Electronic Colloquium on Computational Complexity, Revision 1 of Report No. 139 (2025)

Improved Search-to-Decision Reduction for Random Local Functions

Kel Zin Tan* Prashant Nalini Vasudevan†

February 17, 2026

Abstract

A random local function defined by a d-ary predicate P is one where each output bit is computed by applying P to d randomly chosen bits of its input. These represent natural distributions of instances for constraint satisfaction problems. They were put forward by Goldreich [Gol11] as candidates for low-complexity one-way functions, and have subsequently been widely studied also as potential pseudo-random generators.

We present a new search-to-decision reduction for random local functions defined by any predicate of constant arity. Given any efficient algorithm that can distinguish, with advantage ε, the output of a random local function with m outputs and n inputs from random, our reduction produces an efficient algorithm that can invert such functions with $\tilde{O}(m(n/\varepsilon)^2)$ outputs, succeeding with probability $\Omega(\varepsilon)$. This implies that if a family of local functions is one-way, then a related family with shorter output length is a family of pseudo-random generators.

Prior to our work, all such reductions that were known required the predicate to have additional sensitivity properties, whereas our reduction works for any predicate. Our results also generalise to some super-constant values of the arity d, and to noisy predicates.

* Department of Computer Science, National University of Singapore. Email: kelzin@u.nus.edu
† Department of Computer Science, National University of Singapore. Email: prashvas@nus.edu.sg

ISSN 1433-8092

Contents

1 Introduction

Local cryptographic primitives are those that can be computed in constant parallel time, i.e., implementable by constant-depth circuits (NC^0) [CM01, MST03, AIK06, Gol11]. At its core, local cryptography poses the fundamental question of whether secure cryptographic functions can be designed such that each output bit depends on only a constant number of input bits. Beyond their natural efficiency, concepts from local cryptography have found many other applications, such as in secure computation [ADI+17, BCG+17, BCM23, BCM+24], learning theory [DV21], and indistinguishability obfuscation [LV17, JLS21, JLS22].

Random Local Functions A remarkable natural candidate for local One-Way Functions (OWF) is the family of random local functions, introduced by Goldreich [Gol11]. Fix a d-ary predicate $P : \mathbb{F}_2^d \to \mathbb{F}_2$ for some constant $d \geq 3$. A random local function defined by P is one where each output bit is set to be the result of applying P to d randomly chosen input bits.

To be more precise, this is a function $f_{G,P} : \mathbb{F}_2^n \to \mathbb{F}_2^m$ such that on an input $s \in \mathbb{F}_2^n$, the i-th output bit of the function $f_{G,P}(s)_i$ is $P(s_{j_1}, \ldots, s_{j_d})$, where each $j_k \in [n]$ is a randomly selected index. The collection of indices for each output bit is represented as a tuple (or an ordered hyperedge) $S_i = (j_1, \ldots, j_d) \in [n]^d$. The collection of tuples is described as a hypergraph $G = (S_i)_{i \in [m]}$. Goldreich conjectured that if the predicate P is appropriately chosen and m is not too large or small, then such a function is one-way with high probability.[1]

Crucial to the hardness of inverting these functions is the choice of the predicate P and the relationship between the output and input lengths m and n. For instance, it is known that, regardless of the choice of predicate P, inversion can be performed efficiently for some $m = O(n^{\frac{1}{2}\lfloor 2d/3 \rfloor} \log n)$ [App16, Corollary 3.6]. Over the past few decades, there has been an active line of work studying the complexity of inversion for various predicates and parameter settings [Gol11, AHI05, BQ09, CEMT09, Its10, OW14, AL16, App16, CDM+18, COST19, YGJL22, DMR23]. We discuss this work in more detail in Section 1.3.

Related Problems While conceptually simple, the problem of inverting a random local function is related to several important problems from various areas of Computer Science. Viewing the outputs of the function as constraints that the input needs to satisfy, this is an instance of a random constraint satisfaction problem (CSP) [AIK08, CEMT09, RRS17, GHKM23, DV21].

Furthermore, suppose the predicate P is a noisy XOR function, i.e. P computes the parity of its inputs and flips it with a small probability. This corresponds to the sparse learning parity with noise problem (LPN) [Fei02, Ale03, FGKP06, CSZ24, BBTV24], which has many applications in public key cryptography [ABW10], machine learning [BEG+22], indistinguishability obfuscation [RVV24] and more [CGHKV24, DJ24, ABCM25, BCM+25]. If we then interpret the hypergraph as describing the generator matrix of a linear code, inverting the local function corresponds to decoding its noisy codewords from random noise. Similar to how Linear Regression is usually interpreted, this may also be seen as a learning problem where the task is to learn the hidden linear function $\langle x, \cdot \rangle$ given many noisy evaluations.

[1] Goldreich originally studied a deterministic version of these functions where the hypergraph is fixed to be one with certain expansion properties. Nevertheless, as random graphs are expanding with high probability, his conjectures also imply the one-wayness of the functions we describe here.

Decision Problem Interpreting the problem of inversion as a *search* problem, one can naturally define the corresponding *decision* problem as distinguishing the output of a random local function $(G, f_{G,P}(s))$, on a random input s, from (G, b), where $b \in \mathbb{F}_2^m$ is uniformly random and independent of G. Viewing the input s as the random seed, the hardness of the decision problem would imply that the family of local functions is a family of Pseudo-Random Generators (PRG).

Given their locality, such PRGs could be very efficient to compute, but have a couple of limitations. For example, with probability $1/\mathrm{poly}(n)$, the graph G has a pair of repeated hyperedges, in which case the corresponding output bits of $f_{G,P}$ can be used to distinguish it from random. The best bound on an adversary's distinguishing advantage we can hope for against a random local function is thus some $1/\mathrm{poly}(n)$; we refer to such PRGs as "weak" PRGs. Second, known algorithms place a limit of $\tilde{O}(n^{\lfloor 2d/3 \rfloor/2})$ on the stretch of this PRG if a d-ary predicate is used [App16], and even stronger limits are known for certain classes of predicates [BQ09, AL16].

Regardless, it is still possible to construct a local PRG with negligible distinguishing advantage (a "strong" PRG) from a random local function by processing the output further. Applebaum [App12] showed how to get a strong local PRG with linear stretch by applying a local extractor to the output of an unpredictable random local function. Later, Applebaum and Kachlon [AK19] effectively resolved these issues by showing that any weak local PRG that has non-trivial polynomial stretch can be transformed into a strong local PRG with any desired polynomial stretch, at the cost of some degradation of the locality. The key idea is to apply an additional layer of a local randomness extractor to the output of the weak PRG.

Theorem 1.1 ([AK19, Theorem 2.12] weak-to-strong compiler)**.** *For every constant $d \in N$, $a > 0$ and $c, c' > 1$, there exists a constant d' for which the following holds. Any d-local PRGs that stretch n bits into n^c bits with at most $\varepsilon = 1/n^a$ distinguishing advantage can be converted into d'-local PRGs that stretch n bits to $n^{c'}$ bits with at most a negligible distinguishing advantage.*

Search-to-Decision Reductions Following this discussion, if we can reduce the above search problem to the decision problem, we can construct local PRGs from the one-wayness of random local functions. Further, given the fundamental nature of the family of local functions and the aforementioned connections to CSPs, decoding, etc., such reductions would be of broad interest even outside the context of local cryptography.

Such a reduction was presented by Applebaum [App12] for predicates P that satisfy a sensitivity requirement. We say that a predicate P is sensitive if there exists a variable such that flipping this variable always results in the flipping of the output.

Theorem 1.2 ([App12, Theorem 1.4])**.** *Fix any sensitive d-ary predicate P. Suppose there exists an efficient algorithm with distinguishing advantage ε for the decision problem for random local functions defined by P with m outputs and n inputs. Then there is an efficient algorithm with success probability $\Omega(\varepsilon)$ for the search problem for random local functions defined by P with $O(m^3/\varepsilon^2)$ outputs and n inputs.*

They extend the above result to accommodate noisy predicates P, incurring an additional $\log n$ factor in the output length required by the search algorithm [App12, Theorem 4.5]. The proof uses a distinguisher to derive a "next-bit predictor" that can predict an output bit given all previous output bits. It is then subsequently used to solve the search problem.

They also show that if the next-bit predictor gives a constant advantage, then the function can be inverted with constant probability; this holds without any blowup in output length and without requiring the predicate to be sensitive, but falls short of leading to pseudo-randomness or a reduction to the decision problem.

There have since been a few notable improvements to the above search-to-decision reduction in the special case of P being the noisy XOR predicate. Assuming the existence of a decision algorithm for m outputs and n inputs with ε distinguishing advantage for the noisy d-ary XOR predicate P with noise parameter η,

- [BSV19] showed that there exists an algorithm for the search problem with $O(m(m/\varepsilon)^{2/d})$ outputs, maintaining the same locality constant d, but succeeding only with small probability $\exp(-\tilde{O}(m/\varepsilon)^{6/d})$. Their approach relies on a new approximate local list-decoding algorithm for the d-XOR code at large distances.
- [BRT25] showed that there exists an algorithm for the search problem with $O(nm+n/\eta^2\varepsilon^{2(d-2)})$ outputs and n inputs for the noisy $(d+1)$-ary XOR predicate. They further generalise their result to any sensitive predicate, though again the predicates in the decision and search problems are slightly different. The main approach is a transformation that removes a sensitive variable from the predicate, creating a distributional shift. This shift then allows the construction of a predictor that solves the search problem.

1.1 Our Contribution

In this work, we present a new search-to-decision reduction for random local functions, captured by the following theorem.

Theorem 1.3 (Informal, see Theorem 3.1)**.** *Fix any d-ary predicate P. Suppose there exists an efficient algorithm with distinguishing advantage ε for the decision problem for random local functions defined by P with m outputs and n inputs. Then there is an efficient algorithm with success probability $\Omega(\varepsilon)$ for the search problem for random local functions defined by P with $O(m(n/\varepsilon)^2\log^3(n/\varepsilon))$ outputs and n inputs.*

Remark 1.4. If the predicate P is biased, the distinguishing task with a random binary string is trivial. In this case, we define the decision problem to instead distinguish from a random binary string with the same bias as P. That is, if $\Pr[P(x)=1]=\eta$, then the distinguishing task is with the distribution (G,b) where b is sampled from $Bern(\eta)^m$.

The main improvement of our work over all existing search-to-decision reductions is the removal of the requirement of the predicate being sensitive. This generalisation opens up the possibility of constructing local PRGs (including strong ones, following Theorem 1.1) using the hardness of inverting a wider range of local functions. Lack of sensitivity is potentially beneficial from a cryptographic perspective, as it means one less form of structure, which may make certain classes of attacks harder to mount.

While some of the steps in our reduction are similar to those in prior work, the core of our approach is substantially different. For instance, in contrast to [App12, BSV19], we directly obtain our search algorithm from the decision algorithm, without going through an intermediate predictor for output bits. And in contrast to [BRT25], the predicate is preserved in the course of our reduction.

Our analysis depends only minimally on the predicate itself, relying instead on the mixing properties of certain transformations we perform on the hypergraph. We hope that these techniques will be useful in similar reductions beyond the context of local functions as well. We note that our reduction does not achieve the same level of sample complexity as those for sensitive predicates (approximately losing a factor of n), but we introduce techniques that we hope will allow future work to close the remaining gap between our loss factor and the tighter bounds known for sensitive predicates.

Optimal Reduction For sensitive predicates, there is currently no proven generic sample complexity gap between the search problem and the decision problem. For a general predicate, however, there is a trivial gap of $\sqrt{n}$. Consider a random local function with m outputs and n inputs, with the predicate P being the majority of 3 variables. One can solve the distinguishing task by finding a pair of outputs that depend on the same input. This is because for a majority predicate, sharing an input implies correlated output. With around $m = \sqrt{n}$ samples, such a pair appears with constant probability. However, it is information theoretically impossible to solve the search with just $\sqrt{n}$ outputs. Therefore, the best possible search to decision reduction reduces search with $\tilde{O}_\varepsilon(m\sqrt{n})$ samples to decision with m samples. This shows that our reduction is $\tilde{O}_\varepsilon(n^{1.5})$ factor far from optimal.

Generalisation In our hypergraph modelling of local functions, we allow a hyperedge to contain repeated vertices. The alternative model that requires all vertices in a hyperedge to be distinct is also commonly used. We extend our reduction to work for this model as well (see Section 4.2). We also extend it to handle larger locality $d = \text{polylog}(n)$, under some additional restrictions on m and the advantage ε of the distinguisher (see Section 4.1). Finally, we can also handle noisy predicates where independent Bernoulli noise is added to the output (see Section 4.3).

1.2 Overview of Techniques

In this subsection, we provide an overview of the techniques used to perform our search-to-decision reduction, starting with the basic definitions. For the rest of the section, fix some d-ary predicate P with $\eta = \Pr_x [P(x) = 1]$.

Notation Given a binary string x, we use x_i to denote the i-th bit in the binary string. Denote by $G_{n,m,d}$ the set of all hypergraphs with n vertices and m ordered hyperedges, with d vertices in each hyperedge. For a distribution D, we denote $x \leftarrow D$ to indicate that x is sampled from D; if D is a set, this denotes sampling uniformly from the set. For any $\eta \in [0, 1]$, we denote by $Bern(\eta)$ the Bernoulli distribution with parameter η. We say that a distinguishing algorithm D has advantage ε in the decision problem with predicate P if:

$$\Pr_{\substack{G \leftarrow G_{n,m,d} \\ s \leftarrow \mathbb{F}_2^n}} [\mathsf{D}(G, f_{G,P}(s)) = 1] - \Pr_{\substack{G \leftarrow G_{n,m,d} \\ b \leftarrow Bern(\eta)^m}} [\mathsf{D}(G, b) = 1] \geq \varepsilon$$

Reduction Idea The high-level idea of our reduction is as follows. We start with a distinguisher D with advantage ε as above. We use it to construct a set of predictor algorithms $\mathsf{S}_2, \ldots, \mathsf{S}_n$, where each S_i has a small advantage $\Omega(\varepsilon/t)$ in predicting the value of $s_1 \oplus s_i$, for some $t = O(n \log(n/\varepsilon))$. Further, we show that for an $\Omega(\varepsilon)$ fraction of secrets s, all of the S_i's simultaneously express this advantage. Whenever such a secret happens to be selected, we can then amplify all these advantages using $O((t/\varepsilon)^2 \log n)$ independent samples and learn all the parities $s_1 \oplus s_i$ with high confidence, thus recovering s itself. Altogether, the search algorithm needs $O((n/\varepsilon)^2 \log^3(n/\varepsilon))$ independent instances, each with m outputs, and succeeds with probability $\Omega(\varepsilon)$, as stated in Theorem 1.3.

Predictor We will first provide more details on the construction of the predictors. Suppose we have a decision algorithm D that has advantage ε. Our goal will be to construct a set of algorithms S_i, such that for a large enough fraction of secrets $s \in \mathbb{F}_2^n$, there is a pair of numbers $eq_s, neq_s \in [0, 1]$ where $eq_s > neq_s$, such that the following holds for all $i \in [2, n]$:

- If $s_1 = s_i$, then $\Pr[\mathsf{S}_i(G, f_{G,P}(s)) = 1] = eq_s$
- If $s_1 \neq s_i$, then $\Pr[\mathsf{S}_i(G, f_{G,P}(s)) = 1] \leq neq_s$

As long as the gap between eq_s and neq_s is not negligible, we can tell the relationship between s_1 and s_i by running the algorithm S_i polynomially many times with independent problem instances with the same secret s.

Each S_i, on input some (G, y), functions by applying a randomised transformation T (described later) to the hypergraph G. This transformation is designed such that if $s_1 = s_i$, $(T(G), f_{G,P}(s))$ looks more like a random local function instance $(G, f_{G,P}(s))$, and if $s_1 \neq s_i$, it looks more like (G, b) where $b \leftarrow Bern(\eta)^m$. Next, the transformed problem instance $(T(G), y)$ is fed to the decision algorithm D, and S_i outputs whatever D does. We will show using a hybrid argument that, for any secret s for which D still has $\Omega(\varepsilon)$ advantage in distinguishing between $(G, f_{G,P}(s))$ and (G, b), the gap between eq_s and neq_s is at least $\Omega(\varepsilon/t)$, where $t = O(n \log(nm/\varepsilon)) = O(n \log(n/\varepsilon))$ (since $m = poly(n)$).

Transformation The key to our improvement comes from the transformation. Here, we will show how we achieve the gap of ε/t between eq_s and neq_s where $t = O(n \log(nm/\varepsilon))$.

The transformation $T_{a,b} : G_{n,m,d} \to G_{n,m,d}$ is a randomized function parameterised by two values $a, b \in [n]$ that takes in a hypergraph $G = (S_i)_{i\in[m]}$ with $S_i = (j_1, \ldots, j_d)$ and returns another hypergraph $G' = (S'_i)_{i\in[m]}$ with $S'_i = (j'_1, \ldots, j'_d)$, where the distribution of each of the j'_k is given as follows

$$\Pr[j'_k = j_k] = 1 \quad \text{if } j_k \notin \{a, b\}$$

$$\Pr[j'_k = a] = \frac{1}{2},\ \Pr[j'_k = b] = \frac{1}{2} \quad \text{if } j_k \in \{a, b\}$$

Essentially, if a vertex in the hyperedge is not a or b, then it will remain. Otherwise, the vertex will change to either a or b with probability half. It is easy to see that the uniform distribution over $G_{n,m,d}$ is stable under this process, i.e the uniform distribution of $G_{n,m,d}$ is the same as the distribution of uniformly sampling from $G_{n,m,d}$ and then applying the transformation.

Observe that if $s_a = s_b$, then the output of the random local function remains the same after the transformation: $f_{G,P}(s) = f_{T_{a,b}(G),P}(s)$. The transformation does nothing to the distribution of $(G, f_{G,P}(s))$ when G is uniformly distributed. Using $\approx$ to denote similarity between distributions, in this case, we have:

$$(T(G), f_{G,P}(s)) = (T(G), f_{T(G),P}(s)) \approx (G, f_{G,P}(s))$$

On the other hand, when $s_a \neq s_b$, then in general $f_{G,P}(s)$ is not the same as $f_{T_{a,b}(G),P}(s)$, and the relationship between $T(G)$ and $f_{G,P}(S)$ becomes less coupled than between G and $f_{G,P}(s)$. In fact, we can show that after we apply about $t = O(n \log(nm/\varepsilon))$ such transformations, the transformed hypergraph will become independent of $f_{G,P}(s)$.

Precisely, we show that for any hypergraph G, the distribution of $T_{a_t,b_t} \circ \ldots \circ T_{a_1,b_1}(G)$ is close to uniformly sampling from $G_{n,m,d}$. One can think of the transformation as a Markov process, and $O(n \log(nm/\varepsilon))$ is the mixing time of this process.[2] This implies that with an independently sampled $G' \leftarrow G_{n,m,d}$,

$$(T_{a_t,b_t} \circ \ldots \circ T_{a_1,b_1}(G), f_{G,P}(s)) \approx (G', f_{G,P}(s))$$

To see why this might be true, note that after about $t = \Omega(n \log(nm/\varepsilon))$ such randomly chosen transformations, every vertex in every hyperedge of G would have been touched by at least a few of the transformations. As the random assignment in $T_{a,b}$ happens independently for each a or b that appears as a vertex, this process quickly randomises the whole hypergraph.

We can, in fact, show an even stronger result: after applying $t = O(n \log(nm/\varepsilon))$ transformations, the transformed instance $(T(G), f_{G,P}(s))$ resembles an instance (G, b), where b is sampled from $Bern(\eta)^m$. The main task here is to prove $f_{G,P}(s) \approx Bern(\eta)^m$ when G is sampled at random. Notice that as G itself is not revealed (being completely hidden by $T(G)$), each bit of $f_{G,P}(s)$ is simply an independent output of P when its inputs are chosen to be random bits from s. The bias of this output depends heavily on the Hamming weight of s.

If s is extremely biased towards 1 or 0, one might be able to distinguish between these $f_{G,P}(s)$ and b with just a small output size m. Fortunately, for a randomly sampled s, the Hamming weight is usually fairly balanced, around $n/2 \pm O(\sqrt{n})$, and the bias of P on this input distribution remains close to η. We then claim that, conditioned on s being fairly balanced, one will need a large number of output bits to distinguish between $f_{G,P}(s)$ and b. And when the number of outputs reaches a point where this distinguishing is possible, both the search and decision problems turn out to be easy due to known algorithms from [App16].

Therefore, when m is not so large that the search problem is already easy, applying t transformations to the hypergraph G makes the joint distribution of the transformed hypergraph and $f_{G,P}(s)$ look like (G, b) where $b \leftarrow Bern(\eta)^m$. So when we apply the transformation $T_{a,b}$ on $s_a \neq s_b$, the distribution is indeed a step closer to the (G, b) distribution. The algorithm S_i, on input (G, y), applies a random number $r \leftarrow [0, t-1]$ of transformations $T_{a,b}$ to G with random a, b, and then finally applies the transformation $T_{1,i}$ once. Then, by a straightforward hybrid argument, we obtain the properties of S_i listed earlier, including that the gap between eq_s and neq_s is at least $\Omega(\varepsilon/t)$.

There are a few issues that arise from the possibility of the distinguisher's behaviour being correlated with the secret that we have glossed over here. These are readily dealt with by simple transformations like permuting the secret first, and are accounted for in the actual proof.

1.3 Related Work

Security of Random Local Functions Goldreich [Gol11] originally proposed using a randomly chosen predicate, while cautioning against linear, degenerate, or otherwise structured predicates that lead to easily solvable equation systems. It is known that myopic and drunken backtracking algorithms (algorithms that only look at the instance locally each time they make a decision; this includes many powerful SAT solvers) do not perform well on predicates with some linear component

[2]The proof of this claim is inspired by the convergence of randomised gossip algorithms [BGPS06].

$P(x_1, \ldots, x_d) = x_1 + \cdots + x_k + Q(x_{k+1}, \ldots, x_d)$ for an arbitrary Q with $k > 3d/4$ for some large enough d on n variables and n outputs [AHI05, CEMT09, Its10]. For the XOR-AND$_{3,2}$ predicate $P(x_1, \ldots x_5) = x_1 + x_2 + x_3 + x_4 x_5$, [OW14] show pseudorandomness up to output length $n^{1.499}$ against $\mathbb{F}_2$-linear tests and a wide class of semi-definite programming algorithms. The work of [ABR12] showed a dichotomy – for output length $n^{1+\delta}$, every choice of the predicate results in the random local function that either is secure against linear tests with high probability, or is insecure with high probability.

On the negative side, [BQ09] presented attacks against predicates that exhibit strong correlation with one or two input variables. [AL16] showed necessary conditions on the predicate P to prevent certain families of algebraic attacks. For a random local function with predicate P, and output length $m = n^s$, they show that it is necessary to be $\Omega(s)$-resilient (s-resilient means that P is uncorrelated with any s-subset of its input) and have an algebraic degree of $\Omega(s)$ even after fixing $\Omega(s)$ of its inputs. In particular, they show that for all ℓ, k, the XOR-AND$_{\ell,k}$ predicate suggested by [OW14] is not pseudorandom with $n^{2.01}$ outputs. They then suggest an alternative candidate, XOR-MAJ$_{\ell,k}$. [App16, Corollary 3.6] showed that regardless of the choice of predicate P, observing more than $\Omega(n^{\frac{1}{2}\lfloor 2d/3 \rfloor} \log n)$ outputs gives an efficient algorithm to recover the input.

[CDM$^+$18] looked at the concrete security of Goldreich's function and developed a sub-exponential time attack and analysed the efficiency of algebraic attacks, such as Gröbner basis methods. [YGJL22] built on their work, improving the time complexity. [COST19] showed that the assumption of the hypergraph being a good expander is not sufficient; the neighbour function of the graph must also have high circuit complexity.

More recently, [DMR23] investigated the problem of designing predicates that simultaneously achieve high resilience and high algebraic immunity. A predicate with high algebraic immunity guarantees a high algebraic degree after fixing some inputs, though the converse does not necessarily hold [AL16]. [DMR23] conjectured the existence of such optimal predicates and demonstrated that the commonly used XOR-MAJ predicate does not satisfy this optimality criterion. Furthermore, through experiments, they identified optimal predicates for localities up to 12.

On attacking local PRGs, it is known that there exists a subexponential time algorithm for distinguishing superlinear stretch local PRGs with noticeable probability [AIK06, Zic17, Üna23]. A related variant of local PRG is also studied by [ABCM25], where the input string is not uniformly sampled. Assuming the hardness of sparse LPN, they showed that superlinear stretch with negligible distinguishing advantage is possible.

Local Cryptography Cryan and Miltersen [CM01] initiated the study of local PRGs, where they showed the impossibility of a PRG in NC^0_3 (circuit with depth of 3) with superlinear stretch. Mossel et al. [MST03] then extended the impossibility to NC^0_4.

On the positive side, Applebaum et al. [AIK06] proved the existence of an OWF and a sublinear stretch PRG in NC^0, assuming the existence of an OWF and a PRG in NC^1 (logarithmic depth circuits), which follows from well-established cryptographic assumptions such as the hardness of lattice problems. The same authors [AIK08] further showed the existence of a linear-stretch local PRG assuming the hardness of the average case MAX-3LIN problem [Ale03], which is related to the sparse LPN problem.

Goldreich [Gol11] proposed the local OWF candidate discussed earlier in this section, based on random local functions. It is known that under this assumption, one can obtain a polynomial stretch local PRG from the work of [App12, AK19] and locally computable universal one-way hash functions with linear shrinkage [AM13]. Other related work includes the hardness amplification results for local OWFs by [BR11].

Organization

In Section 2, we describe preliminaries and formally define the terms that we will use throughout the paper. Section 3 will be on the proof of our main search-to-decision reduction theorem. Section 4 will discuss the generalisation of our theorem to other interesting families of problems. Finally, in Section 5, we include deferred proofs and additional details from the other sections.

2 Preliminaries

Notation. Given a binary string x, we use x_i to denote the i-th bit in the binary string. Given two distributions D_1, D_2, If the two distributions are equivalent, we write $D_1 \approx D_2$. If $x \leftarrow D_1$, this means x is sampled from D_1. Here, D_1 can also be a set, in which case this denotes sampling uniformly from the set. We denote a finite field of size p as $\mathbb{F}_p$ where p is a prime. For any $\eta \in [0,1]$, we denote by $Bern(\eta)$ the Bernoulli distribution with parameter η.

Definition 2.1 (Statistical Distance)**.** *Consider two distributions D_1, D_2 defined over space Q, the statistical distance (Total Variation Distance) is defined as*

$$\Delta(D_1, D_2) = \frac{1}{2}\sum_{x\in Q} |\Pr[D_1 = x] - \Pr[D_2 = x]|$$

Definition 2.2 (Hamming weight)**.** *Denote the Hamming weight of a binary string x as $wt(x)$.*

Definition 2.3 (d-ary)**.** *For $d \in \mathbb{N}$, a* d-ary *predicate P is a function $P : \mathbb{F}_2^d \to \mathbb{F}_2$ (that is, it takes d elements as input).*

Definition 2.4 (Bias)**.** *Suppose a* d-ary *predicate P, the* bias *of P is defined as $\mathbf{E}_{x\leftarrow\mathbb{F}_2^d}[P(x)] = \Pr_{x\leftarrow\mathbb{F}_2^d}[P(x) = 1]$. We commonly denote the bias as $\eta \in [0,1]$*

Definition 2.5 (Bounded Bias)**.** *Suppose a* d-ary *predicate P with bias η, we say that the bias is bounded if there exist two constants c_1, c_2 such that $0 < c_1 \leq c_2 < 1$ and $\eta \in [c_1, c_2]$*

Definition 2.6 (c-correlated, [App16, Section 3.2])**.** *We say that a non-constant predicate P is* c-correlated *if c is the minimal positive integer such that it is correlated with the parity of a cardinality-c subset of its inputs. More formally,*

$$\Pr[P(x) = \sum_{i\in T} x_i] \neq \frac{1}{2}$$

for some subset T with cardinality c.

Definition 2.7 (Hypergraph)**.** *For $n, m, d \in \mathbb{N}$, an* (n, m, d)-hypergraph *is a hypergraph on n vertices with m hyperedges, where each hyperedge S_i is an ordered tuple $S_i = (j_1, \ldots, j_d) \in [n]^d$. We commonly write a hypergraph as $G = (S_i)_{i\in[m]}$*

Note that this definition of hypergraph allows repeated values in the hyperedges. We denote the set of all (n, m, d)-hypergraphs as $G_{n,m,d}$, and also use this symbol to denote the uniform distribution over this set.

Definition 2.8 (d-local function)**.** *Consider a d-ary predicate P and a (n, m, d)-hypergraph G, a* d-local function *is a function $f_{G,P} : \mathbb{F}_2^n \to \mathbb{F}_2^m$ such that given input x, the i-th output is defined as*

$$f_{G,P}(x)_i = P(x_{j_1}, \dots, x_{j_d})$$

where $S_i = (j_1, \dots, j_d)$ is the i-th hyperedge of G.

Informally, the i-th hyperedge decides which input indices are applied to the predicate to compute the i-th output.

Definition 2.9 (Decision Problem)**.** *Consider a d-ary predicate P with bias η and the uniform hypergraph distribution $G_{n,m,d}$. An algorithm D is said to have advantage $\varepsilon \in [0, 1]$ in solving the* Decision Problem for (P, n, m)*, if the following holds:*

$$\left| \Pr_{\substack{G \leftarrow G_{n,m,d} \\ s \leftarrow \mathbb{F}_2^n}} [\mathsf{D}(G, f_{G,P}(s)) = 1] - \Pr_{\substack{G \leftarrow G_{n,m,d} \\ b \leftarrow Bern(\eta)^m}} [\mathsf{D}(G, b) = 1] \right| \geq \varepsilon$$

In other words, the algorithm can distinguish the output of the d-local function from a random binary string of length m that has the same bias as the predicate P. The corresponding hypergraph of the local function is randomly sampled from $G_{n,m,d}$, and the input is randomly sampled from $\mathbb{F}_2^n$.

Remark 2.10. Suppose a d-predicate P with bias η, $G \leftarrow G_{n,m,d}, s \leftarrow \mathbb{F}_2^n, b \leftarrow Bern(\eta)^m$. The distribution $(G, f_{G,P}(s))$ is called the *planted distribution* while (G, b) is called the *null distribution.*

Definition 2.11 (Search Problem)**.** *Consider a d-ary predicate P and the uniform hypergraph distribution $G_{n,m,d}$. An algorithm S is said to have success probability $\varepsilon \in [0, 1]$ in solving the* Search Problem for (P, n, m) *if the following holds:*

$$\Pr_{\substack{G \leftarrow G_{n,m,d} \\ s \leftarrow \mathbb{F}_2^n}} \left[\mathsf{S}(G, f_{G,P}(s)) = s', \text{ such that } f_{G,P}(s) = f_{G,P}(s')\right] \geq \varepsilon$$

Remark 2.12. To simplify notation, we assume that the description of the predicate used P is public knowledge. All algorithms have access to the description of P. Also, all predicates P are assumed to be non-constant, as the search problem on constant predicates is trivial.

Lemma 2.13 (Sample threshold, [App16, Theorem 3.5])**.** *Given a c-correlated d-ary predicate P, where $d = O(1)$, there exists a polynomial-time algorithm for both the Search and Decision Problems for (P, n, m) for some $m = O(n^{c/2} + n \log n)$ with success probability and advantage, respectively, that is $1 - o(1)$.*

Proof Sketch. Suppose T is a minimal subset of size c that is correlated to the predicate. Since $d = O(1)$, the correlation with the parity is a constant. Without loss of generality,

$$\Pr[P(x) = \sum_{i \in T} x_i] - \frac{1}{2} \geq 2^{-d} = \Omega(1)$$

Each output of the d-local function can then be written as parity sum + noise,

$$f_{G,P}(x)_j = \sum_{i \in T} x_i + e$$

where e is a Bernoulli random variable. A constant correlation also implies that the Bernoulli parameter is constant. An algorithm to solve the search problem is constructed by reducing the problem to a c-sparse noisy linear system. Having $m = \Omega(n^{c/2})$ gives roughly $\Omega(n)$ pairs of equations where $c-1$ variables are the same, while the rest of the variables are disjoint. Each pair then has its equations combined to obtain 2-sparsity noisy linear equations with constant error rate. Since there exists an efficient algorithm to solve a 2-sparsity noisy linear system with a constant error rate, we are done. □

3 Search-to-Decision Reduction

Theorem 3.1. *Consider a d-ary predicate P, with $d = O(1)$. Suppose there is a polynomial-time algorithm that has advantage $\varepsilon \in [0,1]$ in solving the Decision Problem for (P, n, m) for some $m = poly(n)$. Then there is a polynomial-time algorithm for the Search Problem for $(P, n, \ell m)$ that has success probability $\Omega(\varepsilon)$, for some $\ell = \Theta((n/\varepsilon)^2 \log^3(n/\varepsilon))$.*

The idea of the proof is that we first show there exist $t = \Omega(n \log(n/\varepsilon))$ hybrids $H_0, \ldots, H_t$ where H_0 hybrid has the same distribution as the planted distribution and H_t distribution has a small statistical distance from the null distribution. Then we construct a predictor on the value of s_i where $i \in [2, n]$ by applying a transformation to the H_i hybrid such that when $s_1 = s_i$, the transformed distribution will be H_i hybrid, and H_{i+1} otherwise. Using the hybrid argument, this will give a ε/t advantage in predicting the value of $s_1 \oplus s_i$. This prediction can then be amplified with $\Theta((t/\varepsilon)^2 \log(n/\varepsilon))$ repetition and guessing the value of s_1 to recover the secret s.

Suppose that P is c-correlated, we can then assume that $m = o(n^{c/2-2})$ as if not then both Search and Decision for distribution $G_{n,k\times m,d}$ will become easy due to Lemma 2.13, making our theorem statement trivially true. The same reasoning implies that we can assume $\varepsilon = w(n^{-c/4})$. This assumption will be used in Claim 3.10

The rest of the section will cover the proof of this theorem. It is split into 3 parts: the definition of the hybrids and proofs of their properties, the construction of the predictor algorithm, and the amplification of its advantage. Throughout the proof, fix any d-ary predicate P for any constant d.

3.1 Hybrids

In this subsection, we will define the hybrids H_i and prove that the H_0 hybrid has the same distribution as the planted distribution and the H_t distribution has a small statistical distance from the null distribution. First, we start by explaining the hybrid argument.

Claim 3.2 (Hybrid Argument)**.** *Suppose there are t distributions $H_0, \ldots, H_t$ such that an algorithm D has advantage ε in distinguishing H_0 and H_t. Then the algorithm can distinguish H_i and H_{i+1} with advantage ε/t on a randomly chosen i. Formally,*

$$\left| \Pr_{i \leftarrow [0,t-1]}[D(H_i) = 1] - \Pr_{i \leftarrow [0,t-1]}[D(H_{i+1}) = 1] \right| \geq \varepsilon/t$$

Definition 3.3 (Permuted Hypergraph)**.** *For a permutation π on $[n]$ and a hypergraph $G = (S_i)_{i\in[m]}$, we define the permuted hypergraph $\pi(G) = (S_i')_{i\in[m]}$ as having the values in the hyperedges permuted with π. i.e. Suppose $S_i = (j_1, \ldots, j_d)$, then*

$$S_i' = (\pi(j_1), \ldots, \pi(j_d))$$

Next, we define a randomised transformation on a hypergraph $G = (S_i)_{i\in[m]}$ parameterised by two values $a, b \in [n]$, such that the member value in each hyperedge S_i will remain the same if it is not a and not b. Otherwise, it will switch to a or b with probability half.

Definition 3.4 (Transformation)**.** *Define a* randomized transformation $T_{a,b} : G_{n,m,d} \to G_{n,m,d}$ *as follows: On inputting a hypergraph G, each of the hyperedges $S_i = (j_1, \ldots, j_d)$ is transformed to $S_i' = (j_1', \ldots, j_d')$ independently where*

$$\Pr[j_k' = j_k] = 1 \quad \text{if } j_k \notin \{a, b\}$$

$$\Pr[j_k' = a] = \frac{1}{2},\ \Pr[j_k' = b] = \frac{1}{2} \quad \text{if } j_k \in \{a, b\}$$

The hybrid is then defined by repeatedly applying this transformation to the permuted input hypergraph.

Definition 3.5 (Hybrid)**.** *Given a secret s, define the i-*th hybrid distribution H_i^s *as applying i many randomly selected transformations to the permuted randomly sampled hypergraph of a d-local function.*

$$H_i^s = (T_{a_i,b_i} \circ \ldots \circ T_{a_1,b_1}(\pi(G)), f_{G,P}(s))$$

where $G \leftarrow G_{n,m,d}$, π is a randomly sampled permutation on $[n]$, and for each $j \in [i]$, $a_j, b_j \leftarrow [n]$.

Now we show that the terminal hybrids H_0 and H_t are similarly distributed to the planted and the null distributions, respectively.

Definition 3.6 (ε-fairly balanced secret)**.** *For $\varepsilon \in [0, 1]$, we say that a secret $s \in \mathbb{F}_2^n$ is* ε-fairly balanced *if the Hamming weight $wt(s) \in [n/2 - w, n/2 + w]$ where $w = 2\sqrt{n}\log(1/\varepsilon)$.*

Remark 3.7. By Chernoff bound, the probability that a randomly sampled secret $s \leftarrow \mathbb{F}_2^n$ is not ε-fairly balanced is at most $o(\varepsilon)$.

Lemma 3.8 (Terminal Hybrid)**.** *Suppose $d = O(1)$, $m = o(n^{c/2-2})$ and $\varepsilon = \omega(n^{-c/4})$, consider any c-correlated d-ary predicate P with bounded bias η, and any ε-fairly balanced fixed $s \in \mathbb{F}_2^n$. With the sampling $s' \leftarrow \mathbb{F}_2^n$ conditioned on $wt(s) = wt(s')$ and $G \leftarrow G_{n,m,d}$, we have the following:*

1. *The distribution of H_0^s is identical to $(G, f_{G,P}(s'))$.*
2. *The distribution of H_t^s has statistical distance at most $\varepsilon/4$ from (G, b), where $b \leftarrow Bern(\eta)^m$, for some $t = O(n\log(nm/\varepsilon)) = O(n\log(n/\varepsilon))$.*

Proof. For H_0^s, the only changes made are that a randomly sampled permutation is applied to the hypergraph. However, since the hypergraph is randomly sampled and s' is sampled such that it has the same Hamming weight as s, it is identically distributed to $(G, f_{G,P}(s'))$.

$$H_0^s = (\pi(G), f_{G,P}(s)) = (\pi(G), f_{\pi(G),P}(\pi(s))) \approx (G, f_{G,P}(\pi(s))) \approx (G, f_{G,P}(s'))$$

For H_t^s, we claim that no matter the initial starting hypergraph G, after applying the transformations to G, it will become distributed like a randomly sampled hypergraph. Formally,

Claim 3.9 (Random Graph)**.** *Given any $G \in G_{n,m,d}$, if $a_j, b_j \leftarrow [n]$, $m = poly(n)$ and for some $t = O(n \log(nm/\varepsilon))$, then the statistical distance between $T_{a_t,b_t} \circ \ldots \circ T_{a_1,b_1} \circ G$ and the uniform distribution over $G_{n,m,d}$ is at most $\varepsilon/8$.*

Proof Sketch. Intuitively, each hyperedge will get more randomised when a random transformation is applied. Once we apply $t = \Omega(n \log(nm/\varepsilon))$ random transformations, every hyperedge in the hypergraph would have been touched by a few of the transformations, and would look like a randomly sampled one. This will then imply the distribution of $T_{a_t,b_t} \circ \ldots \circ T_{a_1,b_1} \circ G$ and the uniform distribution from $G_{n,m,d}$ will be close. An alternative view is to think of the transformations as a Markov process, then $t = \Omega(n \log(nm/\varepsilon))$ is the mixing time of the Markov Chain. The details of the proof are deferred to Section 5.1 □

Claim 3.9 implies that H_t^s is similarly distributed as $(G', f_{G,P}(s))$ where $G' \leftarrow G_{n,m,d}$ and is unrelated to the output of the local function. Now, we would like to show that the output of the local function is statistically close to a random binary string with the same bias η.

Claim 3.10 (Random Output)**.** *Suppose s is ε-fairly balanced and $G \leftarrow G_{n,m,d}$, $m = o(n^{c/2-2})$, $\varepsilon = \omega(n^{-c/4})$, $d = O(1)$ and a c-correlated d-ary predicate P with bounded bias η, then the statistical distance between $f_{G,P}(s)$ and $Bern(\eta)^m$ is at most $\varepsilon/8$.*

Proof Sketch. Suppose that the secret s is perfectly balanced, then since each output of the d-local function is independent conditioned on the secret, the distribution of $f_{G,P}(s)$ (for random G) is exactly the same as $Bern(\eta)^m$. Using the ε-fairly balanced assumption on s, we know that the bias towards 1 or 0 is bounded by $1/2 + 2\log(1/\varepsilon)/\sqrt{n}$

Say the bias towards 1 is $1/2+\alpha$. From knowing that the predicate P is c-correlated, we can then show that $\mathbf{E}_{x_i \leftarrow D}\left[P(x_1,\ldots,x_d)\right] - \eta = O(\alpha^c)$. Then, using KL divergence and Pinsker's inequality, it can be shown that one requires $\Omega(n^{c/2})$ samples to distinguish the output from $Bern(\eta)^m$. But having so many samples also means that both search and decision become efficient to solve. In particular, we use the assumption that $m = o(n^{c/2-2})$, $\varepsilon = \omega(n^{-c/4})$, to show that the statistical distance is $o(\varepsilon)$. The details are deferred to Section 5.2. □

Combining both claims, we get an implication that for $t = \Omega(n \log(nm/\varepsilon))$ with a large enough constant, H_t^s has statistical distance at most $\varepsilon/4$ from (G, b) where $b \leftarrow Bern(\eta)^m$ □

The next claim says that if the statistical distance is small, the decision algorithm can still be used to distinguish, even when the distribution is not exactly the same.

Claim 3.11. *Given D for Decision Problem with advantage $\varepsilon \in [0,1]$. If the distribution $null'$ is taken from a distribution that has statistical distance with the null distribution of at most ε/k for constant $k > 1$, then D can distinguish planted and $null'$ with advantage at least $\Omega(\varepsilon)$.*

$$\left|\Pr\left[\mathsf{D}(planted) = 1\right] - \Pr\left[\mathsf{D}(null') = 1\right]\right| \geq \varepsilon - \varepsilon/k = \Omega(\varepsilon)$$

Proof. This is because all algorithms can only distinguish $null'$ and $null$ with an advantage of at most the statistical distance. □

3.2 Predictor

In this subsection, we will describe the construction of a predictor that can predict the value of $s_1 \oplus s_i$ with high probability. The predictor internally constructs hybrids and decides based on the outputs of the distinguisher under these hybrids.

First, we will address the technical challenge that the decision algorithm may just fail to work for some secret s. Since we perform a permutation on the hypergraph, the Hamming weight of the secret s is fixed throughout the reduction. If we are unlucky and get a "bad" Hamming weight that the algorithm always fails, then recovery is impossible. Nevertheless, it can be shown that there is still a sufficiently large number of secrets (parameterised by ε) which are good to distinguish.

Definition 3.12 (Good Secret)**.** *For a distinguishing algorithm D with advantage ε, define $\mathcal{G}_{\mathsf{D}} \subseteq \mathbb{F}_2^n$ to be the set of secrets whose Hamming weight is good to distinguish. i.e.*

$$\mathcal{G}_{\mathsf{D}} = \left\{ s \in \mathbb{F}_2^n \;\middle|\; \Pr_{s' \leftarrow \mathbb{F}_2^n, wt(s)=wt(s')} \left[\mathsf{D}(G, f_{G,P}(s')) = 1\right] - \Pr[\mathsf{D}(G, b) = 1] \geq \varepsilon/2 \right\}$$

where $G \leftarrow G_{n,m,d}, b \leftarrow Bern(\eta)^m$, η is the bias of P.

Note that $\mathcal{G}_D$ is essentially a collection of binary strings that has some Hamming weights. If $wt(s) = wt(s')$ and $s \in \mathcal{G}_D$, then $s' \in \mathcal{G}_D$.

Claim 3.13. *Suppose D has advantage ε, then $|\mathcal{G}_{\mathsf{D}}| \geq \varepsilon/2 \cdot (2^n)$.*

Proof Sketch. Proven using Markov's inequality, the details are deferred to Section 5.3. □

In the next lemma, we describe the predictor algorithm for s_i.

Lemma 3.14. *Consider any d-ary predicate P with bounded bias η, and suppose there is a polynomial-time algorithm D that has advantage ε to solve the Decision Problem for (P, n, m). Then there is a set of polynomial-time algorithms $\{\mathsf{S}_2, \ldots, \mathsf{S}_n\}$ such that for all ε-fairly balanced and good $s \in \mathcal{G}_D$, there is a pair of numbers $eq_s, neq_s \in [0, 1]$ and the algorithms behave as follows for each $i \in [2, n]$:*

- *If $s_1 = s_i$, then $\Pr[\mathsf{S}_i(G, f_{G,P}(s)) = 1] = eq_s$*
- *If $s_1 \neq s_i$, then $\Pr[\mathsf{S}_i(G, f_{G,P}(s)) = 1] \leq neq_s$*

Further, for some $t = O(n \log(mn/\varepsilon))$, the following holds for all such s:

$$eq_s - neq_s \geq \varepsilon/4t$$

Proof. Fix any ε-fairly balanced and good $s \in \mathcal{G}_D$, without loss of generality, we can remove the absolute value on the advantage of D. i.e. $\Pr[\mathsf{D}(planted) = 1] - \Pr[\mathsf{D}(null) = 1] \geq \varepsilon/2$. Then, the following corollary is immediate from the previous lemmas.

Corollary 3.15. *There exists $t = O(n \log(nm/\varepsilon))$ such that:*

$$\Pr_{r \leftarrow [0,t-1]}[\mathsf{D}(H_r^s) = 1] - \Pr_{r \leftarrow [0,t-1]}[\mathsf{D}(H_{r+1}^s) = 1] \geq \varepsilon/4t$$

Proof. Combine the statements from good secret Definition 3.12, hybrid argument Claim 3.2, terminal hybrid Lemma 3.8 and statistical distance Claim 3.11 □

Now we describe the predictor algorithm S_i. On input $(G, f_{G,P}(s))$ and given blackbox access to D, it acts as follows:

```
Algorithm $\mathsf{S}_i(G, f_{G,P}(s))$:

  1. Sample a random number $r \leftarrow [0, t-1]$.
  2. Sample a random permutation $\pi$ on $[n]$.
  3. Get $H_r^s$ by performing $r$ hybridization steps on $G$, i.e.
     $H_r^s = (T_{a_r,b_r} \circ \ldots \circ T_{a_1,b_1}(\pi(G)), f_{G,P}(s))$, where $a_j, b_j \leftarrow [n]$.
  4. Apply $T_{\pi(1),\pi(i)}$ to $H_r^s$ to obtain $H$.
  5. Return $\mathsf{D}(H)$.
```

Note that the specification of S_i does not depend on s itself. From the described algorithm, we know that $\mathsf{S}_i(G, f_{G,P}(s)) = D(H)$. To use Corollary 3.15, we would like to show that when $s_1 = s_i$, the distinguisher's output on H_r^s remains similar even if the input is replaced with H. For the case of $s_1 \neq s_i$, the comparison is with H_{r+1}^s instead.

Claim 3.16 (Equal)**.** *If $s_1 = s_i$, then $H \approx H_r^s$.*

Proof Sketch. When $s_1 = s_i$, this implies $\pi(s)_{\pi(1)} = \pi(s)_{\pi(i)}$, which means that the transformation $T_{\pi(1),\pi(i)}$ on the hypergraph does not affect the output of the d-local function. Essentially, no effective randomisation is performed, and the distribution does not look more like the null distribution. Therefore we are able to conclude that $H \approx H_r^s$. Details are deferred to Section 5.4 □

Claim 3.17 (Not Equal)**.** *If $s_1 \neq s_i$, $\Pr[\mathsf{D}(H_{r+1}^s) = 1] \geq \Pr[\mathsf{D}(H) = 1]$.*

Proof Sketch. When $s_1 \neq s_i$, this implies $\pi(s)_{\pi(1)} \neq \pi(s)_{\pi(i)}$. Therefore, only effective randomisation is performed. The distribution H looks more like the null distribution than H_{r+1}^s, because the last transformation $T_{a,b}$ on H_{r+1}^s could be effective or not, depending on the choice of a, b. Since the distinguisher tends to return 1 less often when given the null distribution, when H is given as an input, it should return 1 less often than H_{r+1}^s. See Section 5.5 for details. □

As noted above, $\Pr[\mathsf{S}_i(G, f_{G,P}(s)) = 1] = \Pr[D(H) = 1]$. Set $eq_s = \Pr[\mathsf{D}(H_r^s) = 1]$ and $neq_s = \Pr[\mathsf{D}(H_{r+1}^s) = 1]$. Combining Corollary 3.15 and the two claims above then proves the claimed behaviour of the S_i's, and also yields $eq_s - neq_s \geq \varepsilon/4t$. □

3.3 Amplification

Now we will show how to amplify the predictor's advantage for each $i \in [2, n]$, and eventually recover s. To use the predictor, we first need to guess the value of s_1. Since there are only 2 possible values, 0 and 1, we can just try both values. This will ultimately give two candidates as the solution to the problem. The correct one can then be verified by re-evaluating the d-local function on the candidates.

Next, as the S_i's only guarantee a gap in the response based on the relationship of s_1 and s_i, to correctly identify the relationship with high confidence, we will need to estimate the response of S_i when $s_1 = s_i$. More precisely, the following value needs to be estimated

$$eq_s = \Pr_r[D(H_r^s) = 1]$$

Furthermore, the value eq_s could vary based on the value of s. Fortunately, since for secrets s with the same Hamming weight, we know that their eq_s is the same, as in our construction of the hybrids the first step is to effectively permute the secret. We just have to perform this estimation for one secret for each possible Hamming weight.

The estimation can be done by locally generating a secret of a given Hamming weight and a random problem instance, then applying hybridisation and sending it to the distinguisher D. With $O((t/\varepsilon)^4 \log(n/\varepsilon))$ number of trials, by Hoeffding's inequality, the average will converge to within $o(\varepsilon/t)$ of the true value of eq_s, with failure probability $o(\varepsilon)$. Once we have these estimates, we run the computations below with each of them until the correct one is used and the secret s is found.

Given the value of eq_s, fix some $i \in [2, n]$. We can amplify the prediction probability by rerunning the predictor $\mathsf{S}_i(G, f_{G,P}(s))$ with a fresh new problem instance of the same secret s, with an independent new G. Suppose l repetitions are performed, take the sum of the outputs of S_i and put a threshold at $l(eq_s - \varepsilon/8t)$. If it is greater than that, return s_1 as the guess for s_i; otherwise, return the flip of s_1. Formally, let X be the random variable of the sum of the output of the distinguisher.

If $s_1 \neq s_i$, $\mathbf{E}[X] \leq l(eq_s - \varepsilon/4t)$, by Hoeffding's bound

$$\Pr[X \geq l(eq_s - \varepsilon/8t)] \leq \exp(-O(l\varepsilon^2/t^2))$$

If $s_1 = s_i$, $\mathbf{E}[X] = l \cdot eq_s$, by Hoeffding's bound

$$\Pr[X \leq l(eq_s - \varepsilon/8t)] \leq \exp(-O(l\varepsilon^2/t^2))$$

Therefore, the failure probability is at most $\exp(-O(l\varepsilon^2/t^2))$.

Now that we have a predictor of s_i with high probability, use the same set of problem instances to learn other s_j for $j \in [2, n]$. Performing a union bound over all i, we set $l = \Theta(t^2 \log(n/\varepsilon)/\varepsilon^2)$ for a failure probability of at most $\varepsilon/4$ in guessing correctly whether $s_i = s_1$ for all $i \in [2, n]$.

Finally, we can now prove our main theorem Theorem 3.1. Since we assumed that $d = O(1)$, the d-ary c-correlated predicate P must have bounded bias (depending on d), and we can assume $m = o(n^{c/2-2})$ and $\varepsilon = w(n^{-c/4})$. The recovery of the secret in total costs $O((n^2/\varepsilon^2)\log^3(n/\varepsilon)) \times m$ samples and runs in time polynomial of n, m and T (runtime of decision algorithm). The success probability is determined by the number of good secrets and whether the secret is ε-fairly balanced, as the above algorithm works if both these conditions are met. The probability that a randomly selected secret is not good is $1 - \varepsilon/2$ (Claim 3.13), the probability that a randomly selected secret is not fairly balanced is $o(\varepsilon)$ (Remark 3.7). By union bound, the probability that a randomly selected secret is both good and fairly balanced and the algorithm succeeds is $\Omega(\varepsilon)$. Finally, if the original distinguisher runs in polynomial time, then all of the above can be done in polynomial time as well. This proves Theorem 3.1.

4 Generalization

In this section, we will describe several interesting generalisations of our technique to a larger family of problems. Throughout, we say that a predicate has *bounded bias* if its bias is bounded away from 0 and 1 (see Definition 2.5).

4.1 Non-Constant Sparsity

Theorem 3.1 has the assumption that the sparsity d must be a constant. A natural question is whether the same result holds for larger non-constant sparsity, say $d = \log n$, and to what extent the techniques fail. The parts of the proof of Theorem 3.1 that requires constant sparsity are Claim 3.10 (that the output distribution of the function is close to Bernoulli), and the sample threshold lemma (Lemma 2.13, which gives algorithms for large enough m, letting us assume in our proof that $m = o(n^{c/2-2})$ and $\varepsilon = \omega(n^{-c/4})$). Unfortunately, the latter lemma might well not be true when $d = \omega(1)$.

Regardless, we can still generalise the relationship between m and ε for which our reduction applies. In particular, we generalise Claim 3.10 so that it allows non-constant locality. The requirement that we will need to add to allow our reduction to work for $d = \text{polylog}(n) = O(\log^r(n))$ for any constant $r > 0$ is:

$$(m/n^c) \cdot (\log^r(n)\log(1/\varepsilon))^{2c} = o(\varepsilon^2) \text{ and } \log^r(n)\log(1/\varepsilon) = o(\sqrt{n}) \tag{1}$$

Although these conditions may appear complicated, they are easily satisfied when $\varepsilon = w(n^{-c/4})$, $m = o(n^{c/2})$ (a stronger condition). The generalised theorem is as follows.

Theorem 4.1. *For $d = O(\log^r n)$ with constant $r > 0$ and $c \leq d$, consider a c-correlated d-ary predicate P with bounded bias. Suppose there is a polynomial-time algorithm that has advantage $\varepsilon \in [0,1]$ in solving the Decision Problem for (P, n, m) for some $m = poly(n)$ such that Eq. (1) is satisfied. Then there is a polynomial-time algorithm for the Search Problem for $(P, n, \ell m)$ that has success probability $\Omega(\varepsilon)$, for some $\ell = \Theta((n/\varepsilon)^2 \log^3(n/\varepsilon))$.*

Proof Sketch. The proof is the same as that of Theorem 3.1, except for the proof of the Terminal Hybrid Lemma 3.8, where we need to re-state Claim 3.10 to account for non-constant locality, as follows.

Claim 4.2. *Suppose s is ε-fairly balanced and $G \leftarrow G_{n,m,d}$, with $d = O(\log^r n)$, a c-correlated d-ary predicate P with bounded bias η and Eq. (1) is satisfied, then the statistical distance between $f_{G,P}(s)$ and $Bern(\eta)^m$ is at most $\varepsilon/8$*

Proof Sketch. Most of the proof remains the same as that of Claim 3.10; the first change that we have to make is that the bound we get on the bias of P when given non-uniform inputs is slightly different, requiring an additional $\log^{cr}(n)$ factor.

$$\mathop{\mathbf{E}}_{x_i \leftarrow D}[P(x_1, \ldots, x_d)] - \eta = O\left(\sum_{k=c}^{d} \left(\frac{ed\log(1/\varepsilon)}{k\sqrt{n}}\right)^k\right) = O\left(\frac{(\log^r(n)\log(1/\varepsilon))^c}{\sqrt{n}^c}\right)$$

Where the last equality is from the assumption that $\log^r(n)\log(1/\varepsilon) = o(\sqrt{n})$.

The second change is in the final part where we show the statistical distance between $f_{G,P}(s)$ and $Bern(\eta)^m$ is $o(\varepsilon)$, where we will use the assumption that $(m/n^c) \cdot (\log^r(n) \log(1/\varepsilon))^{2c} = o(\varepsilon^2)$ as follows.

$$O(\sqrt{m\alpha^{2c}}) = O\left(\sqrt{m \times \frac{(\log^r(n)\log(1/\varepsilon))^{2c}}{n^c}}\right) = o(\varepsilon)$$

□

□

4.2 Distinct values in the hyperedges

Another common definition for d-local functions is to have the requirement that in each edge of the hypergraph, every vertex is distinct – that is, for any output bit, the same bit is not given twice to the predicate as input when computing it. Our technique still works in this model, with a slightly larger number of hybrids t and with $d = O(\log^r n)$. There are two claims in the proof that are affected by this change, which are Claim 3.9 and Claim 3.10, both about the uniformity of the terminal hybrid.

For Claim 3.9, it could be that for the transformation $T_{a,b}$ that is applied, there are hyperedges that contain both a and b, which then cannot be replaced independently. The easiest modification to do here is to not perform any changes in this event. Formally, the transformation $T_{a,b}$ is modified as follows: for a hypergraph $G = (S_i)_{i\in[m]}$, each $S_i = (j_1, \ldots, j_d)$ is transformed to $S_i' = (j_1', \ldots, j_d')$ independently where

$$\Pr[j_k' = j_k] = 1 \qquad \text{if } j_k \notin \{a,b\} \text{ or } \{a,b\} \subseteq \{j_1,\ldots,j_d\}$$

$$\Pr[j_k' = a] = \frac{1}{2},\ \Pr[j_k' = b] = \frac{1}{2} \qquad \text{otherwise}$$

Then, for each edge, on expectation, there will be around $t \times (n^2 - d^2)/n^2$ transformations that have $\{a,b\} \not\subseteq \{j_1,\ldots,j_d\}$. Call these transformations *effective*. The expected number of effective transformations t' is asymptotically the same t when d is small (say $d = O(\log^r n)$).

$$\mathop{\mathbf{E}}_{a,b\leftarrow[n]}\left[t'\right] = t \times (n^2 - d^2)/n^2 = \Theta(t)$$

Since each choice of a, b is independent, using concentration bounds, using $10 \cdot t$ transformations will ensure that there will be at least t effective transformations for every edge with probability at least $1 - \exp(-\Omega(t))$, which is also $1 - o(\varepsilon)$.

The other issue is in Claim 3.10, where after an index in s is selected, it cannot be chosen again. Suppose the input contains more ones than zeros. In the worst case, the previously selected inputs are all zeros, which only increases the fraction of ones among the remaining inputs. Nevertheless, as long as d is small enough, the bias will still be bounded by $1/2 + O(\log(1/\varepsilon)/\sqrt{n})$ with a different constant factor. This is still considered as poly(ε)-fairly balanced secret, and would not be an issue asymptotically.

Define the Distinct Decision (respectively Search) Problem as the Decision (respectively Search) Problem, but with the hypergraph restricted to having distinct vertices in the hyperedges.

Theorem 4.3. *For $d = O(\log^r n)$ with constant $r > 0$ and $c \leq d$, consider a c-correlated d-ary predicate P with bounded bias. Suppose there is a polynomial-time algorithm that has advantage $\varepsilon \in [0,1]$ in solving the Distinct Decision Problem for (P, n, m) for some $m = poly(n)$ such that Eq. (1) is satisfied. Then there is a polynomial-time algorithm for the Distinct Search Problem for $(P, n, \ell m)$ that has success probability $\Omega(\varepsilon)$, for some $\ell = \Theta((n/\varepsilon)^2 \log^3(n/\varepsilon))$.*

Proof Sketch. The proof is almost the same as that of Theorem 3.1, except that first we replace $G_{n,m,d}$ to be the set of all hypergraphs having distinct vertices in the hyperedge. Next, we modify our transformation $T_{a,b}$ to not perform any swap when both a, b are in the hyperedge. It is easy to see that the uniform distribution over $G_{n,m,d}$ is still stable under this modified transformation. Then, we increase the number of transformations by a constant factor, say $10 \cdot t$, to still obtain Claim 3.9 with probability $1 - \exp(\Omega(t)) \geq 1 - o(\varepsilon)$.

Aside from the changes required to account for larger d (see Claim 4.2), we will also need to modify the proof of the bias to account for distinct values. (Refer to Section 5.2)

$$\mathop{\mathbf{E}}_{x_i \leftarrow D}[P(x_1, \ldots, x_d)] - \eta = -\frac{1}{2} \sum_{S \subseteq [d], |S| \geq c} \hat{P}'(S)\, \mathbf{E}\left[\prod_{i \in S} y_i\right]$$

We cannot expand the expectation as a product of expectations because they are not independent. However, we can still do conditional expansion of the expectation. $\mathbf{E}\left[\prod_{i \in S} y_i\right] = \mathbf{E}\left[y_j\, \mathbf{E}\left[\prod_{i \in S, i \neq j} y_i \mid y_j\right]\right]$.

Conditioned on the previous choices, the bias of the distribution does not change significantly, $\mathbf{E}[y_1 | y_2, \ldots, y_d] = \frac{1}{2} + O(\frac{\log(1/\varepsilon)}{\sqrt{n}})$. Therefore, we still achieve $\mathbf{E}_{x_i \leftarrow D}[P(x)] - \eta = O(\frac{\log^c(1/\varepsilon)}{\sqrt{n}^c})$. The rest of the proof follows. □

4.3 Noisy Predicate

Motivated by the LPN problem, we also apply our reduction to the case of the predicate P being noisy. Namely, the predicate P is defined as

$$P(x_1, \ldots, x_d) = R(x_1, \ldots, x_d) + e$$

for some deterministic function $R : \mathbb{F}_2^d \to \mathbb{F}_2$, where e is an independent Bernoulli random variable.

For a noisy predicate P, one affected part in our proof is that we cannot use the Fourier Transformation as is in the proof of Claim 3.10. Nevertheless, we still have the analogous claim on the deterministic predicate R, and it is easy to see that the error would only decrease the statistical distance between $f_{G,R}(s)$ and $Bern(\mathbf{E}[R])^m$.

$$\Delta(f_{G,P}(s), Bern(\mathbf{E}[P])^m) \leq \Delta(f_{G,R}(s), Bern(\mathbf{E}[R])^m)$$

This is due to the fact that the noise distribution added to both distributions is the same. However, since the bias of P might not be the same as the bias of R, we do need R to have bias bounded away from 0 and 1 for this to work. If $d = O(1)$, then any non-constant R will always have bounded bias.

The other part that is affected is the verification of the secret. Since noise is added, and our algorithm returns two secret candidates with $s_1 = 0$ or $s_1 = 1$ for each ε-fairly balanced Hamming weight, it could be difficult to identify which is the correct solution. One could just return the answer that is more likely to be correct in that case.

Theorem 4.4. *For $d = O(\log^r n)$ with constant $r > 0$ and $c \leq d$, consider a c-correlated d-ary noisy predicate P with bounded bias. Suppose there is a polynomial-time algorithm that has advantage $\varepsilon \in [0, 1]$ in solving the Decision Problem for (P, n, m) for some $m = poly(n)$ such that Eq. (1) is satisfied. Then, for some $\ell = \Theta((n/\varepsilon)^2 \log^3(n/\varepsilon))$, there is a polynomial-time algorithm for the Search Problem for $(P, n, \ell m)$ that, with probability $\Omega(\varepsilon)$, returns a set of secrets of size at most $2n$ that contains the solution.*

Proof. The proof is the same as Theorem 3.1, except at the proof of Claim 3.10 (refer to Section 5.2), we will first perform Fourier transformation on the deterministic portion of P, say R. Because the noise distribution added to both distributions is the same, we claim that noise will only result in a lower statistical distance. More formally, suppose $\eta = \mathbf{E}[R]$ and $\beta \in [0, 1/2]$ is the parameter for noise, we will need to show

$$\begin{aligned} &D_{KL}(Bern(\eta + \alpha) \oplus Bern(\beta) || Bern(\eta) \oplus Bern(\beta)) \\ &\leq D_{KL}(Bern(\eta + \alpha) || Bern(\eta)) \end{aligned}$$

The difference of probability of $Bern(\eta+\alpha) \oplus Bern(\beta) = 1$ and probability of $Bern(\eta) \oplus Bern(\beta) = 1$ is $\alpha(1-2\beta)$, which is smaller than α (the original difference). Therefore, the distribution is closer, which is why the KL divergence is smaller.

On the verification part, since we might not be able to perfectly identify which is the correct solution, we just return all candidate solutions in a set $\mathcal{S}$. For each Hamming weight, we will obtain 2 solutions, therefore the size of $\mathcal{S}$ is at most $2n$. □

5 Deferred Proofs

5.1 Random Graph

Claim 3.9 (Random Graph)**.** *Given any $G \in G_{n,m,d}$, if $a_j, b_j \leftarrow [n]$, $m = poly(n)$ and for some $t = O(n \log(nm/\varepsilon))$, then the statistical distance between $T_{a_t,b_t} \circ \ldots \circ T_{a_1,b_1} \circ G$ and the uniform distribution over $G_{n,m,d}$ is at most $\varepsilon/8$.*

Proof. To show the claim, it suffices to show that for a hyperedge $S_i = (j_1, \ldots, j_d)$, after the transformations, say $S'_i = (j'_1, \ldots, j'_d)$, each value j'_k is independently distributed like randomly sampled from $[n]$.

Define a distribution of values as a vector $D = (x_1, \ldots, x_n) \in [0, 1]^n$, such that $\sum x_i = 1$. The value x_k indicates the probability that sampling from D gives the value k. Define $D(i) = (x_1(i), \ldots, x_d(i))$ as the distribution of the value after applying i-th transformation to D.

We represent the distribution of a value in the hyperedge with D. Suppose we are working on vertex j_1 in a hyperedge S_i. Initially, the distribution $D(0)$ looks like $(0, 0, \ldots, 1, \ldots, 0)$. Basically, only $x_{j_1} = 1$ and the rest of the values are 0. Sampling from $D(0)$ gives j_1 with probability 1. Observe that when $T_{a,b}$ is applied to a hypergraph, its effect on $D(i)$ is that $x_a(i)$ and $x_b(i)$ are averaged, while the rest remains unchanged. Writing it down,

$$x_a(i + 1), x_b(i + 1) = (x_a(i) + x_b(i))/2$$

This is because $T_{a,b}$ only changes the value of a vertex in a hypergraph if it is either a or b. Otherwise, it remains unchanged. Our goal is to prove that the random averaging process converges to a uniform distribution; namely, when $t = \Omega(n \log(n/\varepsilon))$, $\sum_{k\in[n]}(|x_k(t) - 1/n|)$ becomes small. This then somewhat resembles the randomised gossip process, which we can then use a similar approach from [BGPS06] to show convergence. To prove that, we use the L_2 deviation function V defined as

$$V(i) = \sum_{k\in[n]} (x_k(i) - \frac{1}{n})^2$$

We claim that $0 \le V(i) \le 1$ by showing that $V(i)$ is a non-increasing function as i increases and $V(0) \le 1$. First on $V(0) \le 1$, writing it down

$$V(0) = \frac{(n-1)}{n^2} + (\frac{n-1}{n})^2 = \frac{n-1}{n} \le 1$$

Next on non-increasing, with a transformation of $T_{a,b}$, the initial contribution from x_a and x_b to the deviation is $(x_a - 1/n)^2 + (x_b - 1/n)^2$. Let $u = x_a - 1/n, v = x_b - 1/n$. After the transformation, the values are averaged and the contribution to $V(i+1)$ becomes

$$2(\frac{x_a + x_b}{2} - \frac{1}{n})^2 = 2(\frac{u+v}{2})^2 = \frac{(u+v)^2}{2}$$

The difference in the value of $V(i)$ and $V(i+1)$ is then

$$V(i) - V(i+1) = u^2 + v^2 - \frac{(u+v)^2}{2} = \frac{(u-v)^2}{2} = \frac{(x_a - x_b)^2}{2} \ge 0$$

Therefore, $V(i)$ is a non-increasing function. The next step is to show that $V(t)$ decays quickly to 0 when we apply random transformations. The expected difference after applying a transformation is

$$\begin{aligned}\mathop{\mathbf{E}}_{a,b\leftarrow[n]}\left[(x_a - x_b)^2/2\right] &= \frac{1}{2}\mathop{\mathbf{E}}_{a,b\leftarrow[n]}\left[x_a^2 - 2x_a x_b + x_b^2\right] \\ &= \frac{1}{n}\sum_{a\in[n]} x_a^2 - \frac{1}{n^2} = \frac{1}{n}\sum_{a\in[n]}(x_a - \frac{1}{n})^2 = \frac{V(i)}{n}\end{aligned}$$

Where the second equality is due to $\mathbf{E}\left[x_a^2\right] = \mathbf{E}\left[x_b^2\right]$ and $\mathbf{E}\left[x_a\right] = \mathbf{E}\left[x_b\right] = 1/n$. The third equality is due to the definition of variance $\frac{1}{n}\sum_{a\in[n]} x_a^2 - \frac{1}{n^2} = \mathbf{E}\left[x_a^2\right] - \mathbf{E}\left[x_a\right]^2 = \mathbf{E}\left[(x_a - \frac{1}{n})^2\right]$.

Therefore,

$$\mathbf{E}\left[V(i+1) \mid V(i) = v_i\right] = v_i - \mathbf{E}\left[(x_a - x_b)^2/2 \mid V(i) = v_i\right] = v_i - \frac{v_i}{n} = (1 - \frac{1}{n})v_i$$

So by law of total expectation,

$$\mathbf{E}\left[V(i+1)\right] = \int \left(\mathbf{E}\left[V(i+1) \mid V(i) = x\right] \Pr\left[V(i) = x\right]\right) dx = (1 - \frac{1}{n})\,\mathbf{E}\left[V(i)\right]$$

Starting with $V(0) \le 1$, this implies $\mathbf{E}\left[V(t)\right] \le (1 - 1/n)^t \le \exp(-t/n)$. By Markov inequality,

$$\Pr_{a_j,b_j\leftarrow[n]}[V(t) \ge (md/\varepsilon^2)\exp(-t/n)] \le \varepsilon^2/md \tag{2}$$

When $V(t) \leq (md/\varepsilon^2)\exp(-t/n)$, we use Cauchy–Schwarz inequality to bound the statistical distance between $V(t)$ and the uniform distribution

$$\sum_{k\in[n]}(|x_k(t)-1/n|) \leq \sqrt{nV(t)} \leq \frac{\sqrt{nmd}}{\varepsilon}\cdot\exp(-t/2n)$$

Each vertex in the hyperedge has its own distribution D. Note that their distributions for different vertices are not independent due to the choice of a_j, b_j. It could also not be identical due to different initial states. Nevertheless, each vertex is sampled independently from its own distribution D.

We say that a distribution of a vertex is bad if the (a_j, b_j)'s are chosen in such a way that $V(t)$ for that vertex is larger than $z = (md/\varepsilon^2)\exp(-t/n)$. The probability that the D is bad is at most ε^2/md (Eq. (2)). By union bound, across the md vertices in all the hyperedges, there is a distribution that is bad with probability at most ε^2. In the case of all distributions being good, the distance from uniform for one of the vertices is at most $\sqrt{nz}$. Therefore, the overall statistical distance between the product distribution and a randomly sampled hypergraph can be upperbounded by $\varepsilon^2 + md\cdot\sqrt{nz}$

As $m = poly(n)$ and $d = O(1)$, set $t = 8n\log(mdn/\varepsilon) = \Theta(n\log(nm/\varepsilon))$ to have the overall statistical distance at most $2\varepsilon^2$. We can then make the statistical distance smaller than $\varepsilon/8$ by adjusting the constant factor. □

5.2 Random Output

Claim 3.10 (Random Output). *Suppose s is ε-fairly balanced and $G \leftarrow G_{n,m,d}$, $m = o(n^{c/2-2})$, $\varepsilon = \omega(n^{-c/4})$, $d = O(1)$ and a c-correlated d-ary predicate P with bounded bias η, then the statistical distance between $f_{G,P}(s)$ and $Bern(\eta)^m$ is at most $\varepsilon/8$.*

Proof. We start the proof by introducing the Fourier transformation and explaining the connection between c-correlation and the Fourier coefficient.

Remark 5.1 (Fourier Transformation). *[O'D14, Chapter 1]* Let a d-ary predicate P, convert the working field from $\mathbb{F}_2$ to $\mathbb{R}$ by mapping $0, 1$ in $\mathbb{F}_2$ to $1, -1$ in $\mathbb{R}$. So we have $P : \{-1,1\}^d \to \{-1,1\}$. It can then be uniquely expressed as a multilinear polynomial

$$P(x_1,\ldots,x_d) = \sum_{S\subseteq[d]}\hat{P}(S)\prod_{i\in S}x_i$$

where the Fourier coefficient $\hat{P}(S) = \mathbf{E}\left[P(x_1,\ldots,x_d)\prod_{i\in S}x_i\right]$

When a predicate is c-correlated, by minimality of c, it also means the Fourier coefficient for all non-empty subsets of size less than c is zero. Now, we proceed to the main argument.

Since s is ε-fairly balanced, s has Hamming weight bounded by $[n/2-w, n/2+w]$ where $w = 2\sqrt{n}\log(1/\varepsilon)$. Without loss of generality, assume that s has at least as many ones as zeros. Let D be the distribution of randomly sampling an element in the binary string s. Then, D is a Bernoulli distribution that is only slightly biased.

$$\mathbf{E}[D] - 1/2 = \frac{2\log(1/\varepsilon)}{\sqrt{n}}$$

Let P', D' be the analog of P, D with output in $\{-1, 1\}$ and working field of $\mathbb{R}$. Assuming that this good event happens, from the Fourier expansion of c-correlated predicate P' (Remark 5.1), we have

$$\underset{x_i \leftarrow D}{\mathbf{E}}[P(x_1, \ldots, x_d)] - \eta = -\frac{1}{2} \sum_{S \subseteq [d], |S| \geq c} \hat{P}'(S) \prod_{i \in S} \mathbf{E}[y_i]$$
$$= O\left(\sum_{k=c}^{d} \binom{d}{k} \cdot \frac{\log^k(1/\varepsilon)}{\sqrt{n}^k} \right) = O\left(\sum_{k=c}^{d} (\frac{ed \log(1/\varepsilon)}{k\sqrt{n}})^k \right) = O(\frac{\log^c(1/\varepsilon)}{\sqrt{n}^c})$$

Where the last equality is due to $d = O(1)$. It is clear to see that if D is closer to $Bern(1/2)$, then the distribution of P when its inputs are sampled from D will also be closer to $Bern(\eta)$.

More precisely, let $\alpha = h_1 \log(1/\varepsilon)/\sqrt{n}$ for some constant $h_1 > 0$, and $D = Bern(1/2 + \alpha)$, then the output distribution of d-local function $f_{P,G}(s)$ (over random G) is $Bern(\eta + h_2\alpha^c)^m$ for some constant $h_2 > 0$ (the output bits are independent conditioned on the secret s). So as α goes smaller or c goes larger, it is closer to $Bern(\eta)^m$.

To quantitatively understand the distance between $Bern(\eta+\alpha^c)^m$ and $Bern(\eta)^m$. Use Pinsker's inequality that relates the statistical distance of two distributions with the KL divergence,

$$\Delta(Bern(\eta + \alpha^c)^m, Bern(\eta)^m) \leq \sqrt{D_{KL}(Bern(\eta + \alpha^c)^m || Bern(\eta)^m)}$$
$$= \sqrt{m \times D_{KL}(Bern(\eta + \alpha^c) || Bern(\eta))}$$

An upper bound on the KL divergence can be achieved by performing Taylor expansion on the KL divergence [CT06, Problem 11.2], which gives

$$D_{KL}(Bern(\eta + \alpha^c) || Bern(\eta)) = O\left(\alpha^{2c}/\eta(1-\eta)\right) = O(\alpha^{2c})$$

The final inequality is due to η being a bounded bias. Finally, the statistical distance is then

$$O(\sqrt{m\alpha^{2c}}) = O\left(\sqrt{m \times \frac{\log^{2c}(1/\varepsilon)}{n^c}} \right) = o(\frac{(c/4)^c \log^c(n)}{n^{c/4+1}}) = o(\varepsilon)$$

Where the second equality is due to $m = o(n^{c/2-2})$ and $\varepsilon = \omega(n^{-c/4})$. The last inequality is from $\varepsilon = \omega(n^{-c/4})$. Since the statistical distance is $o(\varepsilon)$, we can then make the statistical distance smaller than $\varepsilon/8$ by adjusting the constant factor. □

5.3 Good Secret

Claim 3.13. *Suppose* D *has advantage* ε*, then* $|\mathcal{G}_{\mathsf{D}}| \geq \varepsilon/2 \cdot (2^n)$.

Proof. Suppose η is the bias of P, let

$$p_0^s = \Pr_{\substack{G \leftarrow G_{n,m,d} \\ s' \leftarrow \mathbb{F}_2^n, wt(s) = wt(s')}} \left[\mathsf{D}(G, f_{G,P}(s')) = 1\right], \quad p_1^s = \Pr_{\substack{G \leftarrow G_{n,m,d} \\ b \leftarrow Bern(\eta)^m}} [\mathsf{D}(G, b) = 1]$$

Then,

$$\mathop{\mathbf{E}}_{s\leftarrow\mathbb{F}_2^n}[p_0^s] = \sum_{s\in\mathbb{F}_2^n}\frac{1}{2^n}p_0^s = \sum_{k=0}^{n}\sum_{\substack{s\in\mathbb{F}_2^n\\ wt(s)=k}}\frac{1}{2^n}\cdot\Pr_{G\leftarrow G_{n,m,d}}[\mathsf{D}(G,f_{G,P}(s))=1]$$

$$=\Pr_{\substack{G\leftarrow G_{n,m,d}\\ s'\leftarrow\mathbb{F}_2^n}}\left[\mathsf{D}(G,f_{G,P}(s'))=1\right]$$

So, $|\mathbf{E}[p_0^s]-\mathbf{E}[p_1^s]|\geq\varepsilon$. Using linearity of expectation and without loss of generality, $\mathbf{E}[p_0^s-p_1^s]\geq\varepsilon$. Let $q_s=1-(p_0^s-p_1^s)$, then $q_s\in[0,2]$ and $\mathbf{E}[q_s]\leq 1-\varepsilon$. Using Markov's inequality

$$\Pr_{s\leftarrow\mathbb{F}_2^n}[p_0^s-p_1^s<\varepsilon/2]=\Pr_{s\leftarrow\mathbb{F}_2^n}[q_s>1-\varepsilon/2]\leq\frac{1-\varepsilon}{1-\varepsilon/2}=1-\frac{\varepsilon/2}{1-\varepsilon/2}$$

Which implies

$$\Pr_{s\leftarrow\mathbb{F}_2^n}[p_0^s-p_1^s\geq\varepsilon/2]\geq\frac{\varepsilon/2}{1-\varepsilon/2}\geq\varepsilon/2$$

Therefore there should be at least $\varepsilon/2$ fraction of secret $s\in\mathbb{F}_2^n$ such that $s\in\mathcal{G}_{\mathsf{D}}$, which implies $|\mathcal{G}_{\mathsf{D}}|\geq(\varepsilon/2)\cdot 2^n$ □

5.4 Equal

Claim 3.16 (Equal). *If $s_1=s_i$, then $H\approx H_r^s$.*

Proof. If $s_1=s_i$, intuitively, H is still H_r^s because the secret bits are the same. The hybridisation is not really performing any randomisation as the output remains consistent. To prove this more formally, for technical reasons, we will need to define the inverse of the transformation $T_{a,b}$. However, since $T_{a,b}$ is a randomised function, whose inverse is not defined, derandomisation is then necessary.

Definition 5.2 (Derandomized Transformation). *Define a* deterministic transformation *: $T'_{a,b}:\mathbb{F}_2^{md}\times G_{n,m,d}\to G_{n,m,d}$. On input of a vector $v\in\mathbb{F}_2^{md}$ and a hypergraph G, each of the hyperedges $S_i=(j_1,\ldots,j_d)$ is transformed to $S'_i=(j'_1,\ldots,j'_d)$ as follows:*

$$j'_k=\begin{cases} j_k & \text{if } j_k\notin\{a,b\} \text{ or } v_{i\cdot d+k}=1\\ b & \text{if } j_k=a \text{ and } v_{i\cdot d+k}=0\\ a & \text{if } j_k=b \text{ and } v_{i\cdot d+k}=0\end{cases}$$

The randomised transformations from our definitions of hybrids can be formed by sampling a vector v uniformly from $\mathbb{F}_2^{m\times d}$ and applying the above deterministic transformation with that vector. It is also easy to see that given the vector v, the transformation is reversible. Therefore, we define the inverse of $(T'_{a,b})^{-1}:\mathbb{F}_2^{m\times d}\times G_{n,m,d}\to G_{n,m,d}$ to reverse the process of T'. Now, we can show an alternative view on the hybrids

Claim 5.3 (Alternative View on Hybrid). *$H_r^s\approx(K,f_{K',P}(\pi(s)))$ where $K\leftarrow G_{n,m,d}$, π is a random permutation on $[n]$ and $K'=(T'_{a_1,b_1})^{-1}(v_1,\cdot)\circ\ldots\circ(T'_{a_r,b_r})^{-1}(v_r,K)$ where $v_i\leftarrow\mathbb{F}_2^{md}$ and $a_j,b_j\leftarrow[n]$*

Proof. We previously defined hybrid as $H_r^s = (T_{a_r,b_r} \circ \ldots \circ T_{a_1,b_1}(\pi(G)), f_{G,P}(s))$ with $G \leftarrow G_{n,m,d}$ (Definition 3.5). First, rewrite it with the derandomised transformation

$$H_r^s = (T'_{a_r,b_r}(v_r, \cdot) \circ \ldots \circ T_{a_1,b_1}(v_1, (\pi(G)), f_{G,P}(s))$$

where $v_i \leftarrow \mathbb{F}_2^{md}$. Also, for any permutation π on $[n]$, $f_{G,P}(s) = f_{\pi(G),P}(\pi(s))$. Therefore, if we let $K = T'_{a_r,b_r}(v_r, \cdot) \circ \ldots \circ T_{a_1,b_1}(v_1, (\pi(G))$, then

$$K' = \pi(G) = (T'_{a_1,b_1})^{-1}(v_1, \cdot) \circ \ldots \circ (T'_{a_r,b_r})^{-1}(v_r, K)$$

So $H_r^s = (K, f_{G,P}(s)) = (K, f_{\pi(G),P}(\pi(s))) = (K, f_{K',P}(\pi(s)))$. Since $G \leftarrow G_{n,m,d}$ and the uniform distribution is stable under the transformations, so $K \approx G_{n,m,d}$. □

In our case of H, we additionally apply $T_{\pi(1),\pi(i)}$. Using the alternative view on hybrid, $H_r^s = (K, f_{K',P}(\pi(s)))$, we have:

$$H = (K, f_{K'',P}(\pi(s)))$$

where

$$K'' = (T'_{a_1,b_1})^{-1}(v_1, \cdot) \circ \ldots \circ (T'_{a_r,b_r})^{-1}(v_r, \cdot) \circ (T'_{\pi(1),\pi(i)})^{-1}(v, K)$$

where $v_i \leftarrow \mathbb{F}_2^{md}$. But since $s_1 = s_i$, so $\pi(s)_{\pi(1)} = \pi(s)_{\pi(i)}$. This implies that applying $(T'_{a_r,b_r})^{-1}$ does not change the value of the d-local function. i.e.

$$f_{K'',P}(\pi(s)) = f_{K',P}(\pi(s))$$

Which proves that $H \approx H_r^s$. □

5.5 Not Equal

Claim 3.17 (Not Equal)**.** *If $s_1 \neq s_i$, $\Pr[\mathsf{D}(H_{r+1}^s) = 1] \geq \Pr[\mathsf{D}(H) = 1]$.*

Proof. Observe that H_{r+1}^s is a mixture distribution of H_r^s and H. In the proof of Section 5.4, we have already shown that, suppose $\pi(s)_a = \pi(s)_b$ and $T_{a,b}$ is the last transformation applied to the hypergraph for H_{r+1}^s, then no randomisation is applied and it is distributed as H_r^s. Let $\pi(s) = s'$ and A be the event that $s'_a = s'_b$, and B be the event that $s'_a \neq s'_b$. We can say:

$$\Pr[H_{r+1}^s = h] = \Pr_{a,b}[A] \Pr[H_r^s = h] + \Pr_{a,b}[B] \Pr[H = h] \tag{3}$$

This is because in the last transformation that we apply to H_r^s to get H, our assumption on $s_1 \neq s_i$ ensures that $T_{\pi(1),\pi(i)}$ always has $s'_{\pi(1)} \neq s'_{\pi(i)}$. Furthermore, the randomness of π ensures that $(\pi(1), \pi(i))$ is distributed just as sampling (a, b) condition on $s'_a \neq s'_b$. Using Eq. (3),

$$\begin{aligned}\Pr_r[\mathsf{D}(H_{r+1}^s) = 1] &= \Pr_r[\mathsf{D}(H_r^s) = 1] \Pr[A] + \Pr_r[\mathsf{D}(H) = 1] \Pr[B] \\ &= \Pr_r[\mathsf{D}(H_r^s) = 1](1 - \Pr[B]) + \Pr_r[\mathsf{D}(H) = 1] \Pr[B]\end{aligned}$$

Substitute in $\Pr_r[\mathsf{D}(H_r^s) = 1] \geq \Pr_r[\mathsf{D}(H_{r+1}^s) = 1] + \varepsilon/4t$ (from Corollary 3.15),

$$\begin{aligned}\Pr_r[\mathsf{D}(H_{r+1}^s) = 1] &\geq (\Pr_r[\mathsf{D}(H_{r+1}^s) = 1] + \varepsilon/4t)(1 - \Pr[B]) \\ &\quad + \Pr_r[\mathsf{D}(H) = 1] \Pr[B] \\ \implies (\Pr[B] - 1)\varepsilon/4t &\geq (\Pr[B])(\Pr_r[\mathsf{D}(H) = 1] - \Pr_r[\mathsf{D}(H_{r+1}^s) = 1])\end{aligned}$$

Since $0 \geq (\Pr[B] - 1)\varepsilon/4t$, so $\Pr[\mathsf{D}(H_{r+1}^s) = 1] \geq \Pr[\mathsf{D}(H) = 1]$. □

Acknowledgements

Both authors are supported by the National Research Foundation, Singapore, under its NRF Fellowship programme, award no. NRF-NRFF14-2022-0010. AI tools were used as typing assistants for grammar and basic editing. We thank the EUROCRYPT 2026 reviewers for their helpful suggestions.

References

[ABCM25] Benny Applebaum, Dung Bui, Geoffroy Couteau, and Nikolas Melissaris. Structured-Seed Local Pseudorandom Generators and Their Applications. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2025)*, volume 353, pages 63:1–63:26. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2025.

[ABR12] Benny Applebaum, Andrej Bogdanov, and Alon Rosen. A dichotomy for local small-bias generators. In *Proceedings of the 9th International Conference on Theory of Cryptography*, pages 600–617. Springer-Verlag, 2012.

[ABW10] Benny Applebaum, Boaz Barak, and Avi Wigderson. Public-key cryptography from different assumptions. In *Proceedings of the forty-second ACM symposium on Theory of computing*, pages 171–180, 2010.

[ADI$^+$17] Benny Applebaum, Ivan Damgård, Yuval Ishai, Michael Nielsen, and Lior Zichron. Secure arithmetic computation with constant computational overhead. In *CRYPTO*, pages 223–254. Springer, 2017.

[AHI05] Michael Alekhnovich, Edward A. Hirsch, and Dmitry Itsykson. Exponential lower bounds for the running time of dpll algorithms on satisfiable formulas. *J. Autom. Reason.*, 35(1–3):51–72, 2005.

[AIK06] Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. Cryptography in \$NCˆ0\$. *SIAM Journal on Computing*, 36(4):845–888, 2006.

[AIK08] Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. On pseudorandom generators with linear stretch in nc 0. *Computational Complexity*, 17:38–69, 2008.

[AK19] Benny Applebaum and Eliran Kachlon. Sampling Graphs without Forbidden Subgraphs and Unbalanced Expanders with Negligible Error. In *2019 IEEE 60th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 171–179, 2019.

[AL16] Benny Applebaum and Shachar Lovett. Algebraic attacks against random local functions and their countermeasures. In *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing*, pages 1087–1100, 2016.

[Ale03] Michael Alekhnovich. More on average case vs approximation complexity. In *44th Annual IEEE Symposium on Foundations of Computer Science, 2003. Proceedings.*, pages 298–307. IEEE, 2003.

[AM13] Benny Applebaum and Yoni Moses. Locally Computable UOWHF with Linear Shrinkage. In *Advances in Cryptology – EUROCRYPT 2013*, pages 486–502. Springer, 2013.

[App12] Benny Applebaum. Pseudorandom generators with long stretch and low locality from random local one-way functions. In *Proceedings of the forty-fourth annual ACM symposium on Theory of computing*, pages 805–816, 2012.

[App16] Benny Applebaum. Cryptographic hardness of random local functions: Survey. *Computational complexity*, 25:667–722, 2016.

[BBTV24] Kiril Bangachev, Guy Bresler, Stefan Tiegel, and Vinod Vaikuntanathan. Near-optimal time-sparsity trade-offs for solving noisy linear equations. *arXiv preprint arXiv:2411.12512*, 2024.

[BCG$^+$17] Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, and Michele Orrù. Homomorphic Secret Sharing: Optimizations and Applications. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 2105–2122. Association for Computing Machinery, 2017.

[BCM23] Elette Boyle, Geoffroy Couteau, and Pierre Meyer. Sublinear-Communication Secure Multiparty Computation Does Not Require FHE. In *Advances in Cryptology – EUROCRYPT 2023: 42nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Lyon, France, April 23–27, 2023, Proceedings, Part II*, pages 159–189. Springer-Verlag, 2023.

[BCM$^+$24] Dung Bui, Geoffroy Couteau, Pierre Meyer, Alain Passelègue, and Mahshid Riahinia. Fast Public-Key Silent OT and More from Constrained Naor-Reingold. In *Advances in Cryptology – EUROCRYPT 2024: 43rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zurich, Switzerland, May 26–30, 2024, Proceedings, Part VI*, pages 88–118. Springer-Verlag, 2024.

[BCM$^+$25] Lennart Braun, Geoffroy Couteau, Kelsey Melissaris, Mahshid Riahinia, and Elahe Sadeghi. Fast pseudorandom correlation functions from sparse LPN. *IACR Cryptol. ePrint Arch.*, page 1644, 2025.

[BEG$^+$22] Boaz Barak, Benjamin Edelman, Surbhi Goel, Sham Kakade, Eran Malach, and Cyril Zhang. Hidden progress in deep learning: Sgd learns parities near the computational limit. *Advances in Neural Information Processing Systems*, 35:21750–21764, 2022.

[BGPS06] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah. Randomized gossip algorithms. *IEEE Transactions on Information Theory*, 52(6):2508–2530, 2006.

[BQ09] Andrej Bogdanov and Youming Qiao. On the Security of Goldreich's One-Way Function. In *Proceedings of the 12th International Workshop and 13th International Workshop on Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 392–405. Springer-Verlag, 2009.

[BR11] Andrej Bogdanov and Alon Rosen. Input locality and hardness amplification. In *Proceedings of the 8th Conference on Theory of Cryptography*, pages 1–18. Springer-Verlag, 2011.

[BRT25] Andrej Bogdanov, Alon Rosen, and Kel Zin Tan. Sample efficient search to decision for klin. In *Advances in Cryptology – CRYPTO 2025: 45th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 17–21, 2025, Proceedings, Part I*, page 203–220. Springer-Verlag, 2025.

[BSV19] Andrej Bogdanov, Manuel Sabin, and Prashant Nalini Vasudevan. Xor codes and sparse learning parity with noise. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 986–1004. SIAM, 2019.

[CDM$^+$18] Geoffroy Couteau, Aurélien Dupin, Pierrick Méaux, Mélissa Rossi, and Yann Rotella. On the Concrete Security of Goldreich's Pseudorandom Generator. In *Advances in Cryptology – ASIACRYPT 2018: 24th International Conference on the Theory and Application of Cryptology and Information Security, Brisbane, QLD, Australia, December 2–6, 2018, Proceedings, Part II*, pages 96–124. Springer-Verlag, 2018.

[CEMT09] James Cook, Omid Etesami, Rachel Miller, and Luca Trevisan. Goldreich's One-Way Function Candidate and Myopic Backtracking Algorithms. In *Proceedings of the 6th Theory of Cryptography Conference on Theory of Cryptography*, pages 521–538. Springer-Verlag, 2009.

[CGHKV24] Henry Corrigan-Gibbs, Alexandra Henzinger, Yael Kalai, and Vinod Vaikuntanathan. Somewhat homomorphic encryption from linear homomorphism and sparse lpn. *Cryptology ePrint Archive*, 2024.

[CM01] Mary Cryan and Peter Bro Miltersen. On Pseudorandom Generators in NC0. In *Mathematical Foundations of Computer Science 2001*, pages 272–284. Springer, 2001.

[COST19] Igor Carboni Oliveira, Rahul Santhanam, and Roei Tell. Expander-Based Cryptography Meets Natural Proofs. In *10th Innovations in Theoretical Computer Science Conference (ITCS 2019)*, volume 124, pages 18:1–18:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019.

[CSZ24] Xue Chen, Wenxuan Shu, and Zhaienhe Zhou. Algorithms for sparse lpn and lspn against low-noise. *arXiv preprint arXiv:2407.19215*, 2024.

[CT06] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory (Wiley Series in Telecommunications and Signal Processing)*. Wiley-Interscience, 2006.

[DJ24] Quang Dao and Aayush Jain. Lossy Cryptography from Code-Based Assumptions. In *Advances in Cryptology – CRYPTO 2024: 44th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18–22, 2024, Proceedings, Part III*, pages 34–75. Springer-Verlag, 2024.

[DMR23] Aurélien Dupin, Pierrick Méaux, and Mélissa Rossi. On the algebraic immunity—resiliency trade-off, implications for Goldreich's pseudorandom generator. *Des. Codes Cryptography*, 91(9):3035–3079, 2023.

[DV21] Amit Daniely and Gal Vardi. From local pseudorandom generators to hardness of learning. In *Conference on Learning Theory, COLT 2021, 15-19 August 2021, Boulder, Colorado, USA*, volume 134, pages 1358–1394. PMLR, 2021.

[Fei02] Uriel Feige. Relations between average case complexity and approximation complexity. In *Proceedings of the Thiry-Fourth Annual ACM Symposium on Theory of Computing*, pages 534–543. Association for Computing Machinery, 2002.

[FGKP06] Vitaly Feldman, Parikshit Gopalan, Subhash Khot, and Ashok Kumar Ponnuswami. New results for learning noisy parities and halfspaces. In *2006 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06)*, pages 563–574. IEEE, 2006.

[GHKM23] Venkatesan Guruswami, Jun-Ting Hsieh, Pravesh K Kothari, and Peter Manohar. Efficient algorithms for semirandom planted csps at the refutation threshold. In *2023 IEEE 64th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 307–327. IEEE, 2023.

[Gol11] Oded Goldreich. Candidate one-way functions based on expander graphs. *Studies in Complexity and Cryptography. Miscellanea on the Interplay between Randomness and Computation: In Collaboration with Lidor Avigad, Mihir Bellare, Zvika Brakerski, Shafi Goldwasser, Shai Halevi, Tali Kaufman, Leonid Levin, Noam Nisan, Dana Ron, Madhu Sudan, Luca Trevisan, Salil Vadhan, Avi Wigderson, David Zuckerman*, pages 76–87, 2011.

[Its10] Dmitry Itsykson. Lower bound on average-case complexity of inversion of goldreich's function by drunken backtracking algorithms. In *Proceedings of the 5th International Conference on Computer Science: Theory and Applications*, page 204–215. Springer-Verlag, 2010.

[JLS21] Aayush Jain, Huijia Lin, and Amit Sahai. Indistinguishability obfuscation from well-founded assumptions. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pages 60–73. Association for Computing Machinery, 2021.

[JLS22] Aayush Jain, Huijia Lin, and Amit Sahai. Indistinguishability Obfuscation from LPN over Fp, DLIN, and PRGs in NC0. In *Advances in Cryptology – EUROCRYPT 2022: 41st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Trondheim, Norway, May 30 – June 3, 2022, Proceedings, Part I*, pages 670–699. Springer-Verlag, 2022.

[LV17] Alex Lombardi and Vinod Vaikuntanathan. Limits on the Locality of Pseudorandom Generators and Applications to Indistinguishability Obfuscation. In *Theory of Cryptography: 15th International Conference, TCC 2017, Baltimore, MD, USA, November 12-15, 2017, Proceedings, Part I*, pages 119–137. Springer-Verlag, 2017.

[MST03] Elchanan Mossel, Amir Shpilka, and Luca Trevisan. On e-Biased Generators in NC0. In *Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science*, page 136. IEEE Computer Society, 2003.

[O'D14] Ryan O'Donnell. *Analysis of boolean functions*. Cambridge University Press, 2014.

[OW14] Ryan O'Donnell and David Witmer. Goldreich's PRG: Evidence for Near-Optimal Polynomial Stretch. In *Proceedings of the 2014 IEEE 29th Conference on Computational Complexity*, pages 1–12. IEEE Computer Society, 2014.

[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*, pages 121–131, 2017.

[RVV24] Seyoon Ragavan, Neekon Vafa, and Vinod Vaikuntanathan. Indistinguishability Obfuscation from Bilinear Maps and LPN Variants. In *Theory of Cryptography: 22nd International Conference, TCC 2024, Milan, Italy, December 2–6, 2024, Proceedings, Part IV*, pages 3–36. Springer-Verlag, 2024.

[Üna23] Akin Ünal. Worst-case subexponential attacks on prgs of constant degree or constant locality. In *Advances in Cryptology - EUROCRYPT 2023 - 42nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Lyon, France, April 23-27, 2023, Proceedings, Part I*, volume 14004, pages 25–54. Springer, 2023.

[YGJL22] Jing Yang, Qian Guo, Thomas Johansson, and Michael Lentmaier. Revisiting the Concrete Security of Goldreich's Pseudorandom Generator. *IEEE Trans. Inf. Theor.*, 68(2):1329–1354, 2022.

[Zic17] Lior Zichron. Locally computable arithmetic pseudorandom generators. Master's thesis, School of Electrical Engineering, Tel Aviv University, 2017.