code encoding, we also have that

$$\begin{aligned} \Pr_{f \in [n]} [\pi(f) \oplus \pi(f + e_i) = f(\sigma') \oplus (f \oplus e_i)(\sigma')] &\geq \Pr[\pi(f) = f(\sigma') \text{ and } \pi(f \oplus e_i) = (f \oplus e_i)(\sigma')] \\ &\geq 1 - 2\delta \end{aligned}$$

This fails whenever $\sigma(x_i) \neq \sigma'(x_i)$, then $f(\sigma') \oplus (f \oplus e_i)(\sigma') = \sigma'(x_i) \neq \sigma(x_i)$. And this is at least $\delta(1 - 2\delta) = \Omega(\delta)$.

Hence, in either case we would have rejected $\sigma \circ \pi$ with probability at least $\Omega(\delta)$. Because, there are an equal number of constraints of each type, we reject an invalid proof π with probability at least $\Omega(\delta)$.

- *Query Size:* The tester picks at random a long code constraint or a consistency constraint and verifies it. In either case, it makes 3 queries.
- *Randomness:* The tester uses $2 \cdot s$ bits of randomness. □

We denote the size of a linear tester by the n and it refers to the quantity $q \cdot m$, where q is the number of queries made by the tester and m is the total number of equations in the tester. Linear assignment testers cannot have perfect completeness. For otherwise, one may use gaussian elimination to figure out the if there is an assignment that satisfies all the linear constraints and reject if there isn't one. Hence, any linear tester is forced to have a non-zero completeness error. Also, the soundness is always less than $1/2$ as a random assignment satisfies at least $1/2$ of the equations in \mathcal{L} in an expected sense (follows from the fact that the equations are over $GF(2)$). As highlighted earlier, having low completeness error becomes handy in several scenarios. We now highlight on how we intend to use the tools we developed in the previous section to achieve completeness amplification.

6 Completeness Amplification

In this section, we employ tensoring to amplify the completeness of the assignment tester we have built in Section 5. In what follows, we start with notations and a few observations that will be useful to us in amplifying the completeness of the assignment tester.

Definition 5. Let \mathcal{T} denote the assignment tester obtained via Lemma 5.3 and $\psi : [s] \rightarrow \{0, 1\}$ denote the Boolean predicate being tested. For any proof Ψ provided to \mathcal{T} , we denote by Π_ψ the assignment induced by Π on the variables of ψ . Also, we denote by $\psi(\Pi_\psi)$ the evaluation of ψ on Π_ψ .

Observation 6.1. *For any given proof Π to assignment tester \mathcal{T} to verify the satisfiability of ψ , $\Pi_\psi^{\otimes 2}$ is exactly same as Π_ψ . More importantly, the distance of Π_ψ to a satisfying assignment of ψ remains unaffected by tensoring. Moreover, the converse also holds. That is, for any purported proof Π to a tensored system and its diagonal entries, denoted by π , π_ψ is same as Π_ψ .*

Proof Sketch. Since, we are working with $GF(2)$, it follows from the definition of tensoring that the diagonal entries $\{x_{ii}\}$ of $\Pi^{\otimes 2}$ preserve Π . Thus, Π_ψ and $\Pi_\psi^{\otimes 2}$ are exactly the same. The converse follows from the fact that we extract (decode) the assignment π from Π by reading its diagonal entries. \square

In fact, both directions of the aforementioned observation can be extended to any $\Pi^{\otimes k}$, $\Pi^{\otimes j}$, $0 < j \leq k$. We are now ready to amplify the completeness of the basic assignment tester we have constructed in Lemma 5.3.

Lemma 6.2 (Tensor Assignment Tester). *For every $\rho, \delta > 0$, there is a $(1 - \rho^2, \delta^2)$ -assignment tester for any Boolean function $\psi : [s] \rightarrow \{0, 1\}$. Moreover, the predicate of the assignment tester is linear.*

Proof. We invoke Lemma 5.3 to construct an $(1 - \rho, \delta)$ -assignment tester \mathcal{T} . The tests performed by our new $(1 - \rho^2, \delta^2)$ -assignment tester, denoted by $\mathcal{T}^{\otimes 2}$, will be tensored tests of \mathcal{T} . In other words, if the tests of \mathcal{T} are \mathcal{L} , then the tests of $\mathcal{T}^{\otimes 2}$ are linear equations in $\mathcal{L}^{\otimes 2}$ (ref. Definition 2). Let Π denote proof required by \mathcal{T} . The proof to $(1 - \rho^2, \delta^2)$ -assignment tester $\mathcal{T}^{\otimes 2}$ will be $\Pi^{\otimes 2}$. Thus, the new proof format will be $(\sigma \circ \pi)^{\otimes 2}$ and the variables of ψ will be $\{\sigma_{ii}\}$ and the rest will be auxiliary variables of $\mathcal{T}^{\otimes 2}$. We shall now analyze the parameters of $\mathcal{T}^{\otimes 2}$.

- *Completeness:* In the good case, if there is a proof $\sigma \circ \pi$ which \mathcal{T} accepts with probability at least $1 - \rho$. We invoke Proposition 3.1 to conclude that the probability that $\mathcal{T}^{\otimes 2}$ accepts $(\sigma \circ \pi)^{\otimes 2}$ is at least $1 - \text{UNSAT}(\mathcal{L}, \sigma \circ \pi)^2$, which is $1 - \rho^2$.
- *Soundness:* We shall establish that for any purported proof Π to $\mathcal{T}^{\otimes 2}$, if Π_ψ is δ^2 -far from a satisfying assignment to ψ , then $\mathcal{T}^{\otimes 2}$ rejects with probability at least $\Omega(\delta^2)$. In order to establish this, we deal with the contrapositive of the claim. Say, if Π is accepted with probability $1 - \delta^2$, we shall show that Π_ψ is not $\Omega(\delta^2)$ -far to a satisfying assignment of ψ .

Say, Π is accepted with probability at least $1 - \delta'$, we can use Lemma 3.2 to infer that the diagonal vector of Π gives us a proof which \mathcal{T} accepts with probability at least $\alpha = 1 - \lfloor \frac{\delta'}{\delta} \rfloor$, where $\delta' < 1/16$. For the parameters that we have chosen, if $\delta' = \Theta(\delta^2)$ then $\alpha \simeq 1 - \Omega(\delta)$. In other words, if Π has an acceptance probability of $1 - \delta^2$, then we can decode a proof π , which \mathcal{T} would accept with probability $1 - \Theta(\delta)$. And from Lemma 5.3, we know that if \mathcal{T} accepts π with probability $1 - \Theta(\delta)$, then π_ψ isn't $\Omega(\delta)$ -far from a satisfying assignment of ψ .

But, from Observation 6.1 we can infer that Π_ψ is exactly same as π_ψ . Thus, if π_ψ is not δ -far from a satisfying assignment of ψ , so is Π_ψ . Also, by hypothesis, $\mathcal{T}^{\otimes 2}$ accepts Π with probability $1 - \Theta(\delta^2)$. Hence, we can conclude that for every Π , if Π is accepted with probability at least $1 - \delta^2$, then Π_ψ is not $\Omega(\delta)$ -far from a satisfying assignment. Since $\delta^2 < \delta$, it follows that if Π_ψ is δ -far from a string, it is δ^2 -far. Therefore, the soundness of $\mathcal{T}^{\otimes 2}$ holds.

- *Query size:* $\mathcal{T}^{\otimes 2}$ needs to make as many queries as the number of variables in an equation of $\mathcal{L}^{\otimes 2}$. And this happens to be q^2 , where q is query size of \mathcal{T} .
- *Randomness:* The randomness used by $\mathcal{T}^{\otimes 2}$ is $\log \mathcal{L}^{\otimes 2}$, which is $2 \cdot \log \mathcal{L}$. In general, if we blow up the randomness by a factor of k whenever we tensor an assignment tester k times. \square

Notice that the above transformation preserves the linearity of the tester. The downside of tensoring is that while it enhances the completeness, it also destroys the soundness. Ideally, we would like to improve the acceptance probability of the verifier in the good case and leave the acceptance probability in the bad case unperturbed. Thus to improve the soundness of the tester after the tensoring operation, we invariably resort to soundness amplification.

6.1 Soundness Amplification Perserving Linearity

Soundness amplification of a PCP can be achieved via some kind of repetition, either sequentially or in parallel. Repetition of a PCP involves repeating the tests with independent randomness. For instance, performing sequential repetition once would involve repeating the assignment tester on two independently drawn tests and accepting them iff both are satisfied. This transforms an $(1-\rho, \delta)$ -assignment tester into an $(1-2\cdot\rho, 2\cdot\delta)$ -assignment tester. In general, performing sequential repetition for ϑ times on a $(1-\rho, \delta)$ -tester leaves us with a $(1-\vartheta\cdot\rho, \vartheta\cdot\delta)$ -tester. However, this operation does not preserve linearity.

Thus, we resort to a different kind of soundness amplification. We pick $\mathcal{O}(1)$ tests of the assignment tester and then, add all possible linear combinations of these tests to create a new assignment tester. The idea is that even if one of the $\mathcal{O}(1)$ equations are not satisfied by some assignment, half of the linear combinations of these equations will also not be satisfied by the assignment. This also happens to preserve the linearity. However, to keep the size of the output assignment tester instance small, we pseudo-randomly generate only $\mathcal{O}(n)$ of the $n^{\mathcal{O}(1)}$ possible ways to pick $\mathcal{O}(1)$ tests from n tests.

Given a $(1-\rho^2, \delta^2)$ -assignment tester as an input, we wish to produce an $(1-\rho^2, \delta)$ -assignment tester as the output. We achieve this via random walks on expanders. We aim to achieve the following by taking a random walk on an expander: the probability of visiting a subset containing a certain fraction (δ^2 in our case) of the vertices is at least $2\cdot\delta$. And since, at least half of the linear combinations having an unsatisfied equation also remain unsatisfied, we would end up with our cherished objective of amplifying the soundness to δ .

We begin by defining a *walk of length l* in a graph $G = (V, E)$ is a sequence v_0, \dots, v_l of vertices of G , where for $1 \leq i \leq l$, $v_{i-1}v_i$ is an edge in G . By a simple calculation, the total number of walks of length l in any d -regular graph is exactly $|V| \cdot d^l$. Suppose there is a subset C of V , we wish to bound the number of walks that do not contain a vertex from C . If G happens to be disconnected, it may happen that a constant fraction of them avoid C . However, Ajtai, Komlós and Szemrédi [AKS87] showed that if all the eigenvalues of G , with the exception of the largest, are small, then there are far fewer of walks avoiding C . We now state their result.

Theorem 6.3. *Let $G = (V, E)$ be a d -regular graph on n vertices, and suppose that each of its eigenvalues but the first one is at most λ . Let C be a set of cn vertices of G . Then, for every l , the number of walks of length l in G that avoid C does not exceed $(1-c) \cdot n \cdot ((1-c) \cdot d + c \cdot \lambda)^l$.*

Now, a *randomly chosen walk* of length l in G chosen according to a uniform distribution among all walks of that length. If G is regular, then such a walk can be chosen by choosing its starting point v_0 uniformly at random and then picking the next vertex of the walk among the d neighbours of v_0 uniformly at random. We are now ready to state a Corollary of Lemma 6.3 which will be very useful for us.

Corollary 6.3.1. *Let $G = (V, E)$, d, n, λ, C and c be as in Theorem 6.3 and suppose*

$$(1-c) \cdot d + c \cdot \lambda \leq \frac{d}{\sqrt{2}}$$

Then, for every l , the probability that a randomly chosen walk of length l in G avoids C is at most $2^{-l/2}$.

Lemma 6.4 (Soundness Amplification). *There is universal constant ϑ , such that every $(1 - \rho^2, \delta^2)$ -assignment tester for a Boolean predicate ψ over variable set σ can be amplified to a $(1 - \vartheta \cdot \rho^2, \delta)$ -assignment tester. Moreover, the new tester still has a linear predicate.*

Proof. Let us denote the $(1 - \vartheta \cdot \rho^2, \delta^2)$ -assignment tester by V and the tests performed by V with \mathcal{L} . Our approach will be to design a new assignment tester V' whose tests are the sum of ϑ tests of the old verifier V . We construct a graph G in which the vertices are the tests of V . Let ϕ_i denote the equation (constraint) at vertex v_i . We then pick a (test) vertex v_1 uniformly at random. We then take a random walk of length ϑ starting at v_1 . The new tests will be the sum of tests at each of these vertices. Thus, $\mathcal{L}' = \{\phi_1 \oplus \phi_2 \dots \oplus \phi_\vartheta : \phi_i \in \mathcal{L}\}$. We fix ϑ to be the length of random that one must take on an expander so that the probability we hit a δ^2 fraction of the tests is at least $2 \cdot \delta$. It follows from Corollary 6.3.1 that for an appropriate choice of expander and some constant ϑ less than 5 it holds that the probability we hit a δ^2 fraction of the vertices is at least $2 \cdot \delta$. Note that even though the tests of V' are different from V , both of them expect the same proof as in Lemma 5.3. We now establish that V' is in fact a $(1 - \vartheta \cdot \rho^2, \delta)$ -assignment tester.

- *Completeness:* In the good case, V reject a valid proof with probability at least ρ^2 . Now by basic probability, the probability that V' accepts a valid proof it is at most $(1 - \rho^2)^\vartheta$, which is bounded from below by $(1 - \vartheta \cdot \rho^2)$.
- *Soundness:* Say, we are given an assignment tester V that rejects any proof Π if Π_ψ (projection of Π on the variable set σ) is δ -far from a satisfying assignment of ψ , it is rejected with probability at least $\Omega(\delta^2)$. The claim we make is that the assignment tester V' will get reject any Π such that Π_ψ is δ -far from a satisfying assignment to ψ with probability at least $\Omega(\delta)$. Consider the scenario that V rejects a proof with probability δ^2 . It follows from our parameters that probability the random walk hits the bad set is at least $2 \cdot \delta$ and at least half of these tests fail over the choice of our random walk. Thus, the probability the we accept an invalid proof is bounded from above by $1 - \delta$. In other words, if V rejected an invalid proof with probability at least δ^2 , the new tester would reject it with probability at least δ . The distance parameter of Π_ψ is unaffected by this operation and hence, we inherit from Lemma 5.3 and 6.2 that if Π_ψ was δ -far from a satisfying assignment, it continues to be so.
- *Query size:* The arity of the equations in \mathcal{L}' is $\vartheta \cdot q$, where q is the arity of V .
- *Randomness:* Since, we are choosing a test of V uniformly at random and then taking a random walk of length ϑ on an expander G with a constant degree, we would be needing $r_V + \mathcal{O}(\vartheta)$, where r_V is the randomness used by V . \square

Corollary 6.4.1 (Tensoring + Repetition). *For every $\rho > 0$ and $\delta < 1/4$, there is an absolute constant ϑ , such that given an $(1 - \rho, \delta)$ -assignment tester that uses r_v random bits to make q queries, one can construct an $(1 - \mathcal{O}(\rho^2), \delta)$ -assignment tester. The new assignment tester uses $2 \cdot r_v + \mathcal{O}(\vartheta)$ random bits and makes $\mathcal{O}(\vartheta \cdot q^2)$ queries. Moreover, our latest construction has a linear predicate.*

Proof. We start with the assignment tester \mathcal{T} given by Lemma 5.3 that seeks a proof $\Pi = \sigma \circ \pi$, where σ are the variables of the Boolean predicate being verified and π is the auxiliary variables introduced by the assignment tester. Our new assignment tester \mathcal{T}' is obtained by invoking Lemma 6.2 and then followed by Lemma 6.4 on \mathcal{T} with parameters $\delta = 16/100$, $\vartheta = 5$ and $\rho = 1/(2 \cdot \vartheta)$.

<i>Parameters</i>	Tensoring	Soundness Amplification	After k Iterations
Completeness	$1 - \rho^2$	$1 - \mathcal{O}(\rho^2)$	$1 - \mathcal{O}(\rho^{k+1})$
Soundness	δ^2	δ	δ
Queries	q^2	$\vartheta \cdot q^2$	$\vartheta^{\sqrt{k}+1} \cdot q^k$
Alphabet	$\{0, 1\}$	$\{0, 1\}$	$\{0, 1\}$

Table 1: Parameters of a (ρ, δ) -assignment tester after various operations.

The proof required by \mathcal{T}' will be $\Pi^{\otimes 2}$. We now argue the completeness and soundness of the newly constructed assignment tester.

- *Completeness:* In the good case, if we have a proof that the input assignment tester T accepts with probability $1 - \rho$, we can use Proposition 3.1 to claim that after invoking Lemma 6.2 on T , we have a completeness of $1 - \rho^2$. Now, after the application of Lemma 6.4 it follows from our parameters that the completeness of $1 - \mathcal{O}(\rho^2)$.
- *Soundness:* Say, the newly constructed assignment tester is given a purported proof Π such that Π_ψ (denotes the assignment induced on σ by Π) is δ -far from a satisfying assignment to ψ , we will need to establish that it is rejected with probability at least $\Omega(\delta)$. One important inference from Observation 6.1 is that the neither tensoring nor soundness amplification change the assignment to σ and hence, its distance to a satisfying assignment of ψ . Hence, if the input assignment tester has the soundness for $\delta < 1/4$, we can invoke the soundness arguments of Lemma 6.2 and 6.4 to infer that soundness condition remains intact for $\delta < 1/4$. \square

Lemma 6.5 (Completeness Amplification Lemma). *For every $\rho > 0$ and $\delta < 1/4$, given a $(1 - \rho, \delta)$ -assignment tester with query complexity $\mathbf{q} = \mathcal{O}(1)$ and size 2^{r_v} , one can construct a $(1 - \mathcal{O}(\rho^k), \delta')$ -assignment tester with query complexity \mathbf{q} , $\mathbf{q} = 1/\rho^{k'}$, in time polynomial in $2^{k \cdot \mathcal{O}(r_v)}$. Also, the resulting assignment tester has a linear predicate.*

Proof. Say we have a $(1 - \rho, \delta)$ -assignment tester that verifies a Boolean predicate ψ using a proof over variables of ψ (call them σ) and auxiliary variables π . The idea is to let the tests of the $(1 - \mathcal{O}(\rho^k), \delta)$ -assignment tester be the tests of lemma 6.4.1 tensored with itself k times. Our $(1 - \mathcal{O}(\rho^k), \delta)$ -assignment tester uses the proof $(\sigma \circ \pi)^{\otimes k}$.

However, we would like to iteratively apply the output of Lemma 6.4.1 upon on itself. This makes it easier to apply the tools that we have build thus far. The first observation we make is that the after a $(1 - \rho, \delta)$ -tester, of size n , under goes r iterations of the tensoring followed by σ rounds of sequential repetition, the size of the resulting tester is $n^{2^{2^r-1}}$. This can be verified once we observe that we are squaring the size of the input instance in every iteration. By invoking Corollary 6.4.1, we infer that the completeness error is also squared in each iteration. Hence after r iterations, the new error parameter is $\rho^{2^{2^r-1}}$. However, the soundness of the tester remains unaffected due to the amplification given by Lemma 6.4. With this background, the Lemma follows by setting $r = \log \log k$. Notice that the proof required by this recursive construction is same as that of the iterative assignment tester constructed in previous paragraph. The new proof will be $(\sigma \circ \pi)^{\otimes k}$. Also, note that while the auxiliary variables blow up in every step, the variables of the Boolean predicate that we are checking remain the same. The parameters are summarized in Table 1.

- *Completeness:* It is by far the easier case to argue. Say, a $(1 - \rho, \delta)$ -assignment tester is given to us that enables us to verify a Boolean predicate using a proof Π . Then by Corollary 6.4.1,

we have a new assignment tester which has the square of the completeness error after a single iteration. It follows that after $\log \log k$ iterations, we end up with a completeness error of ρ^k .

- *Soundness:* Say that the $(1 - \mathcal{O}(\rho^k), \delta)$ -assignment tester is given access to a purported proof Π , we need to show that if the proof Π_ψ (denotes the assignment induced by Π on the variables of ψ) is δ -far to a satisfying proof of the Boolean predicate, the tester rejects with probability at least $\Omega(\delta)$. We show this property holds after each iteration and hence, the newly constructed assignment tester via this process always has the soundness condition.

The approach is simple. We give a proof by induction on the number of iterations. The assignment tester produced after the first iterate is sound by Corollary 6.4.1. Assume that the assignment tester produced after i -th iterations is sound. We now show that the assignment tester produced after $(i + 1)$ -th iteration is also sound. By assumption, the assignment tester at the i -th level has the soundness property, it now follows from Corollary 6.4.1 that the soundness holds even after $(i + 1)$ -th iteration.

- *Queries:* We again invoke Corollary 6.4.1 to conclude that the number of queries are first squared and then, multiplied by a factor of ϑ in every iteration. Since the basic tester of Lemma 5.3 made $q = \mathcal{O}(1)$ queries, after $\log \log k$ iterations we would have a query complexity of $q^k \simeq 1/\rho^{k'}$.
- *Randomness:* In every tensoring operation, we square the randomness (ref. Lemma 6.2). Thus, the *randomness* of the tester after r iterations is $k \cdot \mathcal{O}(r_v)$, where r_v is the randomness used by the $(1 - \rho, \delta)$ -tester of Lemma 5.3.
- *Alphabet:* The alphabet of the tester obtained via Lemma 5.3 has an alphabet $\{0, 1\}$ and it remains unchanged during the application of Lemma 6.4.1. \square

7 Final Brick: Composition

In this section, we use the assignment tester constructed in Section 6 to construct a PCP with linear predicate and logarithmically small completeness error and a constant soundness. However, the query size will be *poly-logarithmic*. We shall deal with query reduction in Section 8. For now, we shall restrict our attention to compose our assignment tester with low completeness error with a PCP to construct linear PCPs with sub-constant completeness error. The overall strategy will be to start with an outer PCP which has perfect completeness and is also robust. The notion of robust composition of PCP's was pioneered by [BSGH⁺06, DR06]. It involves two steps:

- **Robust Outer:** The key idea is to have a stronger soundness condition. In a robust PCP not only that the verifier must reject any invalid proof with a good probability, also the answers to the queries of PCP verifier are sufficiently far from any satisfying answer. Robust PCP's can be constructed from the extant PCP literature. We highlight two constructions.
 1. *Robust PCP* \equiv LABEL-COVER. The formulation of Robust PCP's might have arrived late, but they have been ever-existent and ubiquitous in the PCP world disguised as LABEL-COVER (ref. Section 8 to know more about LABEL-COVER). An interested reader might refer Lemma 2.5, page 11 of [DH09] to understand this equivalence.
 2. *Via Error Correcting Codes.* Dinur and Reingold [DR06] give a generic transformation of any PCP into a robust one. It roughly involves encoding the alphabet with an error correcting code $E : \Sigma \rightarrow \{0, 1\}^l$. The distance required from the code is determined by

	ROBUST OUTER	LINEAR TESTER	FINAL PCP	AMPLIFICATION
Soundness	ϵ	ϵ	$2 \cdot \epsilon$	$1/\mathcal{O}((\log n)^\beta)$
Completeness	1	$1 - 1/(\log n)^\alpha$	$1 - 1/(\log n)^\alpha$	$1 - 1/\mathcal{O}(\log n)^{\alpha'}$
Queries	$\mathcal{O}(1)$	$(\log n)^\alpha$	$(\log n)^\alpha$	2
Alphabet	$\mathcal{O}(1)$	$\{0, 1\}$	$\{0, 1\}$	$\{0, 1\}^{\log n}$
Size	n	$(\log n)^\beta$	$n \cdot (\log n)^\alpha$	$n \cdot (\log n)^{\mathcal{O}(\alpha)}$

Table 2: Various Parameters during Composition

the robustness parameter ρ . Also, if one would additionally require that the size of the reduction is quasi-linear, then we can use an error correcting code E to with a linear rate.

- **Composition with Assignment Tester:** We now run the assignment tester given by Lemma 6.5 (with required parameters) on every test $c_R : \Sigma^k \rightarrow \{0, 1\}$, determined by the randomness \mathbf{R} , of the aforementioned robust PCP verifier. One complication is that the assignment tester we have constructed is over a binary alphabet but the robust outer must be over a non-binary alphabet. However, notice that c_R can be transformed into a Boolean constraint over Boolean variables. To this end, replace each variable v by a set of l Boolean variables denoted by $[v]$ and the circuit for c_R by Boolean gates (essentially its binary encoding). So, we would end up with a new constraint \tilde{c}_R over new variables $[x_1] \cup [x_2] \dots [x_s]$. Moreover, c_R would be satisfied iff the assignment for $[x_1] \cup [x_2] \dots [x_s]$ is a satisfying assignment for \tilde{c}_R . Also, after carrying out the above transformation, we shall have a system of constraints $\{\tilde{c}_i\}_{i \in [n]}$. Thus, it is well defined to run an assignment tester on c_R .

After composing them with our assignment tester, we will end up with $\{\hat{c}_i\}$ over the old variables $[x_1] \cup [x_2] \dots [x_s]$ and new auxiliary variables Y . Firstly, it is easy to check that the size remains polynomial in n .⁵

Parameters after robust composition. During a robust composition, the randomness and errors of the final PCP are the sum total of the inner and the outer verifiers respectively. The query size, alphabet are inherited from the inner verifier.

As alluded earlier, in our composition, the outer verifier will be a Robust PCP. We start with a verifier possessing perfect completeness⁶ and low soundness error. Hence, the error parameters of the “new” composed verifier are solely determined by the errors of the assignment tester Π . This completes the outline of the composition. We state the robust outer that we intend to use for our composition. We use the low error LABEL-COVER (Definition 8) instance generated by Moshkovitz and Raz [MR10]. For the sake of completeness, we state it here.

Theorem 7.1 (Low-Error LABEL-COVER). *For every n , and every $\epsilon > 0$ (that can be any function of n) the following holds. Solving 3SAT on inputs on size n can be reduced to distinguishing between the case that a LABEL-COVER instance of size $n^{1+o(1)} \cdot \text{poly}(\frac{1}{\epsilon})$ and parameters $\log |\Sigma_A| \leq \text{poly}(\frac{1}{\epsilon})$ and $|\Sigma_B| \leq |\Sigma_A|$ is completely satisfiable and the case that at most ϵ fraction of the edges are satisfiable.*

⁵Follows from the fact that the size of every inner is $\mathcal{O}(\log n)^\beta$, for some $\beta < 1$ and there are at most $\mathcal{O}(n^{1+o(1)})$ constraints to compose.

⁶In other words, no error in completeness.

Notations. We denote by $PCP_{c,s}[r,q]_\Sigma$ the class of languages that have a PCP verifier with completeness c , soundness s , randomness r and makes q queries to the purported proof over an alphabet Σ . It is often useful to imagine all the parameters as a function of n .

Theorem 7.2. *For every $\alpha \leq 1$ and $\epsilon > 0$,*

$$3SAT \in PCP_{1-\frac{1}{(\log n)^\alpha}, \epsilon} \left[(1+o(1)) \cdot \log n, (\log n)^{\alpha'} \right]_{\{0,1\}}$$

Moreover, the verifier performs linear tests which have the projection property.

Proof. We now make the aforementioned construction explicit. We start with a randomness efficient robust PCP with perfect completeness and sub-constant soundness error. To obtain a robust PCP, we use its equivalence to LABEL-COVER. In particular, we start with a LABEL-COVER generated by Theorem 7.1 for some constant $\epsilon > 0$. In particular, we use a Robust PCP with robustness parameter set to some constant ϵ' . Now we invoke Lemma 6.5 to construct a $(1/(\log n)^\alpha, \epsilon)$ -tester (α will be fixed later). We then compose the inner with the LABEL-COVER instance in an obvious way. We summarize the parameters in Table 2. We analyze the parameters of the newly composed verifier.

- *Completeness:* Since the Label-Cover has perfect completeness, the error is solely contributed by the inner. Invoking Lemma 6.5, we conclude that the completeness is $1 - \frac{1}{(\log n)^\alpha}$.
- *Soundness:* The inner and the outer each contribute ϵ to the error. Hence, the overall soundness is $2 \cdot \epsilon$.
- *Query Size:* This is inherited from the query size of the inner. Since, we use Lemma 6.5 to obtain it, the inner makes $(\log n)^{\alpha'}$ queries.
- *Alphabet:* Again, follows from Lemma 6.5 that alphabet is $\{0,1\}$. Hence, the alphabet used by the proof is $\{0,1\}$.
- *Randomness:* The randomness of the composed verifier is the sum total of the randomness used by inner and that used by the outer. In our case, it follows from Theorem 7.1 and Lemma 6.5 that the inner uses $\log \log n \cdot \alpha' \cdot \mathcal{O}(r_v + \vartheta)$ random bits and the outer uses $\log n \cdot (1+o(1))$. So, the total number of random bits is $(1+o(1)) + \log n + \mathcal{O}(1) \cdot \log \log n \simeq (1+o(1)) \cdot \log n$.

□

8 Towards Reducing Queries, Soundness Amplification

One parameter we have not paid attention during the completeness amplification of the tester is the query size. As was the case with every other parameter, we also square the number of the queries the input tester makes. In general, the number of queries the tester makes is $\mathcal{O}(1/\rho)$. To establish PCPs for NP, one must be frugal when it comes to queries. So, we must apply some technique to keep reduce the queries. The obvious approach to break into 3-LIN does not work as we would lose huge factors $(1/\rho)$ in soundness. Thus, we use Bipartite *Locally Decode/Reject Codes* (LDRC) to reduce the number of queries in the PCP (from the perspective of GAP-LIN, we sparify the equation). We begin by defining Bipartite LDRC.

Definition 6 (Bipartite LDRC [MR10]). Consider a list of k -tuples

$$\langle i_{1,1}, \dots, i_{1,k} \rangle, \dots, \langle i_{N,1}, \dots, i_{N,k} \rangle \in [n]^k$$

A Bipartite LDRC for the k -tuples is $\mathcal{G} = \langle G = (A, B, E), A, B, \{\pi_e\}_{e \in E}, \{\tau_e\}_{e \in E}, \{\rho_e\}_{e \in E} \rangle$, where $\mathcal{G}' = \langle G = (A, B, E), A, B, \{\pi_e\}_{e \in E} \rangle$ is an instance of Label-Cover, and every edge $e \in E$ carries a k -tuple τ_e from the list and an evaluation function $\rho_e : \Sigma_A \rightarrow \{0, 1\}^k$. For each $j \in [N]$, the tuple $\langle i_{j,1}, \dots, i_{j,k} \rangle$ appears on the same number of edges.

Given a labeling to the vertices of the graph, i.e., functions $C_A : A \rightarrow \Sigma_A$ and $C_B : B \rightarrow \Sigma_B$, an edge $e = (a, b) \in E$ is said to be “satisfied” if it is satisfied in \mathcal{G}' . For a message $x \in \{0, 1\}^n$, the edge e is said to “decode” x if $\rho_e(C_A(a)) = \langle x_{i_{e,1}}, \dots, x_{i_{e,k}} \rangle$ where $\tau_e = \langle i_1, \dots, i_k \rangle$ is the tuple associated with e .

Let $0 < \delta_{\min} < 1$. Let $l_{\max} : (0, 1) \rightarrow \mathbb{R}^+$ be a decreasing function. We say that the LDRC is a $(\delta_{\min}, l_{\max})$ -bipartite LDRC if it satisfies the following conditions:

1. **Completeness:** For every $x \in \{0, 1\}^n$, one can efficiently compute assignments $C_A : A \rightarrow \Sigma_A$ and $C_B : B \rightarrow \Sigma_B$, such that all edges $e \in E$ are satisfied and decode x .
2. **Soundness:** For every $C_B : B \rightarrow \Sigma_B$, for every real δ such that $\delta_{\min} < 1$, there exist $l \leq l_{\max}(\delta)$ messages $x_1, \dots, x_l \in \{0, 1\}^n$, such that the following holds for any $C_A : A \rightarrow \Sigma_A$: when picking uniformly at random an edge $e \in E$, the probability that e is satisfied but does not decode any one of x_1, \dots, x_l , is at most $\mathcal{O}(\delta)$.

We now present a construction of an almost-linear size bipartite LDRCs due to Moshkovitz and Raz [MR10].

Definition 7 (Construction Algorithm, Theorem 15 in [MR10]). A $(k_{\max}, \delta_{\min})$ -construction algorithm for bipartite LDRCs with parameters $\langle \text{size}, \text{block}_A, \text{block}_B \rangle$ is an efficient algorithm that given a collection of k -tuples, where $k \leq k_{\max}$, outputs a $(\delta_{\min}, l_{\max})$ -bipartite LDRC for the tuples, where $l_{\max}(\delta) \leq \delta^{\mathcal{O}(1)}$. The size of the output is size , the alphabet size of the A vertices is 2^{block_A} and the alphabet size of the B vertices is 2^{block_B} .

Theorem 8.1 (Query Reduction, Theorem 16 in [MR10]). If there is a (q, ϵ) -construction algorithm for bipartite LDRCs with parameters $\langle \text{size} \leq (N+n) \cdot n^{\mathcal{O}(1)}, \text{block}_A, \text{block}_B \rangle$, then for some $\epsilon_0 \geq \epsilon^{\mathcal{O}(1)}$,

$$PCP_{1, \epsilon_0} \left[(1 + o(1)) \cdot \log n, q \right] \subseteq PCP_{1, \mathcal{O}(\epsilon)} \left[(1 + o(1)) \cdot \log n, 2 \right]_{\{0, 1\}^{\text{block}_A}}$$

Moreover, the transformation yields linear projection tests if the original instance had linear constraints.

Though, the claim on linearity is not explicitly stated in [MR10], it does hold. The key observation in establishing this invariant is that locally decode/reject codes are based on linear codes and are themselves linear by definition. One advantage of using LDRC for query reduction is that the composed verifier has all the qualities of a tester and thus, will be for ready for composition with another verifier in a seamless manner.

Theorem 8.2 (Linear PCP with Low Error). For some $\alpha, \beta > 0$

$$3SAT \in PCP_{1 - \frac{1}{(\log n)^{\alpha}}, \frac{1}{(\log n)^{\beta}}} \left[(1 + o(1)) \cdot \log n, 2 \right]_{\{0, 1\}^{\log n}}$$

Proof. Follows by invoking Theorem 8.1 on the tester obtained in Theorem 7.2. We set $\epsilon = \frac{1}{(\log n)^\beta}$ and then use the $((\log n)^\alpha, \epsilon)$ -construction algorithm for query reduction and soundness amplification. We now analyze the parameters of the PCP.

- *Completeness:* The completeness error of the PCP obtained in Theorem 7.2 is $(\log n)^\alpha$. Theorem 8.1 introduces an error which is proportional to the soundness error of the PCP obtained by applying it. In fact, Theorem 8.1 is essentially a parallel-repetition theorem in the low-error regime, albeit with a much worse alphabet tradeoff. Thus, the completeness error is that product of $(\log n)^\alpha \cdot \epsilon$. In other words, the error is not more than $(\log n)^{\alpha-\epsilon'}$, choose $(\log n)^\alpha = \omega(\epsilon')$ to get the parameters we want.
- *Soundness:* It follows from Theorem 8.1 that the soundness of the new construction is ϵ .
- *Alphabet:* We inherit the alphabet of Theorem 8.1, which happens to be $\{0, 1\}^{(\log n)^\alpha \cdot \text{poly}(\frac{1}{\epsilon})}$, where $\text{poly}(\cdot)$ is implicit in Definition 7. We choose $(\log n)^\alpha \cdot \text{poly}(\frac{1}{\epsilon}) = \log n$ to get the necessary parameters.
- *Randomness:* Theorem 8.1 blows up the randomness of the input tester to $1 + o(1) \cdot R$, where R is randomness of the PCP obtained by Theorem 7.2. For the parameters that we have chosen, it is $(1 + o(1)) \cdot \log n \cdot (1 + o(1)) \simeq \log n \cdot (1 + o(1))$.

□

Hardness of Label-Cover. PCP's with the projection property can also be formulated as a certain CSP called the LABEL-COVER problem. An instance of a LABEL-COVER is a bi-partite graph whose edges are between questions of first prover and those of the second prover. For every edge, there is an associated projection that uniquely identifies the label of second vertex given the label of a vertex from the first set. The goal is to find a labelling that maximizes the number of satisfied edges. The following formalizes the notion we have been talking in the prequel.

Definition 8 (LABEL-COVER). *An instance of LABEL-COVER contains a regular bi-partite multi-graph $G = (A, B, E)$ and two finite sets Σ_A and Σ_B , where $|\Sigma_A| \geq |\Sigma_B|$. Every vertex in A is supposed to get a label from Σ_A , and every vertex in B is supposed to get a label from Σ_B . For each edge $e \in E$ there is a projection $\pi_e : \Sigma_A \rightarrow \Sigma_B$ which is a partial function.*

Given a labeling to the vertices of the graph, that is, functions $\psi_A : A \rightarrow \Sigma_A$ and $\psi_B : B \rightarrow \Sigma_B$, an edge $e = (a, b)$ is said to be “satisfied” if $\pi_e(\psi_A(a)) = \psi_B(b)$ (if $\pi(\psi_A(a))$ is undefined; in which case the edge is deemed to be unsatisfied.) The goal is to find a labeling which maximizes the number of satisfied edges.

We say that γ fraction of the edges are satisfiable if there exists a labeling that satisfies γ fraction of the edges. In the LABEL-COVER notation, the size corresponds to the number of vertices $|A| + |B|$. The alphabet corresponds to the larger set of labels Σ_A .

The LABEL-COVER seems to be extremely useful in establishing inapproximability results. Khot's survey [Kho05] is an excellent exposition on the gamut of hardness results that can be obtained from LABEL-COVER. In the world of provers, the projection property is equivalent to a 2 prover 1 round game and has the following correspondence with the LABEL-COVER: The vertices in A and B are the set of questions that can be posed by the verifier to the first prover and the second prover respectively. The set of labels corresponds to the answers of the provers. So, upon receiving an answer from the first prover, the verifier rejects the claim made by the provers or has uniquely determined the answer to the second prover's question.

Theorem 10.12 can be reformulated in the language of LABEL-COVER as follows.

Corollary 8.2.1. *For every n and for some $\alpha, \beta > 0$ the following holds. Solving an instance of SAT of size n can be reduced to distinguishing between the following two cases of a LABEL-COVER instance.*

- **Yes:** *There is a labeling that satisfies at least $(1 - \frac{1}{(\log n)^\alpha})$ -fraction of the edges.*
- **No:** *Any labeling satisfies at most $(\frac{1}{(\log n)^\beta})$ -fraction of the edges.*

Moreover, the size of the LABEL-COVER instance is $n^{1+o(1)}$ and every projection is linear in nature.

9 Hardness Results

We begin by borrowing some machinery from the PCP literature. We will deal with Boolean functions $A : \mathbb{F}_2^u \rightarrow \{-1, 1\}$. A function is called linear if $A(x \oplus y) = A(x) \cdot A(y)$, where \oplus denotes vector addition over \mathbb{F}_2 . Note that there are precisely 2^u linear functions: for every $\alpha \in \mathbb{F}_2^u$, χ_α defined as

$$\chi_\alpha(a) = (-1)^{a \cdot \alpha} \quad \forall a \in \mathbb{F}_2^u$$

Hadamard Codes. Our PCP verifier will expect an encoding of the proof described in the prequel. Specifically, we will use Hadamard codes which we define now.

Definition 9 (Hadamard Encoding). *Hadamard code of $p \in \mathbb{F}_2^u$ is the 2^u bit string $\{\chi_p(a)\}_{a \in \mathbb{F}_2^u}$. We denote it by $\text{Hadamard}(p)$.*

Recall that the string $x = Q(W)$ read by the aforementioned verifier is supposed to satisfy certain linear conditions modulo 2. Let these constraints be $h_1 \cdot x = \eta_1, \dots, h_u \cdot x = \eta_u$, where $h_1, h_2, h_3 \dots h_u \in \mathbb{F}_2^{3u}$ and $\eta_1, \eta_2 \dots \eta_u \in \mathbb{F}_2$.

Folding. Informally, *Folding* is a technique used to enable the verifier to ignore the linear constraint test. Suppose that C is a Hadamard code of x and x satisfies the constraints mentioned above. Let \mathcal{H} be the linear subspace spanned by the vectors $h_1, h_2 \dots h_u$. Then for any vector b and any vector $h \in \mathcal{H}$, $h = \oplus \rho_i \cdot h_i$, we have

$$C(b \oplus h) = C(b) \cdot (-1)^{\sum_i \rho_i \eta_i}$$

One may generalize this observation, for any function $F : \mathbb{F}_2^{3u} \rightarrow \{-1, 1\}$, Let v_b denote the lexicographically smallest vector in the set of vectors $b \oplus \mathcal{H}$ (coset of \mathcal{H}). we one may define a function F' as:

$$b = v_b \oplus \rho_i \cdot h_i, \quad \rho_1, \dots, \rho_u \in \mathbb{F}_2 \quad F'(b) = F(b) \cdot (-1)^{\sum_i \rho_i \eta_i}$$

F' is said to be a folding of F over the linear constraints.

The verifier we design requires the supposed Hadamard codes be folded over the respective constraints. This does not alter much. One difficulty that it may introduce is ensuring that this requirement is enforced. This is done as follows: If the verifier is required to read $F(b)$ from the code, it can read $F(v_b)$ and can calculate $F(b)$ from the expression given in the prequel.

A comment on folding of Hadamard codes is due. We will eventually need to show that verifier accepts the purported proofs with a good enough probability, then these “supposed” proofs can be decoded to construct an assignment of the variables which satisfies a significant fraction of the constraints. Decoding of F gives h with probability \hat{H}_h^2 . Now, *folding* ensures that any h given by this decoding procedure satisfies the linear constraints on variables. Thus, folding lets us to ignore the linear constraints altogether.

9.1 The PCP verifier

We now define and analyze our PCP verifier whose tests will be linear equations over 3 variables. The verifier expects to have proofs (Π_A, Π_B) which are the Hadamard encoding of the assignments given to the vertices in A and B . The verifier does the following:

1. Pick at random edge $e = (x, y)$. Let f, g be the corresponding Hadamard encoding of the labels of vertices x and y respectively; and h be the corresponding projection function between Σ_A and Σ_B . $h^{-1}(q)$ denotes the unique vector $p \in \Sigma_A$ such that $h(p) = q$.
2. Choose \mathbf{v}, \mathbf{y} at random conditioned on $\mathbf{v} \in \{\pm 1\}^{\log |\Sigma_A|}$ and $\mathbf{y} \in \{\pm 1\}^{\log |\Sigma_B|}$. Accept iff

$$f(\mathbf{v}) \cdot g(\mathbf{y}) = f(h^{-1}(\mathbf{y}) \oplus \mathbf{v})$$

We begin by stating a Theorem of Khot [Kho01]. Khot uses a Hadamard based PCP on an instance of LABEL-COVER produced by applying parallel repetition on a linear PCP with completeness $1 - \epsilon$ and soundness μ . The linear PCP is queried k times in parallel and has a soundness of S . We start with a LABEL-COVER with better parameters, so we set $k = 1$. We now take the liberty and rephrase Khot's Theorem about the guarantees provided by the verifier V_{lin} (Page 4, Theorem 3.1 in [Kho01]) in the language of LABEL-COVER.

Theorem 9.1. *Every LABEL-COVER instance with linear projection tests and size n has a verifier V_{lin} which*

- *Uses $r = \log n + \mathcal{O}(1)$ random bits.*
- *Performs linear tests each with arity 3.*
- *Has completeness at least $1 - \epsilon$.*
- *Has soundness $\frac{1}{2} + \delta$ provided $S < \delta^2$, where S is the soundness of LABEL-COVER instance.*

Theorem 9.2. *For some $\alpha, \beta > 0$, 3SAT has a verifier \mathcal{V} that has the following properties.*

- *Uses $\log n \cdot (1 + o(1))$ random bits.*
- *The tests performed by \mathcal{V} are linear over 3 variables.*
- *If the SAT instance is satisfiable, \mathcal{V} accepts it with probability at least $1 - \frac{1}{(\log n)^\alpha}$.*
- *If it is unsatisfiable, \mathcal{V} accepts any proof with probability at most $\frac{1}{2} + \frac{1}{(\log n)^{\beta'}}$.*

Proof. We now use Khot's verifier V_{lin} on the LABEL-COVER instance in Corollary 8.2.1. Specifically, set $\epsilon = \frac{1}{(\log n)^\alpha}$, and $\delta = \frac{1}{(\log n)^{\frac{1}{\beta}}}$. Completeness follows trivially from Theorem 9.1. Since $S < \delta^2$, the soundness of V_{lin} is $\frac{1}{2} + \frac{1}{(\log n)^{\beta'}}$. This completes the proof. \square

9.2 Inapproximability Results

The PCP theorem 9.2 immediately yields a hardness of $\frac{1}{2} + \frac{1}{(\log n)^\beta}$ for 3LIN assuming $P \neq NP$. This translates into a hardness factor of $\frac{7}{8} + \frac{1}{(\log n)^\beta}$ for 3SAT.

Corollary 9.2.1 (3LIN HARDNESS). *For some $\alpha, \beta > 0$, given a 3LIN instance Φ of size n it is NP-hard to distinguish between the following two cases.*

- *There is an assignment which satisfies $(1 - \frac{1}{(\log n)^\alpha})$ -fraction of Φ .*
- *Every assignment can satisfy at most $(\frac{1}{2} + \frac{1}{(\log n)^\beta})$ -fraction of Φ .*

Proof. The constraint satisfaction version of Theorem 9.2. □

Corollary 9.2.2 (3SAT HARDNESS). *For some $\alpha, \beta > 0$, it is NP-hard to distinguish if a given 3SAT instance Υ of size n which of the following holds.*

- *There is an assignment which satisfies $(1 - \frac{1}{(\log n)^\alpha})$ -fraction of Υ .*
- *Every assignment can satisfy at most $(\frac{7}{8} + \frac{1}{(\log n)^\beta})$ -fraction of Υ .*

10 Polynomially Small Completeness Error

In this section, we construct PCPs with polynomially low completeness error and yet, have a low (even sub-constant) soundness error. The bottleneck to achieving polynomially low completeness error was that the number of queries grew hand in hand with the completeness error. Thus, if we set $r = \log \log n$ in Lemma 6.5 to get polynomially small completeness error, the queries made will be $\text{Poly}(n)$. At which point, any technique to rescue the query size will result in either perturbing the completeness error to a constant or soundness error into the $1/\text{polylog}$ regime. And both these means defeat the ends that we set out to achieve as the former destroys the completeness error. While the latter shall take us into a region which we currently cannot come out of. Thus, both these approaches are ruled out.

However, we cannot allow the query size to grow along the completeness if we are to achieve out ends. The approach we adopt is simple, we use composition to reduce the query size and at this point of time, there are a lot of already established composition theorems. However, none of them will be suitable to our needs as we need a low completeness error and almost all the exant theorems either are non-linear or have a constant error in completeness. So, in a sense we are forced to self-compose with our constructions. And as it turns out it is not the worst thing that can happen. To that end, we start by recollecting Corollary 6.4.1 that we construct after a tensoring and performing sequential repetition on a basic assignment tester obtained from Lemma 5.3.

Corollary. *For every $\rho > 0$ and $\delta < 1/4$, there is an absolute constant ϑ , such that an $(1 - \rho, \delta)$ -assignment tester that uses r_v random bits to make q queries, we can construct an $(1 - \mathcal{O}(\rho^2), \delta)$ -assignment tester. The new assignment tester uses $2 \cdot r_v + \mathcal{O}(\vartheta)$ random bits and makes $\mathcal{O}(\vartheta \cdot q^2)$ queries. Moreover, our latest construction has a linear predicate.*

Let us take a moment and reflect the hypothetical parameters that one might obtain if we composed the above Corollary with itself (presented in Table 3). In short, we would double the error while reducing the queries as a function of n (ref. Claim 10.1)⁷. So, our blue print of the construction will be as follows.

Claim 10.1. *For every non-decreasing $f : \mathbb{N} \rightarrow \mathbb{N}$ such that $f(n) = o(n)$, the following holds for every x .*

⁷We prove a strong version of this Claim at a later point

<i>Parameters</i>	INNER	SELF COMPOSED INNER
Completeness error	$\rho(n)$	$2 \cdot \delta(n)$
Soundness error	$\delta(n)$	$2 \cdot \delta(n)$
Queries	$q(n)$	$q(q(n))$
Randomness	$r(n)$	$r(n) + r(r(n))$
Alphabet	$\{0, 1\}$	$\{0, 1\}$

Table 3: Parameters after Self-Composition of Inner

- $f(f(x)) \leq f(x)$.
- $f(f(x)) < x/k$, for all $k > 0$.

Proof. The first part of the claim follows from the hypothesis that f is a non-decreasing, sub-linear function. For the latter half, suppose that $f(f(x)) = x/k$, for some constant k . This implies $f(x) = O(n)$ and this is a contradiction. Thus, the claim holds. \square

Modified Composition

1. We start with the basic $(1 - \rho, \delta)$ -assignment tester constructed in Lemma 5.3. We then tensor the $(1 - \rho, \delta)$ -assignment tester to obtain a $(1 - \rho^2, \delta^2)$ -tester. As in Lemma 6.4 we perform soundness amplification in a randomness efficient manner to obtain $(\vartheta \cdot \rho^2, \delta)$ -assignment tester, denoted by \mathcal{V} . The query complexity $q(n)$ is $\vartheta \cdot q^2$, where q is the number of queries made by the assignment tester in Lemma 5.3.
2. We now use self-composition to compose \mathcal{V} with itself. As we have witnessed in Table 3, this doubles the error parameters⁸. Thus, we end up with a $(1 - 2 \cdot \vartheta \cdot \rho^2, 2 \cdot \delta)$ -tester with a query complexity of $q(q(n))$, where n is the size of the outer. Set $\vartheta = 5, \rho = 1/(8 \cdot \vartheta), \delta = 16/100$ to check that we essentially have a $(1 - \mathcal{O}(\rho^2), \delta)$ -tester
3. We then iterate over the first two steps $\simeq \log n$ times. It follows that the tester we achieve after this is $(1 - 1/\mathcal{O}(\rho^{\log n}), \delta')$ -tester. This is what we would like to achieve as $\mathcal{O}(\rho^{\log n}) \simeq \mathcal{O}(1/n^{\Omega(1)})$. The query complexity of our construction decreases over each iteration in n . It take a bit of work but it is not hard to show that at the end of it all, we end up with a tester that only makes a constant number of queries.

However, all is not well as the above sketch suggests. For one, we have designed the basic tester to be utilized as an inner. To be eligible for composition as an outer, we need an robust variant of the same. We shall use this opportunity to refresh the notion of verifier composition. The prover would need to provide an inner-proof ψ_R for every possible random coin R of the outer PCP. Thus, the proof for the composed verifier is $\Psi = \{\psi_R : R \text{ outer random coins}\}$. Thus, if our tester to be eligible to be used as an outer, it needs to be robust. Informally, any satisfying assignment must be far from an unsatisfying assignment. We now formalize this notion via *robust* assignment testers first introduced by [DR06, BSGH⁺06].

Definition 10 (Robust Assignment Testers). *An assignment-tester is called ρ -robust if in the soundness case in Definition 4 of an assignment tester, for every assignment $\tilde{\pi} : Y \rightarrow \Sigma$, the*

⁸Will be established via Lemma 10.3

assignment $(\pi \cup \tilde{\pi})|_{\psi_i}$ is ρ -far from any satisfying assignment of ψ_i on least $1 - \epsilon$ fraction of $\Psi = \{\psi_1, \dots, \psi_R\}$.

Dinur and Reingold [DR06] provide a generic transformation that transforms any assignment-tester into a robust one. In particular, they prove the Robustization lemma which is very useful in our quest. Specifically, we use a slight variant of their result – one that includes completeness error. Let us denote a tester that uses R random bits, has a completeness error ρ , size S , makes q queries, distance parameter δ , soundness ϵ and robustness parameter μ by $(R, \rho, S, q, \delta, \epsilon, \mu)$.

Lemma 10.2 (Robustization Lemma, Lemma 3.6 in [DR06]). *There exists some $c_1 > 0$ such that given an assignment tester \mathcal{A} with parameters $(R, \rho, S, q, \delta, \epsilon)$, we can construct a ρ -robust assignment tester $\mathcal{A}' = \Psi'$ with parameters $(R, \rho, c_1 \cdot S, q, \delta, \epsilon, \mu = \Omega(1/q))$.*

Based on Lemma 10.2, we can now transform our tester obtained from Lemma 5.3 into a robust tester. The key parameter for us is the completeness error and this is left untouched by this generic robustization. We also recollect the robust composition lemma from [DR06] which basically establishes that robust assignment-testers compose in a modular fashion.

Lemma 10.3 (Robust Composition Lemma 3.5 in [DR06]). *Let \mathcal{A}_1 and \mathcal{A}_2 be two assignment testers with parameters $(R_1, \rho_1, S_1, q_1, \delta_1, \epsilon_1)$ and $(R_2, \rho_2, S_2, q_2, \delta_2, \epsilon_2)$ respectively. If \mathcal{A}_1 is μ -robust with $\mu = \delta_2$ then one can construct an assignment tester \mathcal{A}_3 with parameters $(R_3, \rho_3, S_3, q_3, \delta_3, \epsilon_3)$ such that:*

$$R_3 = R_1(n) \cdot R_2(S_1(n)), \quad S_3(n) = S_1(S_2(n)), \quad q_3(n) = q_2(S_1(n)), \quad \rho_3 = \rho_1(n) + \rho_2(S_1(n)) - \rho_1(n) \cdot \rho_2(S_1(n))$$

and

$$\epsilon_3 = \epsilon_1(n) + \epsilon_2(S_1(n)) - \epsilon_1(n) \cdot \epsilon_2(S_1(n)), \quad \delta_3(n) = \delta_1(n)$$

10.1 Construction of Tester

We now highlight the iterative construction aimed at constructing PCP's with polynomially small completeness error. The construction involves *three* phases as outlined in Section 10.

Tensoring and Soundness Amplification. We begin by recalling the tester we obtained after tensoring a $(1 - \rho, \delta)$ -tester followed by ϑ sequential repetitions in a randomness efficient manner via Corollary 6.4.1.

Corollary (Tensoring + Repetitions). *For every $\rho > 0, \delta < 1/4$, there is an absolute constant ϑ , such that an $(1 - \rho, \delta)$ -assignment tester that uses r_v random bits to make q queries, we can construct an $(1 - \mathcal{O}(\rho^2), \delta)$ -assignment tester. The new assignment tester uses $2 \cdot r_v + \mathcal{O}(\vartheta)$ random bits and makes $\mathcal{O}(\vartheta \cdot q^2)$ queries. Moreover, our latest construction has a linear predicate.*

Robustization. In order to compose the tester we have built with itself. It must be robust in the sense of Definition 10. So, our approach will be to robustize the tester via Lemma 10.2 and then compose it with the tester obtained by Corollary 6.4.1 using Lemma 10.3.

Lemma 10.4 (Robust Tester). *There exists some $\gamma > 0, \delta < 1/4$, such that as $(1 - \rho, \delta)$ -tester \mathcal{T} that uses r_v random bits to make q queries to verify a Boolean property ψ can be made a μ -robust assignment tester \mathcal{A} with parameters $(r_v, \rho, \gamma \cdot S(\mathcal{T}), q, \delta, \epsilon, \mu = \Omega(\frac{1}{q}))$.*

Proof. We invoke Lemma 10.2 on the (ρ, δ) -tester from Lemma 5.3. \square

Self Composition. We now compose the robust tester obtained by Lemma 10.4 with the tester obtained via Corollary 6.4.1 as its inner.

Lemma 10.5. *For every $\rho > 0$ and $\delta < 1/4$, there is some $\epsilon > 0$, the composition of the assignment tester obtained via Corollary 6.4.1 with itself in a robust way yields an assignment tester with the following parameters $(\mathcal{O}(r_v), \mathcal{O}(\rho^2), S(S(n)), q(q(n)), \delta, \epsilon)$ -tester.*

Proof. The idea is very simple. We start with the basic tester $(r_v, \rho^2, S(n), q(n), \delta, s')$ -tester obtained from Corollary 6.4.1. We then invoke Lemma 10.2 on it to obtain its robust cousin $(r_v, \rho^2, \gamma \cdot S(n), q(n), \delta, \epsilon, \mu = \Omega(1/q(n)))$ -tester. We now compose them in an obvious way using Lemma 10.3 to claim the result.

In particular, we begin by setting $\vartheta = 5, \rho = 1/(8 \cdot \vartheta)$ in Lemma 6.4.1 to obtain an $(1/64 \cdot \vartheta, \delta')$ -tester with q queries. Upon composition with a robust version of itself we end up with a $\rho = 1/32 \cdot \vartheta = 1/2 \cdot 1/64 \cdot \vartheta \simeq \mathcal{O}(\rho^2)$ and $\delta = \Omega(\frac{1}{q}) = \mathcal{O}(1) \simeq \epsilon^9$. The rest of the parameters follow vacuously. \square

Iterated Tester

We are now almost set to iterate over Corollary 6.4.1, Lemmas 10.4 and 10.5 to obtain our ends. But before we go any further, we would like to introduce a couple of definitions and make a few observations about the function $q(n)$. At the very outset, we make it clear that in what follows we deal with only sub-linear functions. That is, functions whose output is small than its input by a $\text{poly}(\cdot)$ quantity. Examples of such functions include $\log n$, $\text{poly log } n$, n^ϵ , for $\epsilon < 1$.

Definition 11 (Repetition). *We define $\text{Repetition}(\mathcal{V})$ as an invocation of Corollary 6.4.1 on \mathcal{V} , followed by robustization using Lemma 10.4, and finally self composition using Lemma 10.3. Further, for any $k > 0$ we define $\text{Repetition}^k(\mathcal{V})$ recursively as $\text{Repetition}(\text{Repetition}^{k-1}(\mathcal{V}))$.*

Corollary 10.5.1. *For every $\rho > 0$ and $\delta < 1/4$, there is a $k > 0$ such that $\text{Repetition}(\mathcal{T})$ yields a $(k \cdot r_v, \mathcal{O}(\rho^2), \mathcal{O}(S^k), \delta)$ -tester.*

Proof. Essentially restating Lemma 10.5 expect for we use an explicit constant to account for the randomness used by in a single invocation of Repetition. \square

Definition 12. *Let \mathcal{T} be an assignment tester with a query function $q : \mathbb{N} \rightarrow \mathbb{N}$. For any i , we define $q^i(n)$ as the query size of the tester produced after $\text{Repetition}^i(\mathcal{T})$.*

Proposition 10.6. *For every tester \mathcal{T} with query complexity $q(n)$, $k > 0$, $q^k(n) = [q^{k-1}((q^{k-1}(n))^2)]^2$.*

Proof. Follows from the definition of Repetition. The query size of the input tester to the Repetition^i is q^{i-1} . Recall Corollary 6.4.1 to conclude that the tensor and sequential repetition operations increase the query function to $Q = \mathcal{O}(q^{i-1}(n))^2$. The robustization step does not increase the number of queries. And from Lemma 10.3, the self composition step results in a query size of $Q(Q(n))$, which is $[q^{k-1}((q^{k-1}(n))^2)]^2$. \square

Lemma 10.7. *The sequence $Q_{i \in \mathbb{N}} = \{q^i(n)\}$ is well behaved in the sense that given q^k , one can compute q^{k-1} and q^{k+1} upto constant factors.*

⁹Since, this is a one-time composition, it is a constant.

Proof. In order to be able to compute q^{k-1} we would need uncompute the effect of the k -th Repetition and we are perfectly capable to do à la Proposition 10.6. The argument for computing q^{k+1} follows arguments from Proposition 10.6. \square

Corollary 10.7.1 (Smoothness Property). *The sequence $\mathcal{Q}_{i \in \mathbb{N}} = \{q^i(n)\}$ is smooth in the following sense.*

$$q^k = \Theta(f) \implies q^{k-1} = \Theta(\tilde{f}) \quad \text{and} \quad q^{k+1} = \Theta(\hat{f})$$

Moreover, both \tilde{f} and \hat{f} are sub-linear in their input.

Definition 13 ($\mathcal{Q} = \Upsilon(f)$). *For any $f : \mathbb{N} \rightarrow \mathbb{N}$ and any $\mathcal{Q}_{i \in \mathbb{N}} = \{q^i(n)\}$, we say $\mathcal{Q} = \Upsilon(f)$ if there is a constant C , such that $q^i(n) = \Theta(f^i(n))$ for every $i > C$, where $f^i(n)$ is the function obtained by composing f with itself for i times.*

Lemma 10.8. *The sequence $\mathcal{Q}_{i \in \mathbb{N}} = \{q^i(n)\}$ decreases faster than any function in n . In other words, there is no f such $\mathcal{Q} = \Upsilon(f)$.*

Proof. We give a proof by contradiction. Suppose there is a function f such that $\mathcal{Q} = \Upsilon(f)$. Thus for every $i > C$, $q^i(n) = \Theta(f^i(n))$. Since, every element in \mathcal{Q} can be used to good effect in computing its predecessor, we can go backwards until the point where we have reached the start of the sequence or q^j is no longer sub-linear. In the former, we arrive at a contradiction as the initial tester in the sequence had a query complexity independent of n or a constant number of queries. In the latter case, we contradict the smoothness property of Corollary 10.7.1 as $q^j = \Theta(f)$ implies the existence of $q^{j-1} = \Theta(\tilde{f})$. \square

Proposition 10.9. $\mathcal{Q}_{i \in \mathbb{N}} = \{q^i(n)\}$ is independent of n . In other words, any $q^i(n) = \mathcal{O}(1)$.

Proof. It follows from Lemma 10.8 that the sequence \mathcal{Q} is not bounded from below by any $f(n)$. Thus, \mathcal{Q} is independent of n . \square

Theorem 10.10. *For every $k, \rho > 0$ and $\delta < 1/4$, there exists $m = k(\cdot)$ and constants $q', \epsilon > 0$, such that Repetition^m(r_v, ρ, S, q, δ)-assignment tester yields a $(r_v \cdot \log k, \rho^k, S^k, q', \delta, \epsilon)$ -tester. Also, the newly constructed assignment tester performs only linear tests.*

Proof. Recall that even though sequential repetition, self composition increase the size, the tensoring step dominates the rest of them asymptotically. Since in this study, we primarily deal with asymptotics, we can focus on the blow-up due to tensoring. Repeated tensoring essentially squares the size of the input instance. It is easy to verify that the sequence generated is $\{2^{2^{i-1}}\}_i$.

Corollary 10.5.1 gives the parameters after a single invocation of Repetition. To establish the Theorem would essentially require us to export each parameter across each invocation of Repetition. We invoke it with $m = \log \log k$. We now analyze and argue the parameters of the tester.

- *Randomness:* As highlighted in the prequel, the randomness used is asymptotically dominated by the tensoring operation. Thus, we would square the randomness used at every iteration. Hence if the input tester uses r_v bits, after m iterations the randomness blows up to $r_v \cdot 2^{2^m} \simeq r_v \cdot k$.
- *Completeness:* Again, we square the completeness error in each iteration, the error falls doubly exponential in m , which is $\mathcal{O}(k)$. Hence, the completeness error at the end of m iterations is $\mathcal{O}(\rho^k)$.

- *Query Size:* We recall Proposition 10.9 to conclude that the query $q(n)$ is independent of n and thus, we maintain a constant query size after each invocation.
- *Soundness:* Say that $\text{Repetition}^m((r_v, \rho, S, q, \delta)\text{-tester})$ is given access to a purported proof Π , we need to show that if the proof Π_σ (denotes the projection of Π on the variables of ψ) is δ' -far to a satisfying proof of the Boolean predicate, we reject with probability at least $\Omega(\delta)$. We show this property holds after each iteration and hence, the newly constructed assignment tester via this process always has the soundness condition. We begin by defining the projection of a proof on σ . We denote by Π_σ the assignment which Π makes to the variables in σ . We begin by recalling Observation 6.1, which states that the distance of assignment to σ is invariant to tensoring of proof. In fact, the same is vacuously true of sequential repetition as it does not modify the proof. We are now ready to establish the soundness.

The approach is simple. We give a proof by induction on the number of iterations. The assignment tester produced after the first iterate is sound by Corollary 6.4.1. Assume that the assignment tester produced after i -th iterations is sound. We now show that the assignment tester produced after $(i+1)$ -th iteration is also sound. Since, the assignment tester at the i -th level has the soundness property, it follows from Corollary 6.4.1 that the soundness holds even after $(i+1)$ -th iteration. Thus, it is true that the output of $\text{Repetition}^m((r_v, \rho, S, q, \delta)\text{-tester})$ is sound.

- *Distance Parameter:* In our construction, this is same as the soundness.

The linearity of the tests follows from the observation that the basic steps of this operation including tensoring, sequential repetition, self composition preserve linearity. \square

10.2 Final Composition

Theorem 10.11. *For every n and $\alpha > 0$ and some ϵ ,*

$$3SAT \in PCP_{1 - n^{\frac{\alpha'}{\alpha' + 1}}, \epsilon} \left[(1 + \alpha) \cdot \log n, \mathcal{O}(1) \right]_{\{0,1\}}$$

Moreover, verifier's test are linear, and also have the projection property.

Proof. We invoke Theorem 10.11 to construct a $(1/n^\alpha, \epsilon)$ PCP. We then compose the inner with the Label-Cover instance in an obvious way. We analyze the parameters of the newly composed verifier.

- *Completeness:* Since the Label-Cover has perfect completeness, the error is solely contributed by the inner. Invoking Theorem 10.10, we conclude that the completeness is $1 - n^{\frac{-\alpha'}{\alpha' + 1}}$.
- *Soundness:* The inner and the outer each contribute $1/\epsilon$ to the error. Hence, the overall soundness is $1/\mathcal{O}(\epsilon)$.
- *Query Size:* This is inherited from the query size of the inner. Since, we use the inner obtained via Theorem 10.10, the inner makes $\mathcal{O}(1)$ queries.
- *Alphabet:* Again, follows from Theorem 10.10 that alphabet is $\{0, 1\}$.
- *Randomness:* The randomness of the composed verifier is the sum total of the randomness used by inner and that used by the outer. In our case, it follows from Theorem 10.10 and Lemma 7.1 that the inner uses $\log n \cdot \alpha'$ random bits and the outer uses $\log n \cdot (1 + o(1))$. So, the total number of random bits is $\alpha' \cdot \log n + (1 + o(1)) \cdot \log n \simeq (1 + \alpha + o(1)) \cdot \log n$.

\square

Soundness Amplification

The tester obtained by Theorem 10.11 has a constant soundness, for some arbitrary constant. We now amplify the soundness of the tester whilst losing modest factors in the completeness. Again, we would like to amplify the soundness whilst retaining the good properties which are helpful in making the composition modular even possibly at the expense of a non-binary alphabet. In fact, to achieve non-constant soundness we would need larger alphabet. To this end, we shall use Locally Decode and Reject Codes (Definition 6 of Moshkovitz and Raz [MR10]). To amplify the soundness, we use Theorem 8.1 on the tester obtained from Theorem 10.11. The construction follows.

Theorem 10.12 (Linear PCP with Low Error). *For every $\alpha, \epsilon > 0$*

$$3SAT \in PCP_{1-n^{-\frac{\alpha'}{1+\alpha'}}, \epsilon} \left[(1+o(1)) \cdot \log n, 2 \right]_{\{0,1\}^{\text{poly}(\frac{1}{\epsilon})}}$$

Proof. Follows by invoking Theorem 8.1 on the tester obtained in Theorem 10.11. We use the (q, ϵ) -construction algorithm for query reduction and soundness amplification, where q is constant obtained from Theorem 10.11. We now analyze the parameters of the PCP.

- *Completeness:* The completeness error of the PCP obtained in Theorem 10.11 is n^α . Theorem 8.1 introduces an error which is proportional to the soundness error of the PCP obtained by applying it. In fact, Theorem 8.1 is essentially a parallel-repetition theorem in the low-error regime, albeit with a much worse alphabet tradeoff. Thus, the completeness error is that product of $n^\alpha \cdot \epsilon$. In other words, the error is not more than $n^\alpha \cdot \epsilon$, choose $n^\alpha = \omega(\epsilon')$ to get the parameters we want.
- *Soundness:* It follows from Theorem 8.1 that the soundness of the new construction is ϵ .
- *Alphabet:* We inherit the alphabet of Theorem 8.1, which happens to be $\{0,1\}^{\text{poly}(\frac{1}{\epsilon})}$. For poly alphabet, choose $\text{poly}(\frac{1}{\epsilon}) = \log n$ to get the necessary parameters.
- *Randomness:* Theorem 8.1 blows up the randomness of the input tester to $1 + o(1) \cdot R$, where R is randomness of the PCP obtained by Theorem 10.11. For the parameters that we have chosen, it is $(1 + o(1)) \cdot (1 + \alpha + o(1)) \cdot \log n \simeq \log n \cdot (1 + \alpha + o(1))$.

□

Corollary 10.12.1 (LABEL-COVER). *For every n, α and some $\beta > 0$ the following holds. Solving an instance of SAT of size n can be reduced to distinguishing LABEL-COVER instances between the following two cases.*

- **Yes:** *There is a labeling that satisfies at least $(1 - \frac{1}{n^{\alpha'}})$ -fraction of the edges.*
- **No:** *Any labeling satisfies at most $(\frac{1}{(\log n)^\beta})$ -fraction of the edges.*

The size of the LABEL-COVER instance is $n^{1+\alpha}$ and every projection is linear in nature.

Corollary 10.12.2 (3LIN HARDNESS). *For every α , there is some $\beta > 0$, the following holds. Given a 3LIN instance Φ of size n , it is NP-hard to distinguish between the following two cases.*

- **Yes:** *There is an assignment which satisfies $(1 - \frac{1}{n^{\alpha'}})$ -fraction of Φ .*
- **No:** *Every assignment can satisfy at most $(\frac{1}{2} + \frac{1}{(\log n)^\beta})$ -fraction of Φ .*

Proof. Plugging in the LABEL-COVER obtained from Corollary 10.12.1 instead of the one obtained by Corollary 8.2.1 in Theorem 9.2 by setting $\epsilon = \frac{1}{n^\alpha}$, and $\delta = \frac{1}{(\log n)^{\frac{1}{\beta}}}$. Completeness follows trivially from Theorem 9.1. Since $S < \delta^2$, the soundness of V_{lin} is $\frac{1}{2} + \frac{1}{(\log n)^{\beta\gamma}}$. This completes the proof. \square

Corollary 10.12.3 (MIN-3LIN-DELETION HARDNESS). *For every α and some $\beta > 0$ the following holds. Given a MAX-3LIN-DELETION \mathcal{D} instance of size n it is NP-hard to distinguish between the following two cases.*

- **Yes:** There is a $(\frac{1}{n^{\alpha\gamma}})$ -fraction of equations whose removal makes \mathcal{D} satisfiable.
- **No:** At least β -fraction of the equations in \mathcal{D} need to be removed from it to make it satisfiable.

Proof. An alternate view of Corollary 10.12.2. \square

Corollary 10.12.4 (3SAT HARDNESS). *For every $\alpha > 0$ and some $\beta > 0$, given a 3SAT instance Υ of size n it is NP-hard to distinguish between the following two cases.*

- **Yes:** There is an assignment which satisfies $(1 - \frac{1}{n^{\alpha\gamma}})$ -fraction of Υ .
- **No:** Every assignment can satisfy at most $(\frac{7}{8} + \frac{1}{(\log n)^\beta})$ -fraction of Υ .

11 Nearest Codeword Problem

The Nearest Codeword Problem (NCP) (also known as the maximum likelihood decoding problem) is the following. The input instance consists of a generator matrix $\mathbf{A} \in \mathbb{F}_2^{k \times n}$ and a received word $\mathbf{x} \in \mathbb{F}_2^n$ and the goal is to find the nearest codeword $\mathbf{y} \in \mathcal{A}$ to \mathbf{x} . One relaxation that received attention was estimating the minimum error weight to the nearest codeword. In other words, the distance $d(\mathbf{x}, \mathbf{y})$ to the nearest codeword, without necessarily finding the codeword \mathbf{y} .

Prior Results. Berlekamp, McEliece and van Tilborg [BMvT78] showed that even the relaxed version is NP-hard. Arora *et al.* [ABSS93] established that the error weight is hard to approximate to any constant is NP-hard and $2^{\log^{(1-\epsilon)} n}$ for any $\epsilon > 0$ under $\text{NP} \not\subseteq \text{DTIME}(2^{\text{polylog}})$. This was improved to inapproximability to within $n^{1/\mathcal{O}(\log \log n)}$ under $\text{P} \neq \text{NP}$ by Dinur *et al.* [DKRS03]. On the algorithmic front, Berman and Karpinski [BK02] gave a randomized algorithm for $\epsilon \cdot n / \log n$ approximation, for any fixed $\epsilon > 0$ and $\epsilon \cdot n$ approximation in deterministic time.

In the gap version of nearest codeword problem (NCP), we are given a linear code \mathcal{A} , target vector \mathbf{v} and an integer t with a promise that either the Hamming distance of \mathbf{v} to \mathcal{A} is *less than* $\gamma \cdot t$ or greater than t , one must decide which of them is true. In this section, we show that for the nearest codeword problem of any linear code over a *binary* alphabet cannot be approximated within n^α , for some $\alpha > 0$ unless $\text{P} = \text{NP}$. Before we establish the theorems, we shall define the corresponding promise problems which capture the hardness of approximating the aforementioned problems within a factor of γ .

Definition 14 (Nearest Codeword Problem). *For $\gamma \geq 1$ and $t < 1$, an instance of GAPNCP_γ is a triplet $(\mathbf{A}, \mathbf{v}, t)$, where $\mathbf{A} \in \mathbb{F}_2^{k \times n}$, $\mathbf{v} \in \mathbb{F}_2^n$ and $t \in \mathbb{Z}^+$, such that*

- (\mathbf{A}, d) is a **Yes** instance if $d(\mathbf{A}, \mathbf{v}) \leq \gamma t$.

- (\mathbf{A}, d) is a No instance if $d(\mathcal{A}, \mathbf{v}) > t$.

We begin by recollecting Corollary 10.12.3.

Corollary (MIN-3LIN-DELETION HARDNESS). *For some $\alpha, \beta > 0$ and MAX-3LIN-DELETION \mathcal{D} instance of size n it is NP-hard to distinguish between the following two cases.*

- There exists a $\left(\frac{1}{n^\alpha}\right)$ -fraction of equations whose removal makes \mathcal{D} satisfiable.
- One needs to delete more than β -fraction of the equations in \mathcal{D} to make it satisfiable.

We are now ready to prove the hardness of approximation of $\text{GAPNCP}_{n^{-\beta}}$.

Theorem 11.1. *There exists an absolute constant $\beta > 0$ such that it is NP-hard to approximate $\text{GAPNCP}_{n^{-\beta}}$.*

Proof. We reduce $\text{MIN-3LIN-DELETION}_{\frac{1}{n^\beta}, \frac{1}{2}}$ to $\text{GAPNCP}_{n^{-\beta}}$. Given an instance $\mathcal{L} \equiv \mathbf{AX} + \mathbf{y} = \mathbf{0}$ of $\text{MIN-3LINDEL}_{\frac{1}{n^\beta}, \frac{1}{2}}$, we use the $n^{-\beta}$ approximation scheme for GAPNCP available to us on $(\mathbf{A}, \mathbf{y}, 1/2)$.

- **Yes Case:** Here the deletion value of the instance is less than $1/n^\beta$. Thus, number of unsatisfied equations is at most $n^{-\beta}$. So, Hamming weight of $\mathbf{AX} + \mathbf{y}$ is at most $n^{-\beta}$. Invoking Proposition 2.2, we infer that the Hamming distance between $(\mathbf{AX}, \mathbf{y})$ is at most $n^{-\beta}$. Since, we have an access to $\mathcal{O}(n^{-\beta})$ approximation algorithm to estimate the nearest codeword of any linear code. For the parameters, the algorithm must accept $(\mathbf{AX}, \mathbf{y}, 1/2)$.
- **No Case:** In this case, more than $1/2$ constraints are left unsatisfied by every assignment. Thus, the Hamming distance between $(\mathbf{AX}, \mathbf{y})$ is strictly greater than $1/2$ and the approximation algorithm is forced to reject the instance $(\mathbf{AX}, \mathbf{y}, 1/2)$.

This completes the reduction. □

12 Concluding Remarks

In this paper, we have constructed PCP's with linear predicates that simultaneously achieve polynomially small completeness error and poly logarithmic soundness error. We have also identified some immediate applications to our new constructions. We are sure that the theory of completeness amplification will find more applicability elsewhere in near future.

References

- [AB09] S. Arora and B. Barak. *Computational Complexity - A Modern Approach*. Cambridge University Press, 2009. 15
- [ABS10] S. Arora, B. Barak, and D. Steurer. Subexponential algorithms for unique games and related problems. In *Proc. 51st IEEE Symp. on Foundations of Computer Science*, 2010. 3
- [ABSS93] S. Arora, L. Babai, J. Stern, and Z. Sweedyk. The hardness of approximate optima in lattices, codes, and systems of linear equations. *Foundations of Computer Science, IEEE Annual Symposium on*, 0:724–733, 1993. 4, 35

- [AKS87] M. Ajtai, J. Komlós, and E. Szemerédi. Deterministic simulation in logspace. In *Proceedings of the nineteenth annual ACM symposium on Theory of computing*, STOC '87, pages 132–140, New York, NY, USA, 1987. ACM. 18
- [ALM⁺98] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy. Proof verification and the hardness of approximation problems. *Journal of the ACM*, 45(3):501–555, 1998. 2
- [AS98] S. Arora and S. Safra. Probabilistic checking of proofs: a new characterization of NP. *Journal of the ACM*, 45(1):70–122, 1998. 2
- [AS03] S. Arora and M. Sudan. Improved low-degree testing and its applications. *Combinatorica*, 23(3):365–426, 2003. 2
- [BBC⁺01] B. Bollobás, C. Borgs, J. Chayes, J. H. Kim, and D. B. Wilson. The scaling window of the 2-SAT transition. *Random Struct. Algorithms*, 18(3):201–256, May 2001. 3
- [BFL91] L. Babai, L. Fortnow, and C. Lund. Nondeterministic exponential time has two-prover interactive protocols. *Computational Complexity*, 1:3–40, 1991. 2
- [BFLS91] L. Babai, L. Fortnow, L. A. Levin, and M. Szegedy. Checking computations in poly-logarithmic time. In *Proc. 23rd ACM Symp. on Theory of Computing*, pages 21–32, 1991. 2
- [BGS98] M. Bellare, O. Goldreich, and M. Sudan. Free bits, PCPs, and nonapproximability—towards tight results. *SIAM Journal on Computing*, 27(3):804–915, 1998. 3, 15
- [BK02] Piotr Berman and Marek Karpinski. Approximating minimum unsatisfiability of linear equations. In *SODA*, pages 514–516, 2002. 4, 35
- [BMvT78] E. Berlekamp, R. McEliece, and H. van Tilborg. On the inherent intractability of certain coding problems (corresp.). *Information Theory, IEEE Transactions on*, 24(3):384–386, may 1978. 35
- [BSGH⁺06] E. Ben-Sasson, O. Goldreich, P. Harsha, M. Sudan, and S. Vadhan. Robust PCPs of proximity, shorter PCPs, and applications to coding. *SIAM Journal on Computing*, 36(4):889–974, 2006. 13, 21, 29
- [CKW09] M. Cygan, L. Kowalik, and M. Wykurz. Exponential-time approximation of weighted set cover. *Inf. Process. Lett.*, 109(16):957–961, 2009. 3
- [DFK⁺99] I. Dinur, E. Fischer, G. Kindler, R. Raz, and S. Safra. PCP characterizations of NP: Towards a polynomially-small error-probability. In *Proc. 31st ACM Symp. on Theory of Computing*, pages 29–40, 1999. 2
- [DG10] I. Dinur and E. Goldenberg. The structure of winning strategies in parallel repetition games. In *APPROX-RANDOM 2010*, 2010. 6
- [DH09] I. Dinur and P. Harsha. Composition of low-error 2-query pcps using decodable pcps. In *FOCS*, pages 472–481, 2009. 13, 21
- [Din07] I. Dinur. The PCP theorem by gap amplification. *Journal of the ACM*, 54(3):12, 2007. 15

- [Din10] I. Dinur. Probabilistically checkable proofs and codes. Technical report, Plenary Talk at International Congress of Mathematicians (ICM), 2010. [2](#)
- [DKRS03] Irit Dinur, Guy Kindler, Ran Raz, and Shmuel Safra. Approximating cvp to within almost-polynomial factors is np-hard. *Combinatorica*, 23(2):205–243, 2003. [4](#), [35](#)
- [DM11] I. Dinur and O. Meir. Derandomized parallel repetition via structured pcps. *Computational Complexity*, 20(2):207–327, 2011. [2](#)
- [DR06] I. Dinur and O. Reingold. Assignment testers: Towards a combinatorial proof of the PCP theorem. *SIAM Journal on Computing*, 36(4):975–1024, 2006. [13](#), [21](#), [29](#), [30](#)
- [FGL⁺96] U. Feige, S. Goldwasser, L. Lovasz, S. Safra, and M. Szegedy. Interactive proofs and the hardness of approximating cliques. *Journal of the ACM*, 43(2):68–292, 1996. [2](#)
- [FKN02] Ehud Friedgut, Gil Kalai, and Assaf Naor. Boolean functions whose fourier transform is concentrated on the first two levels. *Adv. in Appl. Math.*, 29:427–437, 2002. [14](#)
- [Hås00] J. Håstad. On bounded occurrence constraint satisfaction. *Inf. Process. Lett.*, 74(1-2):1–6, 2000. [4](#)
- [Hås01] J. Håstad. Some optimal inapproximability results. *Journal of the ACM*, 48(4):798–859, 2001. [3](#), [11](#)
- [IKW09] R. Impagliazzo, V. Kabanets, and A. Wigderson. New direct-product testers and 2-queryPCPs. In *Proceedings of the 41st annual ACM symposium on Theory of computing*, STOC '09, pages 131–140, 2009. [2](#), [6](#)
- [Kho01] S. Khot. Improved inapproximability results for maxclique, chromatic number and approximate graph coloring. In *Proc. 42nd IEEE Symp. on Foundations of Computer Science*, pages 600–609, 2001. [2](#), [11](#), [27](#)
- [Kho02] S. Khot. On the power of unique 2-prover 1-round games. In *Proc. 34th ACM Symp. on Theory of Computing*, pages 767–775, 2002. [2](#)
- [Kho05] S. Khot. Guest column: inapproximability results via long code based PCPs. *SIGACT News*, 36(2):25–42, 2005. [25](#)
- [Kho10] S. Khot. On the unique games conjecture. In *2010 IEEE 25th Annual Conference on Computational Complexity (CCC)*, pages 99 – 121, 2010. [2](#)
- [KP06] S. Khot and A. K. Ponnuswami. Better inapproximability results for maxclique, chromatic number and min-3lin-deletion. In *Proc. Automata, Languages and Programming, 33rd International Colloquium*, pages 226–237, 2006. [4](#)
- [LFKN90] C. Lund, L. Fortnow, H. J. Karloff, and N. Nisan. Algebraic methods for interactive proof systems. In *IEEE Symposium on Foundations of Computer Science*, pages 2–10, 1990. [2](#)
- [Mos12] D. Moshkovitz. The projection games conjecture and the NP-hardness of lnn-approximating set-cover. In *APPROX'12*, 2012. [3](#)
- [MR10] D. Moshkovitz and R. Raz. Two query PCP with sub-constant error. *Journal of the ACM*, 57(5):1–29, 2010. [2](#), [3](#), [11](#), [13](#), [22](#), [24](#), [34](#)

- [Raz98] R. Raz. A parallel repetition theorem. In *SIAM Journal on Computing*, volume 27, pages 763–803, 1998. 2, 3, 5
- [RS96] R. Rubinfeld and M. Sudan. Robust characterizations of polynomials with applications to program testing. *SIAM Journal on Computing*, 25(2):252–271, 1996. 2
- [RS97] R. Raz and S. Safra. A sub-constant error-probability low-degree test and a sub-constant error-probability PCP characterization of NP. In *Proc. 29th ACM Symp. on Theory of Computing*, pages 475–484, 1997. 2
- [Sze99] Mario Szegedy. Many-valued logics and holographic proofs. In *ICALP*, pages 676–686, 1999. 13