XOR Codes and Sparse Learning Parity with Noise

Andrej Bogdanov* Manuel Sabin† Prashant Nalini Vasudevan‡

Abstract

A k-LIN instance is a system of m equations over n variables of the form $s_{i_1} + \cdots + s_{i_k} = 0$ or 1 modulo 2 (each involving k variables). We consider two distributions on instances in which the variables are chosen independently and uniformly but the right-hand sides are different. In a noisy planted instance, the right-hand side is obtained by evaluating the system on a random planted solution and adding independent noise with some constant bias to each equation; whereas in a random instance, the right-hand side is uniformly random. Alekhnovich (FOCS 2003) conjectured that the two are hard to distinguish when $k = 3$ and $m = O(n)$.

We give a sample-efficient reduction from solving noisy planted k-LIN instances (a sparse-equation version of the Learning Parity with Noise problem) to distinguishing them from random instances. Suppose that m-equation, n-variable instances of the two types are efficiently distinguishable with advantage ε. Then, we show that $O(m \cdot (m/\varepsilon)^{2/k})$-equation, n-variable noisy planted k-LIN instances are efficiently solvable with probability $\exp -\widetilde{O}((m/\varepsilon)^{6/k})$. Our solver has worse success probability but better sample complexity than Applebaum's (SICOMP 2013). We extend our techniques to show that this can generalize to (possibly non-linear) k-CSPs.

The solver is based on a new approximate local list-decoding algorithm for the k-XOR code at large distances. The k-XOR encoding of a function $F \colon \Sigma \to \{-1, 1\}$ is its k-th tensor power $F^k(x_1, \ldots, x_k) = F(x_1) \cdots F(x_k)$. Given oracle access to a function G that μ-correlates with F^k, our algorithm, say for constant k, outputs the description of a message that $\Omega(\mu^{1/k})$-correlates with F with probability $\exp(-\widetilde{O}(\mu^{-4/k}))$. Previous decoders, for such k, have a worse dependence on μ (Levin, Combinatorica 1987) or do not apply to subconstant $\mu^{1/k}$. We also prove a new XOR lemma for this parameter regime.

The decoder and its analysis rely on a new structure-versus-randomness dichotomy for *general* Boolean-valued functions over product sets, which may be of independent interest.

1 Introduction

One of the most basic examples of learning in the presence of noise is of solving a noisy system of linear equations: Given m samples $(a_i, \langle a_i, s\rangle + e_i)$ for random $a_i \in \mathbb{F}^n$ and errors e_i according to some error distribution, try to learn $s \in \mathbb{F}^n$ – i.e. given $A \in \mathbb{F}^{m\times n}$ and $As + e$, solve for $s \in \mathbb{F}^n$.

Despite its simplicity to state and the ease of solving noiseless systems of linear equations, noisy linear equations are notoriously hard and form the basis for a large body of study intersecting at learning theory, number theoretic lattices, error-correcting codes, and cryptography. That is, for different choices of $\mathbb{F}$, number of equations m, and error distributions, this problem becomes Learning With Errors (LWE) [Reg09], Learning Parity with Noise (LPN) [Pie12], and Learning With Rounding (LWR) [BPR12], all of which have been objects of study and have cryptographic primitives based on their hardness.

Learning Parity with Noise, in particular, uses $\mathbb{F}_2$ and errors, e_i, being 1 with some constant probability. Here we consider the *sparse* variant of LPN where, for some k, each row a_i has only k 1's in it. Whereas LPN has no restriction on the number of 1's in its rows and can have as many as n of them, we think of its sparse variant with k much less than n and will think of it throughout the paper even as a constant.

Besides being a natural learning problem, one motivation for considering a sparse-equation version of Learning Parity with Noise is that the assumed

*andrejb@cse.cuhk.edu.hk. Department of Computer Science and Engineering and Instutite for Theoretical Computer Science and Communications, Chinese University of Hong Kong.

†msabin@berkeley.edu. Computer Science Division, UC Berkeley. Work partially done while visiting CUHK.

‡prashvas@mit.edu. Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology. Work partially done while visiting CUHK.

hardness of many of these noisy linear equation problems are used as the basis for cryptography, yet often have unwieldy key sizes and inefficient protocols. More specifically, if we think of the matrix A as a public key in some scheme, then we need to store the entire matrix and perform expensive matrix operations. One approach to circumventing this is to use an $\mathbb{F}$ that has *considerable* algebraic structure so that rows can be "re-used" in a structured way meaning less rows need to be stored and so that A can be evaluated quickly [LPR10]. While this accomplishes the goal, it bases hardness on lattice problems where the lattice has considerably more algebraic structure that could possibly be exploited algorithmically, calling into question the hardness of the lattice problems. Another approach to achieving matrices A that are easier to store and evaluate is *sparsity*. With the sparse rows of sparse LPN, we only need to remember where the few 1's are and each row can be evaluated on input s very quickly.

Inspired by this gain in efficiency and key size, if someone were then to try to base cryptography on sparse LPN, a first step to take that is done with other noisy linear equation problems [GL89, AGS03, App17, Reg09, MM11, BLRL$^+$18, BGM$^+$16] would be to give a search-to-decision reduction for the problem: The seemingly easier decision version for noisy linear equations is to, given (A, b), distinguish whether b was given in the form of a system of equations $b = As + e$ or was just a completely random vector.

Thus search-to-decision reductions for these problems show that the one-wayness of solving noisy linear equations (i.e. finding s is hard) yields its pseudorandomness (i.e. $As + e$ is indistinguishable from random), so that these problems are good pseudorandom generators "out of the box" (without having to go through generic expensive constructions like [HILL99]).

A main parameter of study in these reductions is the blow-up in sample complexity, m, needed (see [MM11] for a thorough discussion of this). Our main result is to reduce the sample complexity needed in a search-to-decision reduction for sparse Learning Parity with Noise.

Sparse random linear equations We are given a system of k-LIN equations of the form $s_{i_1} + \cdots + s_{i_k} = 0$ or 1, where the variables in each equation and the left-hand sides of the different equations are independent and identically distributed, and the addition is modulo 2. Such a system has the form $As = b$ where A is a random $m \times n$ matrix with sparse and independent rows. We are interested in the following two problems:

- **Solving a planted instance:** Given A and $As + e$, where $s \sim \{0,1\}^n$ is a random planted solution and e is a vector of random i.i.d. $\{0,1\}$-entries where each entry is 1 with constant probability η, find s.

- **Distinguishing planted from random instances:** Distinguish the distribution $(A, As+e)$ from (A, r), where $r \sim \{0,1\}^m$ is independent of A.

The distinguishing variant was introduced by Alekhnovich [Ale11]. He conjectured that when $k = 3$ and $m = O(n)$ distinguishing with advantage substantially better than $1/n$ is intractable. This is a generalization of Feige's conjecture [Fei02] from which several hardness of approximation results are derived.

A distinguishing advantage of $\Omega(\binom{m}{2}/\binom{n}{k})$ can be attained by a simple collision-finder: The distinguisher looks for two appearances of the same equation, accepts if the right-hand sides are equal, and rejects otherwise.

Distinguishing algorithms have received considerable attention in the regime where their advantage is very close to one. A *refutation algorithm* must accept all planted instances in which the error rate is less than some threshold, say 2η, and reject almost all random instances (a refutation algorithm can be thought of strengthening of a distinguishing algorithm to be so that the error is only one-sided). Polynomial-time refutation algorithms are known for random k-XOR instances provided the number of clauses m exceeds $\omega_k(n^{k/2})$ [AOW15, BM16] and are conjectured not to exist when $m = o(n^{k/2})$ [ABW10, BM16]. In the latter regime, refutation in time $\exp \tilde{O}(n^\delta)$ is possible if $m = \tilde{\omega}_k(n^{k/2-\delta(k/2-1)})$ [RRS17].

On the negative side, Kothari et al. [KMOW17] show that the refutation algorithms of [AOW15, BM16, RRS17] are optimal among a wide class of semidefinite programs. Further, Feldman, Perkins, and Vempala [FPV15] describe a statistical model in which efficient search is possible when $m = \omega(n^{k/2} \log^2 n)$, but distinguishing isn't when $m = o((n/\log n)^{k/2})$.

For the solving variant, Applebaum [App16] describes an efficient solver in the regime $m = \omega_k(n^{k/2})$ by a reduction to a 2CSP instance, the application of a suitable approximation algorithm [GW95, CW04], and some additional post-processing work.[1]

In this paper we are interested in the *relationship* between the solving and distinguishing variants. In one direction, a solver that works on an α-fraction of instances can be used to distinguish with advantage at least $\alpha - 2^{-H(2\eta)m+n}$, indicating that solving should be harder than distinguishing, as expected. In the other direction, Applebaum [App13] gives an efficient search-to-decision reduction from solving a constant fraction of instances of size m to that of distinguishing instances of size $(\varepsilon^2 m/\log n)^{1/3}$, where ε is the advantage of the distinguisher. That is, roughly cubicly many more equations are needed to solve than to distinguish. Our work will reduce this gap.

The starting point of Applebaum's analysis, as is done successfully for the search-to-decision reductions for LPN [GL89, AGS03, App17], LWE [MM11, BLRL$^+$18], and LWR [BGM$^+$16], is an application of Yao's distinguishing-to-prediction reduction [Yao82]. In this context the reduction turns a distinguisher for planted instances with m equations and advantage ε into a predictor that guesses the value of any given k-LIN equation (or, respectively, any given linear functional over $\mathbb{F}$ corresponding to whether we instead started with (dense) LPN, LWE, or LWR) evaluated on the planted solution with advantage ε/m, given $m-1$ planted equations as "training data".

Applebaum's work was concerned with general k-Constraint Satisfaction Problems however and did not focus on k-LIN itself, and so the rest of his techniques diverged from typical search-to-decision techniques for noisy linear equations. Namely, all of LPN [GL89, AGS03, App17], LWE [MM11, BLRL$^+$18], and LWR [BGM$^+$16] treat the predictor for linear equations evaluated on s as *noisy access to a codeword of an encoding of s*. For example, LPN considers arbitrary parities with s which, looking at the string of all possible parities of s, is exactly the Hadamard encoding of s, to which the predictor is noisy access to that codeword. Thus all of these problems then become *decoding problems* of recovering s from the appropriate error-correcting code.

We wish to follow this successfully applied perspective to achieve search-to-decision for the *sparse* LPN problem. While the natural code for LPN was the Hadamard code, sparse LPN is looking at the subset of the Hadamard code corresponding to parities of exactly k positions. This is exactly the k-XOR code.

[1]This approach applies more generally to equations with adversarial noise.

XOR Codes XOR lemmas [Yao82] are statements that are typically used to relate the average-case hardness of a Boolean function $F\colon \Sigma \to \{-1,1\}$ to that of its k-XOR encoding $F^k\colon \Sigma^k \to \{-1,1\}$ given by

$$F^k(x_1,\ldots,x_k) = F(x_1)\cdots F(x_k).$$

That is, if some algorithm A of low complexity computes F on a $(1+\mu^{1/k})/2$-fraction of inputs under the uniform distribution over Σ (equivalently, we say that A $\mu^{1/k}$-correlates with F), then the algorithm $A'(x_1,\ldots,x_k) = A(x_1)\cdots A(x_k)$ computes F^k on a $(1+\mu)/2$-fraction of inputs. Thus, the ability of A to correlate with the function drops exponentially in k from $\mu^{1/k}$ to μ when used naïvely as with A'. XOR lemmas formalize the intuition that this is essentially the best possible average-case algorithm for F^k.

Although XOR lemmas are typically used for hardness amplification in computational complexity, *we will break from this use and interpretation*. More specifically, if we think of the truth table $F \in \{-1,1\}^{|\Sigma|}$ as a string where we call $N = |F|$, then we will be thinking of *extremely* small correlations[2] $\mu = 1/N^{\Omega(k)}$ which would simply not make sense in the non-uniform hardness amplification framework: achieving $(1+\mu)/2$ fraction of agreement with F^k in this regime amounts to being correct on *just a few more bits than half* of F^k's truth table, which we can never guarantee hard in the non-uniform model since you could just hardcode those very few bits into the circuit to match the small advantage required. Thus it is not useful to think of the XOR code in terms of hardness amplification or of the usual techniques used for it for that use-case. We instead look at XOR code in terms of coding theory.

In coding-theoretic language, XOR lemmas are approximate local list-decoding algorithms for the k-XOR code [STV01]. The function F represents a binary message of length $|\Sigma|$ and F^k is its encoding. List-decoding is the task of finding all the codewords F^k that have relative agreement at least $(1+\mu)/2$ with a given corrupted codeword $G\colon \Sigma^k \to \{-1,1\}$.

[2]We care about such small advantage since, in the sparse LPN case, we think of F as our secret s of size N that we want to recover from $M > N$ noisy parities of it. Thus, as mentioned earlier, a sparse LPN *distinguisher* with advantage ε can be converted to a predictor with advantage $\varepsilon/M < \varepsilon/N$. This small advantage necessitates our extreme parameterization of the XOR code for our use-case.

For smaller μ (that we consider here) the number of such codewords can be exponentially large in $|\Sigma|$ which severely restricts the utility of exact list-decoding.[3] It is common to study the following relaxation.

Definition 1.1. *Let $\mu, \alpha \in [0,1]$ be parameters. A binary code is* $(1+\alpha)/2$-approximately list-decodable *for error rate* $(1-\mu)/2$ *with* list size $\ell(\mu,\alpha)$ *if for every corrupted codeword G there exists a list of codewords $C_1,\dots,C_\ell$ such that for any codeword C that agrees with G on at least a $(1+\mu)/2$-fraction of positions, C also agrees on a $(1+\alpha)/2$-fraction with C_i for some i.*

That is, for any codeword that is corrupted on at most a $(1-\mu)/2$-fraction of positions, we want a list that might not contain the original codeword itself but at least contains a codeword *close* to the original one (i.e. agrees on a $(1+\alpha)/2$-fraction). This relaxation may allow for smaller (and non-trivial) list size.

Remark 1.1. *We talk in terms of recovering a list of approximate codewords instead of trying to approximately recover the message since, in the case of the k-XOR code, two codewords F^k, F'^k are at distance $(1-\alpha)/2$ if and only if the corresponding messages F, F' are at distance $(1-\alpha^{1/k})/2$. Thus, we will freely interchange between talking about decoding to codewords versus messages.*[4]

When $\alpha > \mu$ the list size is still exponential in Σ (as we'll show in Proposition 4.2), so the regime of interest is $\alpha \le \mu$. While modern XOR lemmas have a suite of impressive and desirable properties, such being simple, derandomized, uniform, and having extremely small list sizes such as $O(1/\mu^2)$ [IJKW10], almost all XOR lemmas [Imp95a, GNW11, IJK09, IJKW10] have the constraint that the *approximation error* $(1-\alpha^{1/k})/2$ is at least $\Omega((\log 1/\mu)/k)$. Thus, all of them can only talk about smaller μ when k is large and do not address the regime in which k is smaller than $\log 1/\mu$ – e.g. if k is constant, like we're concerned with, then so must be α and μ, and so smaller correlations/noisier codewords can't be discussed. While this makes sense for being applied to hardness amplification where k can be increased as desired, our use needs *much* smaller μ to be discussed in the regime $\mu = o(2^{-k})$ and even $\mu = o(1/|\Sigma|)$, even while k is constant.

To address such extreme parameters we stray from modern XOR lemmas and look to the first written proof of the XOR lemma [Lev87] which, as a notable exception to all other XOR lemmas, achieves near optimal approximation $\alpha^{1/k} = \mu^{1/k} - \varepsilon$ for arbitrary $\varepsilon > 0$ with no constraints. Unfortunately its list size grows at least exponentially in $1/\mu^2$ but it is a main lemma of our work to improve this to exponential in $1/\mu^{4/k}$.

1.1 Our results

Our first result, which we will use to prove our search-to-decision reduction of sparse LPN, is a new approximate local list-decoding algorithm for the k-XOR code at very large distances. We define correlation between two functions A and B using the notation $\mathrm{E}[A \cdot B]$ for the product of $A(x)$ and $B(x)$ averaged over their inputs, i.e., $\mathrm{E}[A\cdot B] = \mathrm{E}_x[A(x)\cdot B(x)]$. Further, we will talk in terms of α-correlating for codewords and $\alpha^{1/k}$-correlating with messages interchangeably as per Remark 1.1.

Definition 1.2. *A* $(\mu, \alpha^{1/k})$ approximate list-decoder *with success probability p for the k-XOR code is an algorithm that, given a corrupted codeword G such that $\mathrm{E}[G\cdot F^k] \ge \mu$ for some message F, outputs a message $\hat{F}$ such that $\mathrm{E}[F\cdot \hat{F}] \ge \alpha^{1/k}$ with probability p. The implicit list size is $O(1/p)$.*

The approximate list-decoder is *local* if G is provided as an oracle, and its output $\hat{F}$ is a circuit (which on input x calculates $\hat{F}(x)$), and *uniform* if its dependence on the parameters μ, $|\Sigma|$, k, and α is uniform (that is, it uses no non-uniform advice).

We now state a simple version of the theorem we attain for achieving an approximate local list-decoder for the k-XOR code where we think of k as constant and $\mu = 1/n^{O(1)}$, where $n = |F|$ is the message size. A more generally parameterized and more carefully quantified full Theorem 2.1 in Section 2.

Informal Theorem 1.1. *There is a uniform local $(\mu, \Omega(\mu^{1/k}))$-approximate list decoder for the k-XOR code that succeeds with probability at least $2^{-\tilde{O}(1/\mu^{4/k})}$ and runs in time $\tilde{O}(\mu^{-4})$.*

In contrast, for constant k, most other XOR lemmas [Imp95a, GNW11, IJK09, IJKW10] cannot address our small $\mu = 1/n^{O(1)}$ (nor does it make sense

[3]More precisely, the list can be of size $2^{h((1-\mu^{1/k})/2)|\Sigma|}$, where h is the binary entropy function.

[4]This holds since, for $\{-1,1\}$-valued functions, the k-XOR code F is simply the tensor product $F^{\otimes k}$ and since $\langle F^{\otimes k}, F'^{\otimes k}\rangle = \langle F, F'\rangle^k$ (we'll see correlation defined in terms of inner product/expectation later).

for them to in the use-case of hardness amplification). Further, the algorithm implicit in Levin's XOR lemma [Lev87], which we derandomize, succeeds with probability inverse exponential in μ^{-2} whereas ours is inverse exponential in $\mu^{-4/k}$. For concreteness, Levin can technically address $\mu = 1/n^{O(k)}$ but only to attain a more than trivial list size $2^{1/\mu^2} = 2^{n^{O(k)}}$, whereas for $\mu = 1/n^{k/5}$ we can achieve a non-trivial list size $2^{1/\mu^{4/k}} = 2^{n^{4/5}}$.

We apply Informal Theorem 1.1 to derive the following informal statement of our search-to-decision reduction for sparse noisy linear equations where we think of k and advantage ε as a constant. A more carefully quantified version of the full Theorem 3.1 can be found in Section 3.

Informal Theorem 1.2. *Suppose that m-equation, n-variable planted η-noisy kLIN instances are distinguishable from random ones in time t with advantage ε, where $\eta < 1/2$. Then, planted instances with $O(m^{1+2/k})$ equations and n variables can be solved in time polynomial in t, m, and n, and $1/\varepsilon$, with probability at least $2^{-\tilde{O}(m^{6/k})}$ over the choice of the instance and the randomness of the solver.*

In contrast, the solver in Applebaum's reduction requires more than m^3 equations but succeeds with constant probability. It is possible to obtain other tradeoffs between the sample complexity and the success probability of the solver.

In Section 3.1 we show how this search-to-decision reduction can be generalized to (possibly non-linear) k-Constraint Satisfaction Problems (CSPs) other than the k-LIN one.

By combining Theorem 1.2 with the refutation algorithm of Raghavendra, Rao, and Schramm, it follows that for constant k and constant noise rate, for every constant $2/3 < \delta < 1$, m-equation, n-variable planted noisy k-LIN instances can be solved with probability $2^{-\tilde{O}(m^{6/k})}$ in time $2^{\tilde{O}(n^{\delta})}$ as long as $m = \tilde{\Omega}(n^{(1-\delta)k/2+1+2\delta/k})$.[5]

Other consequences As a corollary of (the more thoroughly quantified) Theorem 2.1 we obtain an upper bound on the list size for k-XOR codes at high error rates.

Corollary 1.1. *For $\alpha = (\mu^{1/k} - \varepsilon)^k$, $\ell(\mu, \alpha) = O(|\Sigma|/\varepsilon)^{O(k^2/\mu^{2/k}\varepsilon^2)}$.*

[5]It has been pointed out by an anonymous reviewer that for sums-of-squares refutation algorithms the solution can also be recovered by the method of Barak, Kelner, and Steurer [BKS14].

The value of α in Corollary 1.1 is close to optimal. In Section 4 we will show that when $\alpha > \mu$ the list size becomes exponential in $|\Sigma|$ (see Proposition 4.2). We do not know, however, if the list size has to be exponentially large in $\mu^{\Theta(1/k)}$ when $\alpha \le \mu$. Further, Proposition 4.3 will prove the lower bound $\ell = \Omega(\alpha^{2/k}\mu^{-2})$ for all α, assuming $\mu \ge |\Sigma|^{-1/2}$. Also in Section 4, Proposition 4.1 gives a much tighter non-constructive upper bound when $\alpha < \mu^2$.

We also obtain the following consequence for non-uniform hardness amplification in the low-correlation regime. The following corollary (proved in Section 2.4) improves the amount of advice in Levin's proof [Lev87] from linear in μ^{-2} to linear in $\mu^{-O(1/k)}$.

Corollary 1.2. *There is a log-time uniform oracle circuit L with $O(k\log(1/\varepsilon)(n+\mu^{-2/k}\varepsilon^{-2}))$ bits of advice and size $\tilde{O}(k^k\mu^{-2}\varepsilon^{-2k})$ such that, if G predicts F^k with advantage μ, then for some setting of the advice L^G predicts $F\colon \{0,1\}^n \to \{-1,1\}$ with advantage at least $\mu^{1/k} - \varepsilon$.*

1.2 Techniques for list-decoding the XOR code

Our proof of Theorem 1.1 is a derandomization of Levin's proof [Lev87] (see also [GNW11]). We begin with a short outline of his proof and point out its limitations with respect to list size. This motivates the two main innovations introduced in our work: A new notion of regularity for functions over product sets, and an analysis of a natural sampler for regular functions.

Remark 1.2. *It is worth noting that the "derandomization" of XOR lemmas has a long history [Imp95b, IW97, IJKW10], however we mean something qualitatively different when we talk about derandomizing our XOR lemma. Namely, all other derandomizations do so in the* forward direction *of the encoding process; that is, they derandomize the* encoding *process so that, rather than looking at all possible k-tuples of indices of F to XOR, they generate the k-tuples pseudorandomly so that less of them are considered and the resulting codeword is smaller. This is typically done for better hardness amplification guarantees since a smaller codeword means that it is the truth table over a smaller input size and so the hardness guarantees suffer less of a parameter loss in the transformation of input sizes from message to codeword. In contrast, our derandomization is of the* backwards direction *of the encoding process; that is, we derandomize the* decoding *process so that, instead of taking random samples for which we need advice,* *we generate the samples we need advice for* pseudorandomly *in order to decrease the list size we result with. This can say something in terms of hardness amplification as well since we require less advice and so lose less in the circuit size degradation of a hardness amplification, but this is uninteresting in light of the extremely small list sizes (and thus uniformity) of works like [IJKW10]. We instead care about decoding Levin's proof so that we can consider extremely small μ (which would not be useful for hardness amplification anyway) while still shaving* some *advice/list-size off.*

In the ensuing discussion we ignore the locality of the list-decoder, so the concepts are introduced in less general form than in Section 2.

Levin's XOR lemma Here is an outline of Levin's proof for $k = 2$. The correlation assumption $\mathrm{E}[G \cdot F^2] \geq \mu$ can be written in the form

$$\mathrm{E}_x\left[F(x) \cdot \mathrm{E}_y[F(y)G(x,y)]\right] \geq \mu.$$

One case is that the inner expectation is at least $\sqrt{\mu}$ in absolute value for some x. Then the column function $G_x(y) = G(x,y)$ or its negation predicts F with advantage $\sqrt{\mu}$ and we're already done. Otherwise, all inner expectations are bounded by $\sqrt{\mu}$ in magnitude. Then the function $\mu^{-1/2}\tilde{F}$, where

$$\tilde{F}(x) = \mathrm{E}_y[F(y)G(x,y)]$$

is $[-1,1]$-bounded and predicts F with advantage $\sqrt{\mu}$. With constant probability, the function $\mu^{-1/2}\tilde{F}$ can be pointwise estimated to within ε or, equivalently, $\tilde{F}$ can be estimated to precision $\varepsilon\mu^{1/2}$ from $\tilde{\Theta}(1/\varepsilon^2\mu)$ samples $F(y)G(x,y)$ for random y via a Hoeffding or Chebyshev bound (although a Chebyshev bound will be more appropriate to think of by the time we begin derandomizing this algorithm). Then F can be predicted with advantage $\sqrt{\mu} - \varepsilon$ given $\tilde{O}(1/\varepsilon^2\mu)$ pairs $(y, F(y))$ as advice.

More generally, given a correlation assumption of the form $\mathrm{E}[A(x)B(y)G(x,y)] \geq \alpha\beta$, either some column of G predicts B up to sign with advantage β, or else the empirical average $\beta^{-1}\,\mathrm{E}[G(x,y)B(y)]$ taken over $\tilde{\Theta}(1/\varepsilon^2\beta^2)$ samples usually predicts A with advantage $\alpha - \varepsilon$. Since ε must be less than α, the number of required samples grows at least quadratically in the inverse of the advantage $1/\alpha\beta$.

Levin's k-XOR lemma is proved by applying this proposition inductively. By setting $A = F^i, \alpha = \mu^{i/k}$ and $B = F^{k-i}, \beta = \mu^{(k-i)/k}$, proving a k-XOR lemma is reduced to proving an i-XOR lemma and a $(k-i)$-XOR lemma. Even though different choices of the parameter i lead to different proofs, the resulting decoder always requires at least $\tilde{\Omega}(1/\alpha^2\beta^2) = \tilde{\Theta}(1/\mu^2)$ values of F as advice. The decoding we achieve in this paper only needs $\tilde{\Theta}(1/\mu^{4/k})$ values. For $k = 2$ we match Levin's algorithm and do substantially better as k increases.

Our derandomized XOR lemma We illustrate our improvement on the list size for the 3-XOR code, for which Informal Theorem 2.1 gives a list of size exponential in $\tilde{O}(1/\mu^{4/3})$, which improves upon Levin's exponent of $\tilde{\Theta}(1/\mu^2)$.

Assume $\mathrm{E}[F(x)F(y)F(z)G(x,y,z)] \geq \mu$. In case $\mathrm{E}[F(y)F(z)G(x,y,z)]$ is at least $\mu^{2/3}$ in magnitude for some x, we apply Levin's 2-XOR lemma to the function $G_x(y,z) = G(x,y,z)$ to obtain a list size exponential in $\tilde{O}(1/(\mu^{2/3})^2)$ and we're done. Otherwise, we may assume that the function

$$\tilde{F}(x) = \mathrm{E}_{y,z}[H_x(y,z)],$$

(where $H_x(y,z) = F(y)F(z)G_x(y,z)$) is bounded in magnitude by $\mu^{2/3}$ for all x. Levin's proof proceeds by estimating $\mu^{-2/3}\tilde{F}$ pointwise with precision, say, $\varepsilon = \mu^{1/3}/2$, or equivalently estimating $\tilde{F}$ pointwise with precision $\mu/2$. This requires $\tilde{\Theta}(1/\mu^2)$ samples, and thus advice, of the form $F(y)F(z)$ coming from a set S of *independent* random pairs (y,z).

The source of our improvement in list size is an emulation of the random set S by a (small number of) random *product* set(s) $S_Y \times S_Z$ of comparable size, such that $|S_Y| = |S_Z|$. Since the advice we want is essentially the values of the 2-XOR code $F^2(y,z)$ on tuples (y,z) from say S or $S_Y \times S_Z$, then knowing $F(y)$ and $F(z)$ is enough to reconstruct $F^2(y,z) = F(y)F(z)$ by definition of the XOR code; thus, receiving just $F(y)$ for all $y \in S_y$ and $F(z)$ for all $z \in S_z$ is enough to reconstruct all of $F(y)F(z)$ for all $(y,z) \in S_y \times S_z$. This significantly reduces our advices since $|S| = |S_Y \times S_Z|$ will mean that $|S_Y| = |S_Z| = |S|^{1/2}$ and so we reduce the amount of advice bits quadratically.[6] Since the list size is exponential in the advice length, this effectively reduces it from from $\exp(|S|)$ bits to $\exp(|S_Y| + |S_Z|) = \exp(|S|^{1/2})$ bits.

Unfortunately though, random product sets may be poor samplers in general. For example, if $H_x(y,z)$

[6]For the k-XOR code, these product spaces will become become sub-cubes of the k-dimensional boolean cube, and so only requiring advice along its borders gives us our k^{th} root in savings in advice we achieve.

happens to be a dictator in y (i.e. independent of z), then a random sample of size s^2 would produce a precision of a $\Theta(1/s)$-biased estimate. Yet a random product sample, which would only depend on S_y in this case, of the same size would yield a $\Theta(1/\sqrt{s})$-biased estimate since $|S_y| = (s^2)^{1/2} = s$; to achieve the same $\Theta(1/s)$-bias a random sampler achieves, we would need $|S(y)| = s^2$ as well, wiping away any potential savings. On the other hand, this unfortunate example of a dictator H is so bad since$|\mathrm{E}_z[H_x(y_0, z)]|$ equals one for any dictator value y_0, so $G_x(y_0, z)$ or its negation is an exact decoding of $F(z)$ and so we have a decoding anyways.

We show that this phenomenon is true in general: Either a random product sampler is a good pseudorandom generator for our decoding algorithm or it fails because H was so structured that it could have been decoded easily anyways. Said in a slightly more formal way, our approximate list-decoder for the k-XOR code is based on a *structure versus randomness dichotomy*: Either the function H_x is "regular", in which case the product sampler accurately emulates a truly random sampler, or else one of the rows or columns of H_x is "structured", in which case the problem reduces to approximately list-decoding the $(k-1)$-XOR code.

Sampling regular functions Let $H(y, z)$ be a function with $|\mathrm{E}[H]| = \mu^{2/3}$. We call H *regular* if all rows and columns of H are pseudorandom in the sense that $|\mathrm{E}_y[H(y, z)]| \leq \mu^{1/3}$ for all z and $|\mathrm{E}_z[H(y, z)]| \leq \mu^{1/3}$ for all y. If one of the functions H_x is not regular, then one of the columns or rows of G_x already predicts F with advantage $\mu^{1/3}$ up to sign.

Our main technical result is Lemma 2.1, which shows that if H is regular, then a product sampler of complexity $|S_Y| = |S_Z| = \tilde{O}(\mu^{-4/3})$ estimates $\mathrm{E}[H]$ to within $\mu/2$ with constant probability. If all H_x are regular then F can be predicted with $\tilde{O}(\mu^{-4/3})$ bits of advice, giving the desired list size.

The product sampler is an unbiased estimator of $\mathrm{E}[H]$. Lemma 2.1 is proved by upper bounding its variance for regular functions by $o(\mu^2)$. This amounts to comparing the bias of the product and random samplers on a typical pair of samples (y, z) and (y', z'). The only difference is that the pairs (y, y') and (z, z') have a higher collision probability in the product sampler. Conditioned on neither of these pairs colliding, (y, z) and (y', z') are identically distributed for both samplers.

For the variance analysis, the product sampler is therefore modeled by the following process: With probability $1 - o(\mu^{4/3})$ emulate the random sampler, with probability $o(\mu^{4/3})$ fix $y = y'$ to a random value and emulate the random sampler for the function $H_y(z) = H(y, z)$, with probability $o(\mu^{4/3})$ do the same with the roles of the two coordinates reversed, and with probability $o(\mu^{8/3})$ fix both $y = y'$ and $z = z'$ to random values and output the constant $H_{yz} = H(y, z)$. By the regularity assumption, each of these cases contributes $o(\mu^2)$ to the variance, giving the desired conclusion.

1.3 Techniques for hardness versus randomness of noisy linear equations The proof of Theorem 1.2 is based on the paradigm of Goldreich and Levin [GL89] for converting hardness into pseudorandomness in cryptographic settings. Yao's reduction [Yao82] is first applied to convert the distinguisher into a predictor. The truth-table of the predictor is then viewed as a corrupted codeword with respect to a suitable encoding of the planted solution. A decoding algorithm is then used to recover the solution.

In the setting of noisy random k-LIN instances, the predictor is a function that, given $m-1$ equations from the planted distribution as "training data", produces a guess for the value of the m-th equation. Given good training data, the truth-table of the predictor is therefore a corrupted codeword of the k-XOR code. A distinguisher with advantage ε yields a predictor with advantage $\mu = \varepsilon/(1 - 2\eta)m$ in expectation over the choice of the training data. This step of the reduction is also carried out (in greater generality) in the work of Applebaum [App13]. He then amplifies the advantage of the predictor by using independent samples of the training data up to the point where the solution can be uniquely extracted.

To avoid the increase in sample complexity, we instead apply our list-decoding algorithm for the k-XOR code to the predictor. With noticeable probability, the list-decoder outputs an approximate solution $\hat{s}$ that $\mu^{1/k}/2$-correlates with the planted solution s. In other words, the output of the list-decoder predicts the value of s_i for a random index i with advantage $\mu^{1/k}/2$. Our main insight is that, owing to the symmetries of the k-LIN instance, the same advantage can be attained for an arbitrary i, which allows the advantage to be amplified by repetition (this is thoroughly explored in the full version of the paper). Once it is sufficiently large, the solution can be extracted using a technique of

Bogdanov and Qiao [BQ12].

In our search-to-decision reduction, the list size of the k-XOR decoder governs the advantage, while the approximation quality α affects the sample complexity. As our value of α is close to optimal by Proposition 4.2 the sample complexity blowup appears to be inherent in our method. On the other hand, better bounds on list size would yield a corresponding improvement in the advantage of the reduction.

2 Approximately List-Decoding the XOR Code

In this section we prove state a more general and more thoroughly quantified version of Informal Theorem 1.1. Namely, we will prove the follow theorem.

Theorem 2.1. *There is a uniform local $(\mu, \mu^{1/k} - \varepsilon)$-approximate list decoder for the k-XOR code that succeeds with probability at least $\Omega(\varepsilon)^{O(k^2/\mu^{2/k}\varepsilon^2)}$ and runs in time $\tilde{O}(k^k \mu^{-2} \varepsilon^{-2k} \log|\Sigma|)$.*

Section 2.1 introduces the notion of regularity for product functions and analyzes the variance of product samplers. Section 2.2 describes and analyzes the list-decoding algorithm for a function G under the assumption that most of the functions $G(x_1, \ldots, x_{k-1}, a)F(x_1) \cdots F(x_{k-1})$ are regular. Section 2.3 describes the list-decoder for general codewords G and proves Theorem 2.1.

2.1 Regularity and product samplers Suppose $R\colon \Sigma^k \to \{-1, 1\}$ is a random function each of whose entries are i.i.d. with some unknown bias and we are interested in estimating R's expectation up to precision μ. Chebyshev's inequality guarantees that about $1/\mu^2$ samples are sufficient to produce an accurate estimate with constant probability. When R is random, it is irrelevant how the samples are chosen as long as they are all distinct. In particular they can be chosen from a *product sample* of the form $S_1 \times \cdots \times S_k$ where $|S_1| = \cdots = |S_k| = 1/\mu^{2/k}$.

For general functions, however, product samplers produce substantially poorer estimates than random samplers of the same size. For example, if $H\colon \Sigma^k \to \{-1, 1\}$ is a dictator (that is, fully determined by one of its inputs) then the precision of the product sampler drops to $O(\mu^{1/k})$.

Regularity is a pseudorandomness property of bounded functions over product sets that guarantees the product sampler has about the same accuracy as for a random function. In this context, the crucial property of the random function turns out to be its "closure" under input restriction: If some subset I of the indices of the k inputs is restricted, the product sampler on the remaining inputs has standard deviation $\mu^{2(k-|I|)/k}$. This motivates the following definition of regularity.

Definition 2.1. *A function $H\colon \Sigma^k \to \{-1, 1\}$ is* (μ, λ)-regular *if for all nonempty $I \subseteq [k]$,*

$$\mathrm{E}[H(x_1, \ldots, x_k)H(x'_1, \ldots, x'_k) \mid x_I = x'_I] \leq \mu^2 \lambda^{-|I|}.$$

H is strongly (μ, λ)-regular *if the inequality also holds for $I = \varnothing$.*

(The notation $x_I = x'_I$ is shorthand for "$x_i = x'_i$ for all $i \in I$.")

The regularity requirement is worst-case in the sense that it must hold for all subsets of coordinates, but average-case in the sense that once the coordinates of the input variables to be restricted are fixed, the deviation need only be small on average over the choice of the restricted values (note that this average-case notion is slightly weaker than the informal notion of regularity we gave in Section 1.2 which required that this be worst-case as well).

The parameter setting that is consistent with the above discussion is $\lambda = \mu^{2/k}$. For our intended application it is convenient to allow for a small deviation from this value and so the definition is stated in this more general form.

The main result of this section is the following lemma which bounds the variance of the estimator when using the product sampler with respect to regular functions. This is the main utility of the notion of regular as bounding the variance of the estimate gotten when using the product sampler *will allow us to use the Chebyshev bound* as before to bound the number of samples required to estimate up to precision μ.

Lemma 2.1. *If H is (μ, λ)-regular, the sets $S_1, \ldots, S_k \subseteq \Sigma$ of size s each are mutually independent of each other, and the elements within each set S_i are pairwise independent, then*

$$\mathrm{Var}_{S_1,\ldots,S_k} \mathrm{E}[H(x_1, \ldots, x_k) \mid x_i \in S_i \textit{ for all } i] \leq \left(\left(1 + \frac{1}{s\lambda}\right)^k - 1\right) \cdot \mu^2.$$

An interesting setting of parameters is $\lambda = \mu^{2/k}$ and $s = O(k \cdot \mu^{-2/k})$. The variance of the product sampler is then bounded by μ^2 just like for a random function at a (small) multiplicative cost of $O(k)$ in the sizes of the sets $S_1, \ldots, S_k$.

Proof. Let $\mathbf{S} = (S_1, \dots, S_k)$, $\mathbf{x} = (x_1, \dots, x_k)$ and $\mathbf{x}' = (x'_1, \dots, x'_k)$. In this notation, the variance of interest equals

$$\begin{aligned} \operatorname{Var}_{\mathbf{S}} \mathrm{E}_{\mathbf{x}}[H(\mathbf{x}) \mid \mathbf{x} \in \mathbf{S}] \\ &= \mathrm{E}_{\mathbf{S}}\big[\mathrm{E}[H(\mathbf{x}) \mid \mathbf{x} \in \mathbf{S}]^2\big] - \mathrm{E}[H]^2 \\ &= \mathrm{E}_{\mathbf{S},\mathbf{x},\mathbf{x}'}[H(\mathbf{x})H(\mathbf{x}') \mid \mathbf{x}, \mathbf{x}' \in \mathbf{S}] - \mathrm{E}[H]^2. \qquad (2.1) \end{aligned}$$

The triples $(S_1, x_1, x'_1), \dots, (S_k, x_k, x'_k)$ in the first term are independent. Moreover, the induced marginal distribution on every pair (x_i, x'_i) is

$$(x_i, x'_i) \sim \begin{cases} \text{identical uniformly random element} \\ \quad \text{in } \Sigma\text{, with probability } 1/s, \\ \text{uniformly random pair of distinct ele-} \\ \quad \text{-ments in } \Sigma\text{, with probability } 1 - 1/s. \end{cases}$$

This distribution has the following alternative description: Flip a coin C_i with probability of heads $p = (1/s - 1/|\Sigma|)/(1 - 1/|\Sigma|) \le 1/s$ and sample

$$(x_i, x'_i) \sim \begin{cases} \text{identical uniformly random element} \\ \quad \text{in } \Sigma\text{, if } C_i \text{ is heads,} \\ \text{independent uniformly random pair in} \\ \quad \Sigma \times \Sigma\text{, if } C_i \text{ is tails.} \end{cases}$$

Letting $I \subseteq [k]$ denote the set of those i for which C_i came out heads we can write

$$\begin{aligned} &\mathrm{E}_{\mathbf{S},\mathbf{x},\mathbf{x}'}[H(\mathbf{x})H(\mathbf{x}') \mid \mathbf{x}, \mathbf{x}' \in \mathbf{S}] \\ &= \mathrm{E}_{I,\mathbf{x},\mathbf{x}'}[H(\mathbf{x})H(\mathbf{x}') \mid x_I = x'_I] \\ &= \sum_{I \subseteq [k]} p^{|I|}(1-p)^{k-|I|} \, \mathrm{E}_{\mathbf{x},\mathbf{x}'}[H(\mathbf{x})H(\mathbf{x}') \mid x_I = x'_I] \\ &\le E[H]^2 + \sum_{I \subset [k]} \left(\frac{1}{s}\right)^{|I|} \cdot \mu^2 \lambda^{-|I|}. \end{aligned}$$

Plugging into (2.1) it follows that

$$\begin{aligned} \operatorname{Var}_{\mathbf{S}} \mathrm{E}_{\mathbf{x},\mathbf{x}'}[H(\mathbf{x}) \mid \mathbf{x} \in \mathbf{S}] &\le \sum_{I \subset [k]} \left(\frac{1}{s}\right)^{|I|} \cdot \mu^2 \lambda^{-|I|} \\ &\le \mu^2 \sum_{I \subset [k]} \left(\frac{1}{\lambda s}\right)^{|I|} \\ &= \mu^2 \left(\left(1 + \frac{1}{\lambda s}\right)^k - 1 \right). \end{aligned}$$

The success probability of the sampler can be increased by taking the median run of several independent repetitions.

Repeated product sampler S^H:

```
1   Choose independent sets $S_{ij}$, $1 \le i \le k$,
        $1 \le j \le t$ of size $s$ each.
2   Output the median value of
        $\mathrm{E}[H(x) \mid x_i \in S_{ij}$ for all $i]$ among all $t$
        such values.
```

The following claim states the effectiveness of the product sampler. The additional parameter θ controls the tradeoff between the accuracy of the estimate and the product sample size and can be initially thought of as a small constant.

Claim 2.1. *Assuming H is (μ, λ)-regular, $s \ge k/\theta\lambda$, and $t \ge 8 \log 1/\eta$, with probability at least $1 - \eta$, $|S^H - \mathrm{E}[H]| \le 2\sqrt{\theta/(1-\theta)} \cdot \mu$.*

Proof. By Chebyshev's inequality, for any j, the probability that the estimator

$$E_j = \mathrm{E}[H(x) \mid x_i \in S_i \text{ for all } i]$$

deviates by more than two standard deviations from its mean $\mathrm{E}[H]$ is at most $1/4$. By Lemma 2.1 and the choice of parameters, the standard deviation is at most $\sqrt{(1+\theta/k)^k - 1} \cdot \mu \le \sqrt{\theta/(1-\theta)} \cdot \mu$.

Since the estimators E_j are independent and each one falls within two standard deviations of $\mathrm{E}[H]$ with probability at least $3/4$, by a large deviation bound the probability that more than half of them fall outside this range is less than $2^{-t/8} \le \eta$.

2.2 Approximately list-decoding product-sampleable functions In this section we describe and analyze the list-decoder assuming the correlation function $H = G \cdot F^k$ is "product-sampleable", meaning that most restrictions to the last coordinate yield a regular function. The argument follows Levin's proof of the XOR lemma, except that we apply the product sampler from Section 2.1 in lieu of Levin's random sampler.

Definition 2.2. *A function $H \colon \Sigma^k \to \{-1, 1\}$ is* (μ, ε)-product-sampleable *if for all but an ε-fraction of inputs $a \in \Sigma$, the functions $H_a(x_1, \dots, x_{k-1}) = H(x_1, \dots, x_{k-1}, a)$ are all strongly $(\mu^{(k-1)/k}, \frac{1}{2}\mu^{2/k})$-regular.*

Lemma 2.2. *Assume $k \ge 2$. There is a uniform local $(\mu, \mu^{1/k} - \varepsilon)$ list decoder that succeeds with* *probability at least* $(\varepsilon/80)^{O((k-1)^2/\mu^{2/k}\varepsilon^2)}$ *and runs in time* $k^2\log|\Sigma|\cdot\mathrm{poly}(\mu^{-1/k},\varepsilon^{-1})$ *on input* G*, assuming* $H = G\cdot F^k$ *is* $(\mu, \varepsilon\mu^{(k-1)/k}/80)$*-product-sampleable and* $|\mathrm{E}[H]| \geq \mu$.

Approximate list-decoder $LPS^G(k,\mu,|\Sigma|,\varepsilon)$**:**

```
1  Set $s = \lceil 514(k-1)/\mu^{2/k}\varepsilon^2\rceil$ and
       $t = \lceil 8\log(80/\varepsilon)\rceil$.
2  Choose independent sets $S_{ij} \subseteq \Sigma$,
       $1 \leq i \leq k-1$, $1 \leq j \leq t$ of size $s$ each.
3  Guess the values $F(x)$ at random for all
       $x \in S_{ij}$.
4  For every $a$ in $\Sigma$:
5      Let $G_a\colon \Sigma^{k-1} \to \{-1,1\}$ be the function
           $G(x_1,\dots,x_{k-1},a)F(x_1)\cdots F(x_{k-1})$.
6      Let $\tilde{F}(a)$ be the median value of
           $\mathrm{E}[G_a(x) \mid x_i \in S_{ij}$ for all $i]$.
7  Choose a uniformly random $B$ from the
       range $[-1,1]$ within $\lceil\log(4/\varepsilon)\rceil$ bits
       of precision.
8  Let $\tilde{F}_B(a) = 1$ if $\mu^{-(k-1)/k}\tilde{F}(a) \geq B$ and
       $-1$ if not.
9  Output $\pm\tilde{F}_B$ where the sign is chosen
       at random.
```

Steps 2 to 6 implement the product sampler. The output of this sampler produces real-valued estimates $\mu^{-(k-1)/k}\tilde{F}(a)$ of the message bits $F(a)$. The accuracy of the product sampler guarantees that when G_a is regular, these estimates significantly correlate with F on average. In order to extract a $\{-1,1\}$-valued codeword from $\hat{F}$, steps 7 and 8 in effect round the values with respect to a random threshold B. The rounding preserves the correlation in expectation. The expectation can be turned into a noticeable probability at a small price in accuracy.

In the proof of Lemma 2.2 we use the rounding function $[\![\cdot]\!]\colon \mathbb{R} \to [-1,1]$ given by

$$[\![t]\!] = \begin{cases} 1, & \text{if } t > 1,\\ t, & \text{if } -1 \leq t \leq 1,\\ -1, & \text{if } t < -1.\end{cases}$$

FACT 2.1. $[\![\cdot]\!]$ *is a contraction, i.e.,* $|[\![s]\!]-[\![t]\!]| \leq |s-t|$ *for all* s *and* t.

Proof. [Proof of Lemma 2.2] Let R be the set of all a for which the function H_a is strongly $(\mu^{(k-1)/k}, \frac{1}{2}\mu^{2/k})$-regular. By assumption, $\overline{R}$ has measure at most $\varepsilon\mu^{(k-1)/k}/80$. For every $a \in N$, the function

$$\begin{aligned} G_a(x_1,\dots,x_{k-1}) &= H_a(x_1,\dots,x_{k-1})F(a)\\ &= G(x_1,\dots,x_{k-1},a)F(x_1)\cdots F(x_{k-1}).\end{aligned}$$

is also strongly $(\mu^{(k-1)/k}, \frac{1}{2}\mu^{2/k})$-regular, as G_a and H_a may differ only in sign.

By Claim 2.1 with parameters $\theta = \varepsilon^2/257$ and $\eta = 80/\varepsilon$, for all but at most $\varepsilon/80$ of those a that are in R,

$$\left|\tilde{F}(a) - \mathrm{E}[G_a(x)]\right| \leq \mu^{(k-1)/k}\cdot\frac{\varepsilon}{8} \tag{2.2}$$

with probability at least $1-\varepsilon/80$ over the random choices in step 2. Let $A \subseteq R$ be the set of those a's for which inequality (2.2) holds. Then A has expected measure at least $1-\varepsilon/80-\varepsilon\mu^{(k-1)/k}/80 \geq 1-\varepsilon/40$. By Markov's inequality, A has measure at least $1-\varepsilon/20$ with probability at least $1/2$ (2.3)

If a is in A, then by (2.2),

$$\left|\mu^{-(k-1)/k}\tilde{F}(a) - \mu^{-(k-1)/k}\,\mathrm{E}[G_a(x)]\right| \leq \frac{\varepsilon}{8}.$$

By the strong regularity of G_a, $|\mu^{-(k-1)/k}\,\mathrm{E}[G_a(x)]| \leq 1$. Since $[\![\cdot]\!]$ is a contraction,

$$\left|[\![\mu^{-(k-1)/k}\tilde{F}(a)]\!] - \mu^{-(k-1)/k}\,\mathrm{E}[G_a(x)]\right| \leq \frac{\varepsilon}{8}.$$

If a is in R (but not in A), then strong regularity still holds and

$$\left|[\![\mu^{-(k-1)/k}\tilde{F}(a)]\!] - \mu^{-(k-1)/k}\,\mathrm{E}[G_a(x)]\right| \leq 2,$$

as both terms take values between -1 and 1. Finally, if a is not in R then

$$\begin{aligned} &\left|[\![\mu^{-(k-1)/k}\tilde{F}(a)]\!] - \mu^{-(k-1)/k}\,\mathrm{E}[G_a(x)]\right|\\ &\qquad \leq 1 + \mu^{-(k-1)/k}\\ &\qquad \leq 2\mu^{-(k-1)/k}.\end{aligned}$$

Therefore

$$\begin{aligned} &\left|\mathrm{E}_a\left[F(a)\cdot\left([\![\mu^{-(k-1)/k}\tilde{F}(a)]\!] - \mu^{-(k-1)/k}\,\mathrm{E}[G_a(x)]\right)\right]\right|\\ &\qquad \leq \frac{\varepsilon}{8}\cdot\Pr[a\in A] + 2\cdot\Pr[a \in R\setminus A]\\ &\qquad\qquad + 2\mu^{-(k-1)/k}\cdot\Pr[a\notin R]\\ &\qquad \leq \frac{\varepsilon}{8} + 2\cdot\frac{\varepsilon}{20} + 2\mu^{-(k-1)/k}\cdot\frac{\varepsilon\mu^{(k-1)/k}}{80}\\ &\qquad \leq \frac{\varepsilon}{4}. \qquad (2.4)\end{aligned}$$

By the assumption $|\mathrm{E}[H]| \geq \mu$,

$$\begin{aligned} &\big|\mathrm{E}_a\big[F(a)\cdot\mu^{-(k-1)/k}\,\mathrm{E}[G_a(x)]\big]\big| \\ &= \mu^{-(k-1)/k}\big|\mathrm{E}_{x,a}\big[G(x,a)\cdot F(x_1)\cdots F(x_{k-1})F(a)\big]\big| \\ &\geq \mu^{1/k}. \end{aligned} \tag{2.5}$$

From (2.4), (2.5) and the triangle inequality it follows that

$$\big|\mathrm{E}_a\big[F(a)\cdot[\![\mu^{-(k-1)/k}\tilde{F}(a)]\!]\big]\big| \geq \mu^{1/k}-\frac{\varepsilon}{4}. \tag{2.6}$$

If B was a uniform $[-1,1]$ random variable, $\mathrm{E}_B[\tilde{F}_B(a)]$ would equal $[\![\mu^{-(k-1)/k}\tilde{F}(a)]\!]$. As B is precision-bounded, we have the weaker guarantee

$$\big|\mathrm{E}_B[\tilde{F}_B(a)]-[\![\mu^{-(k-1)/k}\tilde{F}(a)]\!]\big| \leq \frac{\varepsilon}{4} \tag{2.7}$$

for every $a\in\Sigma$. From (2.6) and (2.7) it follows that

$$\big|\mathrm{E}_{B,a}[F(a)\cdot\tilde{F}_B(a)]\big| \geq \mu^{1/k}-\frac{\varepsilon}{2}.$$

By Markov's inequality, the inequality

$$\big|\mathrm{E}_a[F(a)\cdot\tilde{F}_B(a)]\big| \geq \mu^{1/k}-\varepsilon.$$

must hold for at least a $\varepsilon/2$ (2.8) fraction of B's. If such a B is chosen, the correlation between the output and F is at least $\mu^{1/k}-\varepsilon$ with probability $1/2$ (2.9) over the choice of sign in step 9.

To summarize, conditioned on events (2.3), (2.8), and (2.9) occurring and the guesses in step 3 of the algorithm being correct, the output of the algorithm has the desired correlation with F. As step 3 involves guessing at most $(k-1)st$ boolean values, the algorithm succeeds with probability at least

$$\frac{\varepsilon}{8}\cdot 2^{-(k-1)st} \geq \left(\frac{\varepsilon}{80}\right)^{O((k-1)^2/\mu^{2/k}\varepsilon^2)}$$

by our choice of parameters.

2.3 Proof of Theorem 2.1

The approximate list-decoder L guesses whether the function $H=G\cdot F^k$ is product-sampleable. If its guess is positive it runs the list-decoding algorithm LPS for product-sampleable functions from Section 2.2. Definitions 2.2 and 2.1 ensure that if H is not product-sampleable then a noticeable fraction of its restrictions have large bias. In this case, L guesses the suitable restriction and runs recursively on it.

The following specification is obtained by unwinding the recursion with uniform guessing probabilities. This choice turns out to be convenient for the analysis. We use the notation a_I to describe a partial assignment restricted to the subset of indices $I\subseteq[k]$, and a_i as a shorthand for $a_{\{i\}}$.

Algorithm $L^G(k,\mu,|\Sigma|,\varepsilon)$**:**

```
1   Choose a random subset $R\subseteq[k]$ and a random
        partial assignment $a_R\sim\Sigma^R$.
2   Let $G'\colon\Sigma^{\overline{R}}\to\{-1,1\}$ to be the function $G$
        restricted to $x_R=a_R$.
3   If $|\overline{R}|=0$, fail.
4   If $|\overline{R}|=1$, output $G'$ or $-G'$ with equal
        probability.
5   Otherwise, output $LPS^{G'}(|\overline{R}|,\mu^{|\overline{R}|/k},|\Sigma|,\varepsilon)$.
```

To prove Theorem 2.1, we will show by strong induction on k that L is a $(\mu,\mu^{1/k}-\varepsilon)$-approximate list decoder with success probability at least

$$p(k,\mu)=2^{-k-1}\cdot(\varepsilon/80)^{C(k-1)^2/\mu^{2/k}\varepsilon^2}$$

where C is a sufficiently large constant. Assume that $|\mathrm{E}[H]|\geq\mu$ for $H=G\cdot F^k$.

Base case $k=1$: R is non-empty with probability $1/2$. In this case the larger one of $\mathrm{E}[F\cdot G]$ and $\mathrm{E}[F\cdot(-G)]$ is at least $\mu\geq\mu-\varepsilon$, so the output of L has correlation at least $\mu-\varepsilon$ with F with probability $1/4$, which is larger than $p(1,\mu)$.

Inductive step: Assume $k\geq 2$ and the claim holds up to $k-1$. We consider two cases.

If H is $(\mu,\varepsilon\mu^{(k-1)/k}/80)$-product-sampleable, then in step 1 the empty set is chosen with probability 2^{-k}, in which case step 5 is triggered with $G'=G$. By Lemma 2.2, the output of LPS^G $(\mu^{1/k}-\varepsilon)$-correlates with F with probability at least $(\varepsilon/80)^{C(k-1)^2/\mu^{2/k}\varepsilon^2}$, so the overall success probability exceeds $p(k,\mu)$ as desired.

The following claim summarizes the irregularity of functions that are not product-sampleable.

CLAIM 2.2. *If H is not $(\mu,\varepsilon\mu^{(k-1)/k}/80)$-product-sampleable then with probability at least $\varepsilon\mu^{(k-1)/k}/80$ over the choice of a_k there exists a proper subset $I\subset[k]$ with $k\in I$ for which with probability at least $\mu^{2|\overline{I}|/k}$ over the choice of $a_{I\setminus\{k\}}$, $|\mathrm{E}[H(x)\mid x_I=a_I]|\geq\mu^{|\overline{I}|/k}$.*

Proof. By Definition 2.2, for more than a $\varepsilon\mu^{(k-1)/k}/80$ fraction of a_k there exists a subset $I \subseteq [k]$, $k \in I$ that violates strong regularity in the following sense:

$$(2.10)\quad \begin{aligned} \mathrm{E}[H(x)H(x') \mid x_I = x'_I, x_k = x'_k = a_k] \\ > (\mu^{(k-1)/k})^2 \cdot (\tfrac{1}{2}\mu^{2/k})^{-|I\setminus\{k\}|} \\ = 2^{|I|-1} \cdot \mu^{2|\overline{I}|/k}. \end{aligned}$$

The subset I must be proper because when $|I| = k$, the right-hand side exceeds one. Fix such an a_k and let a_I be a random extension of a_k. By the independence of $x_{\overline{I}}$ and $x'_{\overline{I}}$,

$$(2.11)\quad \begin{aligned} \mathrm{E}_{a_I}\left[\mathrm{E}[H(x) \mid x_I = a_I]^2\right] \\ = \mathrm{E}[H(x)H(x') \mid x_I = x'_I, x_k = x'_k = a_k]. \end{aligned}$$

When $I = \{k\}$, the claim follows from (2.10) and (2.11) after taking square roots. Otherwise, by Markov's inequality $\mathrm{E}[H(x) \mid x_I = a_I]^2 \geq \mu^{2|\overline{I}|/k}$ with probability at least $(2^{|I|-1} - 1)\mu^{2|\overline{I}|/k}$ over the choice of a_I. Because $2^{|I|-1} - 1$ is at least 1, the claim follows again after taking square roots.

Let U be the event "$|\mathrm{E}[H(x) \mid x_I = a_I]| \geq \mu^{|\overline{I}|/k}$ and $I \subseteq R$." By Claim 2.2 and the uniform choice of R, U has probability at least

$$(2.12)\quad \Pr[U] \geq 2^{-|I|} \cdot \frac{\varepsilon\mu^{(k-1)/k}}{80} \cdot \mu^{2|\overline{I}|/k} \geq \frac{(\varepsilon/80)\mu^{3(k-1)/k}}{2^{|I|}}.$$

Claim 2.3. *Conditioned on U, the output $L^G(k,\mu)$ has correlation at least $\mu^{1/k} - \varepsilon$ with F with probability at least $p(|\overline{I}|, \mu^{|\overline{I}|/k})$.*

Proof. Conditioned on U, the view of $L(k,\mu)$ when querying the oracle G is identical to the view of $L(|\overline{I}|, \mu^{|\overline{I}|/k})$ when querying the oracle $\hat{G}(x_{\overline{I}}) = G(x_{\overline{I}}, a_I)$. Also conditioned on U, the function $\hat{H}(x_{\overline{I}}) = H(x_{\overline{I}}, a_I)$ is at least $\mu^{|\overline{I}|/k}$-biased. The function $\hat{H}$ equals

$$\hat{H}(x_{\overline{I}}) = \sigma \cdot \hat{G}(x_{\overline{I}}) \cdot \prod_{i \in \overline{I}} F(x_i)$$

where $\sigma = \prod_{i \in I} F(a_i)$ is a possible change of sign. By inductive assumption, $L^{\hat{G}}(|\overline{I}|, \mu^{|\overline{I}|/k})$ then has correlation at least $(\mu^{|\overline{I}|/k})^{1/|\overline{I}|} - \varepsilon = \mu^{1/k} - \varepsilon$ with F with the desired probability.

From (2.12) and Claim 2.3 it follows that $L^G(k,\mu)$ succeeds with probability at least

$$\begin{aligned} &\Pr[U] \cdot p(|\overline{I}|, \mu^{|\overline{I}|/k}) \\ &\geq \frac{(\varepsilon/80)\mu^{3(k-1)/k}}{2^{|I|}} \cdot 2^{-|\overline{I}|-1} \cdot (\varepsilon/80)^{C(|\overline{I}|-1)^2/\mu^{2/|\overline{I}|}\varepsilon^2} \\ &\geq 2^{-k-1} \cdot (\varepsilon/80)^{3k-2} \cdot (\varepsilon/80)^{C(k-2)^2/\mu^{2/k}\varepsilon^2} \\ &\geq 2^{-k-1} \cdot (\varepsilon/80)^{C(k-1)^2/\mu^{2/k}\varepsilon^2} \\ &= p(k,\mu) \end{aligned}$$

assuming $\varepsilon \leq \mu^{1/k}$ in the second inequality and $C \geq 4$ in the third one. This completes the inductive step and the proof of Theorem 2.1.

2.4 Proof of Corollary 1.2 Let $\Sigma = \{0,1\}^n$. The proof of Theorem 2.1 shows that the circuit L^G computes a function that predicts F with advantage $\mu^{1/k} - \varepsilon$ with positive probability. In particular, the prediction succeeds for some fixed choice of the randomness. The amount of advice is therefore upper bounded by the randomness complexity of L^G.

The randomness complexity of the list-decoder LPS is governed by the $(k-1)t$ choices of the sets S_{ij} chosen in step 2 and the $(k-1)st$ values of F guessed in step 3. If the elements of each set S_{ij} are chosen in a pairwise independent manner, step 2 can be performed using at most $2ktn$ bits of randomness. Plugging in the parameters for s and t we conclude that LPS has randomness complexity $O(k\log(1/\varepsilon)(n + \mu^{-2/k}\varepsilon^{-2}))$.

In step 5 of the list-decoder L the randomness complexity of the call to LPS is maximized when R is the empty set. In step 1, L requires $k(n+1)$ additional bits of randomness, so the randomness complexity of L is also $O(k\log(1/\varepsilon)(n + \mu^{-2/k}\varepsilon^{-2}))$ as desired.

3 From Distinguishing to Solving Random Noisy Linear Equations

In this section, we show how to use an approximate list-decoder of the k-XOR code to reduce solving random planted k-LIN instances to distinguishing them from completely random instances. Namely, we will prove the following more thoroughly quantified version of Informal Theorem 1.2.

Theorem 3.1. *Suppose that m-equation, n-variable planted η-noisy kLIN instances are distinguishable from random ones in time t with advantage ε, where $\eta < 1/2$. Then, planted instances with $O(m \cdot$*

$(m/\varepsilon)^{2/k} + 2^{2k} n \log n/k)$ equations and n variables can be solved in time polynomial in t, m, n, and $1/\varepsilon$, with probability at least $(\varepsilon/m)^{O(k(m/\varepsilon)^{6/k})}$ over the choice of the instance and the randomness of the solver.

Definition 3.1. (k-LIN) *Let $k, m, n \in \mathbb{N}$, and $\eta \in [0, 1/2)$. An m-equation n-variable k-LIN* instance *is a pair (A, b), where $A \in \{0,1\}^{m\times n}$ is such that each row a_i of A has at most k non-zero entries, and $b \in \{0,1\}^m$. A* random planted η-noisy *k-LIN instance is such a pair $(A, As + e)$ where:*

- *Each row of $A \in \{0,1\}^{m\times n}$ is sampled independently from the* row distribution $\mathcal{R}_{n,k}$, *which is the modulo 2 sum of k independent random indicator vectors in $\{0,1\}^n$ (i.e. vectors of the form $(0, \dots, 0, 1, 0, \dots, 0)$).*
- *$s \in \{0,1\}^n$, called the* planted solution, *is chosen uniformly at random.*
- *$e \in \{0,1\}^m$, called the* noise vector, *is chosen such that each bit in it is 1 independently with probability η.*

In the rest of this section, unless specified otherwise, the number of equations in a k-LIN instance is to be taken to be m, the number of variables to be n, and the noise to be η.

Remark 3.1. *Another natural distribution on the rows of A is the uniform distribution on strings on Hamming weight k. Our results and analysis can be modified to apply to this distribution as well.*

Definition 3.2. (Solving/Distinguishing k-LIN) *We define the following two operations for random k-LIN instances:*

- *An algorithm S is said to* solve *planted η-noisy k-LIN instances with* success probability *p if, given a random planted η-noisy k-LIN instance $(A, As + e)$, it outputs s with probability p (over the randomness of A, e and S itself).*
- *An algorithm D is said to* distinguish *planted η-noisy k-LIN instances from random with* advantage *ε if it distinguishes, with advantage ε, between a random planted η-noisy k-LIN instance $(A, As + e)$ and (A, r), where A is chosen as in k-LIN, but $r \sim \{0,1\}^m$ is chosen at uniform independently of A. That is,*

$$\mathrm{E}_{A,s,e}[\mathsf{D}(A, As + e)] - \mathrm{E}_{A,r}[\mathsf{D}(A, r)] \geq \varepsilon$$

We will be reducing the task of solving a k-LIN instance with m'-equations to that of distinguishing instances with m equations from random (with advantage ε) for some $m' > m$. Our objective here is to keep m' as small as possible in relation to m. It is already known (from [App13]) how to perform such a reduction with $m' = \tilde{\Theta}(m^3/\varepsilon^2)$ that results in constant success probability for the solver. Using the approximate list-decoder constructed in Section 2, we are able to bring m' down significantly at the cost of lower success probability.

Theorem 3.2. (Refined Theorem 3.1) *Suppose that m-equation n-variable planted η-noisy kLIN instances are distinguishable from random in time t with advantage ε. Then planted η-noisy kLIN instances with m' equations and n variables can be solved in time polynomial in t, m, n, $1/\varepsilon$, and $1/(1-2\eta)$ with probability at least p where:*

$$m' = O\big((1-2\eta)^{2/k} \cdot m \cdot (m/\varepsilon)^{2/k} \\ + 2^{2k}\ n \log n/k(1-2\eta)^2\big)$$

and

$$p = (\varepsilon/(1-2\eta)m)^{O(k((1-2\eta)m/\varepsilon)^{6/k})}.$$

The rest of this section is dedicated to the proof of this theorem. Our approach for getting a solver for k-LIN from a distinguisher is broadly divided into the following three parts, each of which we describe briefly below:

1. Using the distinguisher to get a predictor.
2. Using the predictor to get an approximate-solver.
3. Using the approximate-solver to get an actual solver.

The first step is to show (as stated in Lemma 3.1) that a distinguisher for k-LIN can be used to construct a predictor that, given a noisy k-LIN instance $(A, As + e)$, has a small advantage in predicting the answers to random equations – that is, the value of $\langle a, s\rangle$ for random a from the row distribution $\mathcal{R}_{n,k}$. This operation is defined as below.

Definition 3.3. *An algorithm P is called a* predictor *for η-noisy k-LIN with* advantage *δ if, when given a random planted η-noisy k-LIN instance $(A, As+e)$ and a random "row" a from the row distribution $\mathcal{R}_{n,k}$, predicts $\langle a, s\rangle$ with advantage δ. That is,*

$$\mathrm{E}_{A,s,e,a}[\mathsf{P}(a; A, As + e) \cdot (-1)^{\langle a,s\rangle}] \geq \delta$$

We say that such a predictor P predicts s *with advantage* δ *from the* training data $(A, As+e)$.

The predictor is constructed from the distinguisher using standard hybrid arguments. The following lemma is proven in the full version of the paper.

Lemma 3.1. *Suppose there is an algorithm that distinguishes m-equation n-variable planted η-noisy k-LIN instances from random with advantage ε and runs in time t. Then, there is a predictor for m-equation n-variable η-noisy k-LIN that also runs in time t and has advantage $\varepsilon/(1-2\eta)m$.*

Once we have such a predictor, we then use it to solve k-LIN "approximately". That is, given a planted instance $(A, As+e)$, we recover an $\tilde{s}$ that correlates well with s. We use the following shorthand for the measure of correlation between two binary strings. Given $s, \tilde{s} \in \{0,1\}^n$,

$$s \cdot \tilde{s} = \mathrm{E}_i[(-1)^{s[i]}(-1)^{\tilde{s}[i]}]$$

where the expectation is over i drawn at random from $[n]$. Note that this quantity is contained in $[-1,1]$, and $s \cdot \tilde{s} = \gamma$ is the same as saying that s and $\tilde{s}$ agree on a $(1+\gamma)/2$ fraction of coordinates. We also overload this notation to handle the case where s (or $\tilde{s}$) is a $\{-1,1\}$-string, in which case $(-1)^{s[i]}$ in the expression above is to be replaced with $s[i]$.

The operation of approximately solving k-LIN instances is now defined as below.

Definition 3.4. (Approximately Solving k-LIN) *An algorithm* $\tilde{\mathsf{S}}$ *is said to* γ-approximately solve *planted η-noisy k-LIN instances with* success probability p *if, given a random planted η-noisy k-LIN instance $(A, As+e)$, with probability p it outputs some $\tilde{s}$ such that $s \cdot \tilde{s} \geq \gamma$.*

To construct an approximate solver for k-LIN from a predictor, we use the approximate list-decoder for the k-XOR code. Given a k-LIN instance $(A, As+e)$ as training data, we view the "truth-table" of the predictor $\mathsf{P}(\cdot; A, As+e)$ as a corrupt codeword of the k-XOR code. Intuitively, the correctness of the predictor should say that this codeword is not too far from the k-XOR encoding of s. We are unable to decode or list-decode this codeword, however, as the noise in the codeword is too high. Instead, we use the approximate list-decoder and obtain, as one of the elements in the list, an $\tilde{s}$ that is noticeably correlated with s. We then amplify this correlation by exploiting certain symmetries of k-LIN. See the full version of the paper for a more thorough exposition of the intuition behind this approach and the proof of the following lemma that states our results in this respect.

Lemma 3.2. *Let $\mu, \alpha, \gamma \in [0,1]$, and $k \in \mathbb{N}$ be a constant. For some $m, n \in \mathbb{N}$, suppose:*

1. *There is a predictor for m-equation n-variable η-noisy k-LIN that runs in time t_1 and has advantage δ.*
2. *There is a $(\mu, \alpha^{1/k})$ approximate list-decoder for the k-XOR code with messages of length n that runs in time t_2 and has success probability p.*

Let $r = 8\log(8/(1-\gamma))/\alpha^{2/k}$. Then, there is an algorithm that γ-approximately solves (mr)-equation n-variable planted η-noisy k-LIN instances that runs in time $\widetilde{O}(r(t_1+t_2+mn))$, and has success probability $\frac{3}{4}\left[p\left(\delta-\mu\right)\right]^r$.

The final step in our reduction is to convert the approximate solution produced by the approximate solver above into an actual solution. To do this, we employ a technique of Bogdanov and Qiao [BQ12]. In brief, given an approximate solution $\tilde{s}$, to recover the first bit $s[1]$ of the actual solution, we first find a number of equations where the first bit is involved. In each of these equations, we pretend that $\tilde{s}$ is correct about the values of the remaining bits and solve for $s[1]$, and finally set $s[1]$ to be the majority answer. This is repeated for each bit of s and, if enough equations are used, all the bits are recovered correctly. The end result in our case is stated in the following lemma.

Lemma 3.3. *Assuming $m \geq 40\ n\log n/k(1-2\eta)^2\gamma^{2(k-1)}$, there is a $O(mn^2)$-time algorithm that, given a m-equation, n-variable planted noisy k-LIN instance $(A, As+e)$ and a γ-approximate solution $\hat{s}$ that is independent of A and e, outputs s with probability $1-o(1)$.*

We finish our proof by putting the above lemmas together with the approximate list-decoder for the k-XOR code from Section 2.

Proof. [Proof of Theorem 3.2] The hypothesis of the theorem promises a k-LIN distinguisher that runs in time t and has advantage ε. Lemma 3.1 now immediately implies the existence of a k-LIN predictor P that runs in time t and has advantage $\delta = (\varepsilon/(1-2\eta)m)$.

Set $\mu = \delta/2$ and α to be such that $\alpha^{1/k} = \mu^{1/k}/2$ $(= \mu^{1/k} - \mu^{1/k}/2)$. Theorem 2.1 implies a $(\mu, \alpha^{1/k})$ approximate list-decoder for the k-XOR code that runs in time $\widetilde{O}(\mu^{-4}\log n)$, and has success probability $\Omega(\mu)^{O(k/\mu^{4/k})}$.

Set $\gamma = 1/2$. Along with the above predictor and list-decoder, Lemma 3.2 now implies an algorithm that γ-approximately solves m'-equation n-variable planted η-noisy k-LIN instances, where m' is equal to:

$$\begin{aligned} 10 \cdot \frac{32\ m}{\alpha^{2/k}} &= 1280 \cdot \frac{m}{\mu^{2/k}} \\ &\leq 2560 \cdot \frac{m}{\delta^{2/k}} \\ &= 2560 \cdot (1-2\eta)^{2/k} \cdot m \cdot \left(\frac{m}{\varepsilon}\right)^{2/k} \end{aligned}$$

This approximate solver runs in time on the order of:

$$\begin{aligned} &\frac{1}{\alpha^{2/k}} \cdot (t + \mu^{-4}\log n + mn)) \\ &\quad = \mathrm{poly}(t, (1-2\eta), m, n, 1/\varepsilon) \end{aligned}$$

It has success probability at least:

$$\begin{aligned} &\frac{3}{4}\left[(\mu)^{O(k/\mu^{4/k})} \cdot (\delta - \mu)\right]^{32/\alpha^{2/k}} \\ &\qquad \geq \mu^{O(k/\mu^{4/k}\alpha^{2/k})} \\ &\qquad \geq \left(\frac{\varepsilon}{(1-2\eta)m}\right)^{O\left(k((1-2\eta)m/\varepsilon)^{6/k}\right)} \end{aligned}$$

With the above probability, we have a γ-approximate solution. In order to recover the actual solution, we apply Lemma 3.3 with this approximate solution and a fresh set of m'' equations with the same planted solution, where $m'' = 40 \cdot 2^{2(k-1)}\ n\log n/k(1-2\eta)^2$. This gives us the actual solution, incurs an additional running time of $O(mn^2)$ and the final success probability becomes the above multiplied by $(1-o(1))$. This completes the proof of the theorem.

See the full version for the proof of Lemmas 3.1 to 3.3.

3.1 Generalizing to Other Predicates Next, we sketch how to extend our results to apply to predicates other than k-XOR. Let ϕ be any k-ary predicate. The problems we will be talking about are defined by repeated applications of ϕ on ordered subsets of the bits of a string $s \in \{0,1\}^n$. Correspondingly, we use the following notation in this subsection. We will denote by a an ordered tuple in $[n]^k$, and by s_a the string $(s[a[1]], \ldots, s[a[k]]) \in \{0,1\}^k$; $\phi(s_a)$ then denotes the application of ϕ to the bits of s pointed to by a. We denote by $A = \{a_i\}_{i\in[m]}$ an ordered collection of m such tuples, and call each a_i a "row" of A (in analogy to the matrix A from the case of k-XOR). Finally, $\phi(s_A)$ denotes the vector $(\phi(s_{a_1}), \ldots, \phi(s_{a_m}))$.

We call such a pair $(A, \phi(s_A))$ an m-equation n-variable ϕ-CSP instance. The problems we are interested in are the following, where each row a_i of A is sampled uniformly at random from $[n]^k$, and s is sampled uniformly from $\{0,1\}^n$.

Solving a planted instance: Given A and $\phi(s_A)$, find s.

Distinguishing planted from random instances: Distinguish the distribution $(A, \phi(s_A))$ from (A, r), where $r \sim \{0,1\}^m$ is independent of A.

The formal definitions of the above follow as the natural generalization of Definition 3.2. Let k_ϕ be the size of the smallest parity that has non-zero correlation with ϕ. We show the following theorem.

THEOREM 3.3. *Let ϕ be any k-ary predicate, and $m < o(n^{k/2})$. Suppose that m-equation, n-variable planted ϕ-CSP instances are distinguishable from random ones in time t with advantage ε. Then, planted instances with $O(m\cdot(m/\varepsilon)^{2/k_\phi} + 2^{2k_\phi} n\log n/k_\phi)$ equations and n variables can be solved in time polynomial in t, m, n^k, and $1/\varepsilon$, with probability at least $(\varepsilon/m)^{O(k(m/\varepsilon)^{6/k_\phi})}$ over the choice of the instance and the randomness of the solver.*

This theorem is proved following the same outline as the proof of Theorem 3.2. The ϕ-CSP analogue of Lemma 3.1 (which would obtain a predictor for $\phi(s_A)$ from a distinguisher) follows from the same hybrid argument used there, and that of Lemma 3.3 (which transforms an approximate solver to an actual solver) follows from the work of Bogdanov and Qiao [BQ12]. All that remains is to prove the following analogue of Lemma 3.2.

LEMMA 3.4. *Let $\mu, \alpha, \gamma \in [0,1]$, and ϕ be a k-ary predicate for some constant k. For some $m, n \in \mathbb{N}$, suppose:*

1. *There is a predictor for m-equation n-variable ϕ-CSP that runs in time t_1 and has advantage δ.*

2. *There is a $(\mu, \alpha^{1/k_\phi})$ approximate list-decoder for the k_ϕ-XOR code with messages of length n that runs in time t_2 and has success probability p.*

Let $r = 32\log(8/(1-2\gamma))/\alpha^{2/k_\phi}$, and suppose $\gamma > 10\sqrt{\log(1/p)/n}$. Then, there is an algorithm that γ-approximately solves (mr)-equation n-variable planted ϕ-CSP instances that runs in time $\widetilde{O}(r(t_1+t_2+mn))$, and has success probability at least $\frac{1}{4}\left[\frac{p}{4}\left(\delta\cdot 2^{-k}-\mu\right)\right]^{r+1}$.

The proof of this lemma is the only place where the fact that we are dealing with ϕ-CSP rather than k-LIN matters, and we prove it in the full version by showing an alternative reduction that does not make use of symmetries that are specific to k-LIN.

4 Bounds on list size

In this section we state and prove upper and lower bounds on the list size $\ell(\mu,\alpha)$. The upper bound in Corollary 1.1 follows from Theorem 2.1 and a counting argument. Proposition 4.1 gives a substantially tighter non-constructive upper bound in the regime $\alpha < \mu^2$. The lower bound of Proposition 4.2 in the regime $\alpha > \mu$ is proved by a volume argument. The lower bound in Proposition 4.3, which applies to the whole range of parameters, is obtained by analyzing a specific corrupted codeword.

4.1 Proof of Corollary 1.1 We show that the existence of an approximate list-decoder for a code of message length $|\Sigma|$ that succeeds with probability at least p implies $\ell \le \ln 2\cdot|\Sigma|/p$. Plugging in the value of p from Theorem 2.1 then gives Corollary 1.1.

Let *list* be the collection of outputs generated by $\ln 2\cdot|\Sigma|/p$ independent runs of the approximate list-decoder L^G. If a codeword μ-correlates with G, the probability that it doesn't α-correlate with anything in *list* is at most $(1-p)^{\ln 2\cdot|\Sigma|/p} < 2^{-|\Sigma|}$. Since there are at most $2^{|\Sigma|}$ codewords that μ-correlate with G, by a union bound there is a positive probability that *list* covers all of them.

4.2 Non-constructive upper bound in the regime $\alpha < \mu^2$ The following lemma, which is essentially the proof of the Johnson bound, gives a much tighter upper bound on list size than Corollary 1.1 in the regime $\alpha < \mu^2$.

PROPOSITION 4.1. *For every $0 < \alpha < \mu^2$ and every binary code, $\ell(\mu,\alpha) \le (1-\alpha)/(\mu^2-\alpha)$.*

For example, $\ell(\mu,\mu^2/2) \le 4/\mu^2$. In the case of the k-XOR code, Proposition 4.1 shows the existence of a list $F_1,\dots,F_\ell$ of *messages* such that $\mathrm{E}[G\cdot F^k] \ge \mu$ implies $|\mathrm{E}[F\cdot F_i]| \ge \alpha^{1/k}$ for some $i \in [\ell]$ (since $\mathrm{E}[F^k\cdot F_i^k] = \mathrm{E}[F\cdot F_i]^k$).

Proof. Let ℓ be the maximal value for which there exists a list $C_1,\dots,C_\ell$ such that $\mathrm{E}[G\cdot C_i]\ge\mu$ for all $1\le i\le\ell$ and $\mathrm{E}[C_i\cdot C_j]\le\alpha$ for all $i\ne j$. Then for every $t\ge 0$,

$$\begin{aligned}0 &\le \mathrm{E}\big[(C_1+\cdots+C_\ell-tG)^2\big]\\ &\le \sum_{i=1}^{\ell}\mathrm{E}[C_i^2] + \sum_{i\ne j}\mathrm{E}[C_i\cdot C_j]\\ &\qquad - 2t\sum_{i=1}^{\ell}\mathrm{E}[C_i\cdot G] + t^2\,\mathrm{E}[G^2]\\ &\le \ell + \ell(\ell-1)\delta - 2\ell\mu t + t^2\end{aligned}$$

This is only possible if the discriminant $4\ell^2\mu^2 - 4(\ell + (\ell^2-\ell)\alpha)$ (of the quadratic in t) is nonnegative, implying that $\delta \ge \mu^2$ or $\ell \le (1-\alpha)/(\mu^2-\alpha)$.

If $\mathrm{E}[G\cdot C] \ge \mu$ then $\mathrm{E}[C\cdot C_i^k]$ must be greater than α for some i, for otherwise the maximality of ℓ would be contradicted.

4.3 Lower bound in the regime $\alpha > \mu$

PROPOSITION 4.2. *For the k-XOR code, when $\mu < \alpha < 1$,*

$$\ell(\mu,\alpha) \ge \frac{4}{e\sqrt{1-\mu^{2/k}\cdot|\Sigma|}}\cdot\exp\big(\tfrac{1}{2}(\alpha^{2/k}-\mu^{2/k})|\Sigma|\big).$$

Let h denote the binary entropy function and $0\le\delta\le 1$. We will use the following bounds on the volume of Hamming balls:

$$\binom{N}{\le(1-\delta)N/2} \le 2^{Nh\left(\frac{1-\delta}{2}\right)} \tag{4.13}$$

$$\binom{N}{(1-\delta)N/2} \ge \frac{4}{e\sqrt{1-\delta^2 N}}2^{Nh\left(\frac{1-\delta}{2}\right)} \tag{4.14}$$

The following Taylor expansion is valid for all $-1\le\delta\le 1$:

$$h\left(\frac{1-\delta}{2}\right) = 1 - \frac{1}{2\ln 2}\cdot\sum_{i=1}^{\infty}\frac{\delta^{2i}}{i(2i-1)}. \tag{4.15}$$

Proof. Let $N = |\Sigma|$ and G be the constant function 1. The codewords $\mathcal{C}$ that agree on a $(1-\mu)/2$-fraction to

G are exactly those that encode messages of relative Hamming weight at most $(1-\mu^{1/k})/2$, so the number of such codewords is at least

$$|\mathcal{C}| \geq \binom{N}{(1-\mu^{1/k})N/2} \geq \frac{4}{e\sqrt{1-\mu^{2/k}\cdot N}} \cdot 2^{h\left(\frac{1-\mu^{1/k}}{2}\right)\cdot N},$$

by (4.14). On the other hand, the codewords that agree on a $(1-\alpha)/2$-fraction to any given codeword $C_i = F_i^k$ are those that encode messages within Hamming distance at most $(1-\alpha^{1/k})/2$ from F_i (as noted in Remark 1.1), so there are at most $2^{h((1-\alpha^{1/k})/2)\cdot N}$ of them by (4.13). Therefore covering $\mathcal{C}$ requires a list of size

$$\frac{4}{e\sqrt{1-\mu^{2/k}\cdot N}} \cdot 2^{h\left(\frac{1-\mu^{1/k}}{2}\right)\cdot N - h\left(\frac{1-\alpha^{1/k}}{2}\right)\cdot N}.$$

Using the Taylor expansion (4.15), we can lower bound $h((1-\mu^{1/k})/2) - h((1-\alpha^{1/k})/2)$ by the difference of the leading terms in the summation, which equals $(\alpha^{2/k} - \mu^{2/k})/2\ln 2$, completing the proof.

4.4 A general lower bound

Proposition 4.3. *For the k-XOR code, when* $\mu \geq |\Sigma|^{-1/2}$, $\ell(\mu, \alpha) \geq \Omega(\alpha^{2/k}\mu^{-2})$, *assuming* $|\Sigma|$ *is a power of two.*

Proof. We first assume that μ is equal to $|\Sigma|^{-1/2}$. Let Σ be a $\mathbb{F}_2$-vector space and $\mathcal{H} \subseteq \{-1,1\}^\Sigma$ be the Hadamard code. Its codewords are the functions $H(x) = (-1)^{\langle a,x\rangle}$. The codewords of the k-wise tensor product $\mathcal{H}^k$ of $\mathcal{H}$ are given by $H^k(x_1,\dots,x_k) = H(x_1)\cdots H(x_k) = H(x_1+\dots+x_k)$. Thus the code $\mathcal{H}^k$ is isomorphic to $\mathcal{H}$ as a linear space.

Let G be the corrupted codeword $G(x_1,\dots,x_k) = B(x_1+\dots+x_k)$, where B is the bent function

$$B(z) = (-1)^{z_1z_2+\dots+z_{t-1}z_t}, \qquad t = \log|\Sigma|.$$

For every codeword H of $\mathcal{H}$, $\mathrm{E}[GH^k] = \mathrm{E}[BH]$. The correlation of B with every linear function is identical up to sign, so by Parseval's identity $\mathrm{E}[BH]$ always equals $|\Sigma|^{-1/2}$ or $-|\Sigma|^{-1/2}$. After a possible change of sign in G we may assume that $\mathrm{E}[GH^k] \geq |\Sigma|^{-1/2}$ for at least half the codewords in $\mathcal{H}^k$. Since all these codewords also belong to the k-XOR code, there must exist a list $X_1^k,\dots,X_\ell^k$ of k-XOR codewords such that half the codewords in $\mathcal{H}$ $\alpha^{1/k}$-correlate to some X_i. Viewed as vectors in $\mathbb{R}^\Sigma$, the elements of $\mathcal{H}$ are orthonormal. By Pythagoras' theorem any X_i can $\alpha^{1/k}$-correlate with at most $\alpha^{-2/k}$ of them. It follows that $\ell = \Omega(|\Sigma|\cdot\alpha^{2/k})$.

When $\mu > |\Sigma|^{-1/2}$ we apply the argument to a dimension-$\lceil\log 1/\mu^2\rceil$ quotient of the Hadamard code.

5 Open Questions

The main coding-theoretic question left open is the dependence of the list size on the correlation μ at large distances for the k-XOR code. The upper bound in Corollary 1.1 is exponential in $\mu^{\Theta(1/k)}$, while the lower bound in Proposition 4.3 is proportional to $1/\mu^2$. A tensoring argument shows that $\ell_{2k}(\mu^k, \alpha^k) > \ell_2(\mu, \alpha)$, where ℓ_k is the list size for the k-XOR code. If, say, $\ell_2(\mu, \mu/2)$ were lower bounded by an exponential in $1/\mu$, an exponential lower bound certifying the optimality of Corollary 1.1 would follow. On the other hand, any improvement in the success probability in the decoder (and therefore the list size) would improve the success probability of our k-LIN reduction.

While our reduction is sample-efficient, it still incurs a loss of $O(m^{2/k})$. Is it possible to reduce this loss by reusing training data in different uses of the predictor? One intriguing possibility is suggested by our product sampler for regular functions. When predicting the answer to a fixed equation a, the predictor P takes as input m samples $(a_1,b_1),\dots,(a_m,b_m)$, and the bias of the predictor over all these samples is towards $\langle a,s\rangle$. So amplifying the probability of successful prediction is the same as estimating the bias of P. And if P were a regular function, we would be able to use our product sampler to reuse samples during amplification. While there is no reason to expect an arbitrary predictor to be regular, it might be possible to convert it into a regular one.

Finally, the success probability of the solver produced by our reduction becomes trivial for small values of k (that is, if $k \leq 6$ and $m = \Omega(n)$). Is it possible to perform meaningful solving-to-distinguishing reductions for k-LIN for such small values of k?

Acknowledgments We thank Benny Applebaum, Marco Carmosino, Siyao Guo, Russell Impagliazzo, Sebastian Muller, Luca Trevisan, and Osamu Watanabe for helpful discussions.

Andrej Bogdanov is supported by RGC GRF grant CUHK14238716. Manuel Sabin is supported by the NSF GRFP under Grant #DGE-1106400.

Prashant Nalini Vasudevan was also supported by NSF Grant CNS-1350619, and by DARPA (IBM – DARPA Prime) under Agreement #4915012803.

References

[ABW10] Benny Applebaum, Boaz Barak, and Avi Wigderson. Public-key cryptography from different assumptions. In Leonard J. Schulman, editor, *Proceedings of the 42nd ACM Symposium on Theory of Computing, STOC 2010, Cambridge, Massachusetts, USA, 5-8 June 2010*, pages 171–180. ACM, 2010.

[AGS03] Adi Akavia, Shafi Goldwasser, and Shmuel Safra. Proving hard-core predicates using list decoding. In *44th Symposium on Foundations of Computer Science (FOCS 2003), 11-14 October 2003, Cambridge, MA, USA, Proceedings*, pages 146–157. IEEE Computer Society, 2003.

[Ale11] Michael Alekhnovich. More on average case vs approximation complexity. *Computational Complexity*, 20(4):755–786, 2011.

[AOW15] Sarah R. Allen, Ryan O'Donnell, and David Witmer. How to refute a random CSP. In Venkatesan Guruswami, editor, *IEEE 56th Annual Symposium on Foundations of Computer Science, FOCS 2015, Berkeley, CA, USA, 17-20 October, 2015*, pages 689–708. IEEE Computer Society, 2015.

[App13] Benny Applebaum. Pseudorandom generators with long stretch and low locality from random local one-way functions. *SIAM J. Comput.*, 42(5):2008–2037, 2013.

[App16] Benny Applebaum. Cryptographic hardness of random local functions. *Computational Complexity*, 25(3):667–722, Sep 2016.

[App17] B. Applebaum. Exponentially-hard gap-csp and local prg via local hardcore functions. In *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*, volume 00, pages 836–847, Oct. 2017.

[BGM$^+$16] Andrej Bogdanov, Siyao Guo, Daniel Masny, Silas Richelson, and Alon Rosen. On the hardness of learning with rounding over small modulus. In *Proceedings of the 13th Theory of Cryptography Conference (TCC)*, 2016. To appear.

[BKS14] Boaz Barak, Jonathan A. Kelner, and David Steurer. Rounding sum-of-squares relaxations. In *Proceedings of the Forty-sixth Annual ACM Symposium on Theory of Computing*, STOC '14, pages 31–40, New York, NY, USA, 2014. ACM.

[BLRL$^+$18] Shi Bai, Tancrède Lepoint, Adeline Roux-Langlois, Amin Sakzad, Damien Stehlé, and Ron Steinfeld. Improved security proofs in lattice-based cryptography: Using the rényi divergence rather than the statistical distance. *Journal of Cryptology*, 31(2):610–640, Apr 2018.

[BM16] Boaz Barak and Ankur Moitra. Noisy tensor completion via the sum-of-squares hierarchy. In Vitaly Feldman, Alexander Rakhlin, and Ohad Shamir, editors, *Proceedings of the 29th Conference on Learning Theory, COLT 2016, New York, USA, June 23-26, 2016*, volume 49 of *JMLR Workshop and Conference Proceedings*, pages 417–445. JMLR.org, 2016.

[BPR12] Abhishek Banerjee, Chris Peikert, and Alon Rosen. Pseudorandom functions and lattices. In David Pointcheval and Thomas Johansson, editors, *Advances in Cryptology - EUROCRYPT 2012 - 31st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cambridge, UK, April 15-19, 2012. Proceedings*, volume 7237 of *Lecture Notes in Computer Science*, pages 719–737. Springer, 2012.

[BQ12] Andrej Bogdanov and Youming Qiao. On the security of Goldreich's one-way function. *Computational Complexity*, 21(1):83–127, 2012.

[CW04] Moses Charikar and Anthony Wirth. Maximizing quadratic programs: Extending grothendieck's inequality. In *45th Symposium on Foundations of Computer Science (FOCS 2004), 17-19 October 2004, Rome, Italy, Proceedings*, pages 54–60, 2004.

[Fei02] Uriel Feige. Relations between average case complexity and approximation complexity. In *Proceedings of the Thiry-fourth Annual ACM Symposium on Theory of Computing*, STOC '02, pages 534–543, New York, NY, USA, 2002. ACM.

[FPV15] Vitaly Feldman, Will Perkins, and Santosh Vempala. On the complexity of random satisfiability problems with planted solutions. In Rocco A. Servedio and Ronitt Rubinfeld, editors, *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015, Portland, OR, USA, June 14-17, 2015*, pages 77–86. ACM, 2015.

[GL89] Oded Goldreich and Leonid A. Levin. A hard-core predicate for all one-way functions. In David S. Johnson, editor, *Proceedings of the 21st Annual ACM Symposium on Theory of Computing, May 14-17, 1989, Seattle, Washigton, USA*, pages 25–32. ACM, 1989.

[GNW11] Oded Goldreich, Noam Nisan, and Avi Wigderson. *On Yao's XOR-Lemma*, pages 273–301. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.

[GW95] Michel X. Goemans and David P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *J. ACM*, 42(6):1115–1145, November 1995.

[HILL99] Johan Håstad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. A pseudorandom generator from any one-way function. *SIAM J. Comput.*, 28(4):1364–1396, 1999.

[IJK09] Russell Impagliazzo, Ragesh Jaiswal, and Valentine Kabanets. Approximate list-decoding of direct product codes and uniform hardness amplification. *SIAM J. Comput.*, 39(2):564–605, 2009.

[IJKW10] Russell Impagliazzo, Ragesh Jaiswal, Valentine Kabanets, and Avi Wigderson. Uniform direct product theorems: Simplified, optimized, and derandomized. *SIAM J. Comput.*, 39(4):1637–1665, 2010.

[Imp95a] Russell Impagliazzo. Hard-core distributions for somewhat hard problems. In *In 36th Annual Symposium on Foundations of Computer Science*, pages 538–545. IEEE, 1995.

[Imp95b] Russell Impagliazzo. Hard-core distributions for somewhat hard problems. In *36th Annual Symposium on Foundations of Computer Science, Milwaukee, Wisconsin, 23-25 October 1995*, pages 538–545. IEEE Computer Society, 1995.

[IW97] Russell Impagliazzo and Avi Wigderson. $P = BPP$ if E requires exponential circuits: Derandomizing the XOR lemma. In Frank Thomson Leighton and Peter W. Shor, editors, *Proceedings of the Twenty-Ninth Annual ACM Symposium on the Theory of Computing, El Paso, Texas, USA, May 4-6, 1997*, pages 220–229. ACM, 1997.

[KMOW17] Pravesh K. Kothari, Ryuhei Mori, Ryan O'Donnell, and David Witmer. Sum of squares lower bounds for refuting any CSP. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017*, pages 132–145, 2017.

[Lev87] Leonid A. Levin. One way functions and pseudorandom generators. *Combinatorica*, 7(4):357–363, Dec 1987.

[LPR10] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. In Henri Gilbert, editor, *Advances in Cryptology - EUROCRYPT 2010, 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Monaco / French Riviera, May 30 - June 3, 2010. Proceedings*, volume 6110 of *Lecture Notes in Computer Science*, pages 1–23. Springer, 2010.

[MM11] Daniele Micciancio and Petros Mol. Pseudorandom knapsacks and the sample complexity of LWE search-to-decision reductions. In Phillip Rogaway, editor, *Advances in Cryptology - CRYPTO 2011 - 31st Annual Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2011. Proceedings*, volume 6841 of *Lecture Notes in Computer Science*, pages 465–484. Springer, 2011.

[Pie12] Krzysztof Pietrzak. Cryptography from learning parity with noise. In Mária Bieliková, Gerhard Friedrich, Georg Gottlob, Stefan Katzenbeisser, and György Turán, editors, *SOFSEM 2012: Theory and Practice of Computer Science - 38th Conference on Current Trends in Theory and Practice of Computer Science, Špindlerův Mlýn, Czech Republic, January 21-27, 2012. Proceedings*, volume 7147 of *Lecture Notes in Computer Science*, pages 99–114. Springer, 2012.

[Reg09] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. *J. ACM*, 56(6), 2009.

[RRS17] Prasad Raghavendra, Satish Rao, and Tselil Schramm. Strongly refuting random csps below the spectral threshold. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017*, pages 121–131, 2017.

[STV01] Madhu Sudan, Luca Trevisan, and Salil P. Vadhan. Pseudorandom generators without the XOR lemma. *J. Comput. Syst. Sci.*, 62(2):236–266, 2001.

[Yao82] Andrew Chi-Chih Yao. Theory and applications of trapdoor functions (extended abstract). In *23rd Annual Symposium on Foundations of Computer Science, Chicago, Illinois, USA, 3-5 November 1982*, pages 80–91. IEEE Computer Society, 1982.