Pseudorandom generators without the XOR Lemma[*]

[Extended Abstract]

Madhu Sudan[†] Luca Trevisan[‡] Salil Vadhan[§]

Abstract

Impagliazzo and Wigderson [IW97] have recently shown that if there exists a decision problem solvable in time $2^{O(n)}$ and having circuit complexity $2^{\Omega(n)}$ (for all but finitely many n) then $\mathsf{P} = \mathsf{BPP}$. This result is a culmination of a series of works showing connections between the existence of hard predicates and the existence of good pseudorandom generators.

The construction of Impagliazzo and Wigderson goes through three phases of "hardness amplification" (a multivariate polynomial encoding, a first derandomized XOR Lemma, and a second derandomized XOR Lemma) that are composed with the Nisan–Wigderson [NW94] generator. In this paper we present two different approaches to proving the main result of Impagliazzo and Wigderson. In developing each approach, we introduce new techniques and prove new results that could be useful in future improvements and/or applications of hardness-randomness trade-offs.

Our first result is that when (a modified version of) the Nisan-Wigderson generator construction is applied with a "mildly" hard predicate, the result is a generator that produces a distribution indistinguishable from having large min-entropy. An extractor can then be used to produce a distribution computationally indistinguishable from uniform. This is the first construction of a pseudorandom generator that works with a mildly hard predicate without doing hardness amplification.

We then show that in the Impagliazzo–Wigderson construction only the first hardness-amplification phase (encoding with multivariate polynomial) is necessary, since it already gives the required average-case hardness. We prove this result by (i) establishing a connection between the hardness-amplification problem and a list-decoding problem for error-correcting codes based on multivariate polynomials; and (ii) presenting a list-decoding algorithm that improves and simplifies a previous one by Arora and Sudan [AS97].

[*] The full version of this paper appears as [STV98].

[†] Laboratory for Computer Science, 545 Technology Square, MIT, Cambridge, MA 02141. E-mail: madhu@theory.lcs.mit.edu.

[‡] Department of Computer Science, Columbia University, 500W 120th St., New York, NY 10027. Email: luca@cs.columbia.edu. Work done at MIT.

[§] Laboratory for Computer Science, 545 Technology Square, MIT, Cambridge, MA 02141. E-mail: salil@theory.lcs.mit.edu. URL: http://theory.lcs.mit.edu/~salil. Supported by a DOD/NDSEG graduate fellowship and partially by DARPA grant DABT63-96-C-0018.

1 Introduction

This paper continues the exploration of hardness versus randomness trade-offs, that is, results showing that randomized algorithms can be efficiently simulated deterministically if certain complexity-theoretic assumptions are true. We present two new approaches to proving the recent result of Impagliazzo and Wigderson [IW97] that, if there is a decision problem computable in time $2^{O(n)}$ and having circuit complexity $2^{\Omega(n)}$ for all but finitely many n, then $\mathsf{P} = \mathsf{BPP}$. Impagliazzo and Wigderson prove their result by presenting a "randomness-efficient amplification of hardness" based on a derandomized version of Yao's XOR Lemma. The hardness-amplification procedure is then composed with the Nisan–Wigderson (NW) generator [NW94] and this gives the result. The hardness amplification goes through three steps: an encoding using multivariate polynomials (from [BFNW93]), a first derandomized XOR Lemma (from [Imp95]) and a second derandomized XOR Lemma (which is the technical contribution of [IW97]).

In our first result, we show how to construct a "pseudoentropy generator" starting from a predicate with "mild" hardness. Roughly speaking, a *pseudoentropy generator* takes a short random seed as input and outputs a distribution that is indistinguishable from having high min-entropy. Combining our pseudoentropy generator with an *extractor*, we obtain a pseudorandom generator. Interestingly, our pseudoentropy generator is (a modification of) the NW generator itself. Along the way we prove that, when built out of a mildly hard predicate, the NW generator outputs a distribution that is indistinguishable from having high Shannon entropy, a result that has not been observed before. The notion of a pseudoentropy generator, and the idea that a pseudoentropy generator can be converted into a pseudorandom generator using an extractor, are due to Håstad et al. [HILL98].[1] Our construction is the first construction of a pseudorandom generator that works using a mildly hard predicate and without hardness amplification.

We then revisit the hardness amplification problem, as considered in [BFNW93, Imp95, IW97], and we show that the first step alone (encoding with multivariate polynomials) is sufficient to amplify hardness to the desired level, so that the derandomized XOR Lemmas are not necessary in this context. Our proof is based on a list-decoding algorithm for multivariate polynomial codes and exploits a connection between the list-decoding and the hardness-amplification problems. The list-decoding algorithm described in this paper is quantitatively better than a previous one by Arora and Sudan [AS97], and has a simpler analysis.

[1] To be accurate, the term *extractor* comes fron [NZ96] and postdates the paper of Håstad et al. [HILL98].

An overview of previous results. The works of Blum and Micali [BM84] and Yao [Yao82] introduce the notion of a cryptographically strong pseudorandom generator (csPRG) and show how to construct pseudorandom generators based on the existence of one-way permutations. A csPRG is a polynomial-time algorithm that on input a randomly selected string of length n^ϵ produces an output of length n that is computationally indistinguishable from uniform by any adversary of $\mathrm{poly}(n)$ size, where ϵ is an arbitrarily small constant. Yao also observes that a given polynomial-time randomized algorithm can be simulated deterministically using a csPRG in time $2^{n^\epsilon} \cdot \mathrm{poly}(n)$ by trying all the seeds and taking the majority answer.

In a seminal work, Nisan and Wigderson [NW94] explore the use of a weaker type of pseudorandom generator (PRG) in order to derandomize randomized algorithm. They observe that, for the purpose of derandomization, one can consider generators computable in time $\mathrm{poly}(2^t)$ (instead of $\mathrm{poly}(t)$) where t is the length of the seed, since the derandomization process cycles through all the seeds, and this induces an overhead factor 2^t anyway. They also observe that one can restrict to generators that are good against adversaries whose running time is bounded by a fixed polynomial, instead of every polynomial. They then show how to construct a pseudorandom generator meeting this relaxed definition under weaker assumptions than those used to build cryptographically strong pseudorandom generators. Furthermore, they show that, under a sufficiently strong assumption, one can build a PRG that uses seeds of logarithmic length (which would be impossible for a csPRG). Such a generator can be used to simulate randomized algorithms in polynomial time, and its existence implies $\mathsf{P} = \mathsf{BPP}$. The condition under which Nisan and Wigderson prove the existence of a PRG with seeds of logarithmic length is the existence of a decision problem (i.e., a predicate $P\colon \{0,1\}^n \to \{0,1\}$) solvable in time $2^{O(n)}$ such that for some positive constant ϵ no circuit of size $2^{\epsilon n}$ can solve the problem on more than a fraction $1/2 + 2^{-\epsilon n}$ of the inputs.[2] This is a very strong hardness requirement, and it is of interest to obtain similar conclusions under weaker assumptions.

An example of a weaker assumption is the existence of a *mildly hard* predicate. We say that a predicate is mildly hard if for some fixed $\epsilon > 0$ no circuit of size $2^{\epsilon n}$ can decide the predicate on more than a fraction $1 - 1/\mathrm{poly}(n)$ of the inputs. Nisan and Wigderson prove that mild hardness suffices to derive a pseudorandom generator with seed of $O(\log^2 n)$ length, which in turn implies a quasi-polynomial deterministic simulation of BPP. This result is proved by using Yao's XOR Lemma [Yao82] (see, e.g., [GNW95] for a proof) to convert a mildly hard predicate over n inputs into one which has input size n^2 and is hard to compute on a fraction $1/2 + 2^{-\Omega(n)}$ of the inputs. A series of subsequent papers attacks the problem of obtaining stronger pseudorandom generators starting from weaker and weaker assumptions. Babai et al. [BFNW93] show that a predicate of *worst-case* circuit complexity $2^{\Omega(n)}$ can be converted into a mildly hard one.[3] Impagliazzo [Imp95] proves a derandomized XOR Lemma which implies that a mildly hard predicate can be converted into one that cannot be predicted on more than some *constant* fraction of the inputs by circuits of size $2^{\epsilon n}$. Impagliazzo and Wigderson [IW97] prove that a predicate with the latter hardness condition can be transformed into one that meets the hardness requirement of [NW94]. The result of [IW97] relies on a different derandomized version of the XOR Lemma than [Imp95]. Thus, the general structure of the original construction of Nisan and Wigderson [NW94] has been preserved in most subsequent works, progress being achieved by improving the single components. In particular, the use of an XOR Lemma in [NW94] continues, albeit in increasingly sophisticated forms, in [Imp95, IW97]. Likewise, the NW generator and its original analysis have always been used in conditional derandomization results since.[4] Future progress in the area will probably require a departure from this observance of the NW methodology, or at least a certain amount of revisitation of its main parts.

In this paper, we give two new ways to build pseudorandom generators with seeds of logarithmic length. Both approaches bypass the need for the XOR Lemma, and instead use tools (such as list decoding, extractors, and pseudoentropy generators) that did not appear in the sequence of works from [NW94] to [IW97]. For a diagram illustrating the steps leading up to the results of [IW97] and how our techniques depart from that framework, see Figure 1. Both of our approaches are described in more detail below.

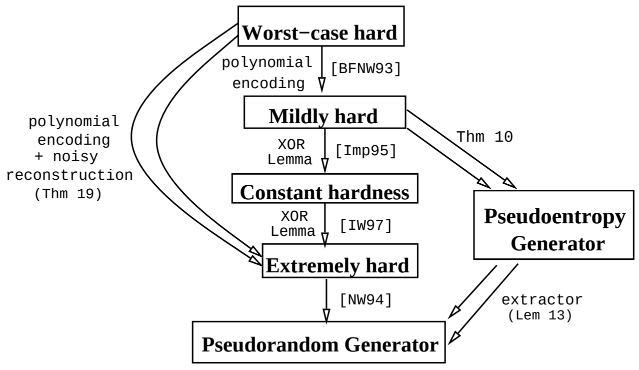

Figure 1: A comparison of our approach with previous ones. Double arrows indicate our results.

A Pseudoentropy Generator. Nisan and Wigderson show that when their generator is constructed using a very hard-on-average predicate, then the output of the generator is indistinguishable from the uniform distribution. It is a natural question to ask what happens if there are stronger or weaker conditions on the predicate. In this paper we consider the question of what happens if the predicate is only mildly hard. Specifically we are interested in whether exponential average-case hardness is really necessary for direct pseudorandom generation. In this paper we first show that, when a mildly hard predicate is used in the NW generator, then there exists a distribution having high Shannon entropy that is indistinguishable from the output of the generator. Our main result is then that, for a mildly hard predicate, a modified version of the NW generator has an output indistinguishable from a distribution with high min-entropy. Such a generator is essentially a "pseudoentropy generator" in the sense of Håstad et al. [HILL98]. The intuition behind our proof is that if a predicate is hard to compute on more than a fraction $1 - \delta$ of the inputs then there should be some subset of the inputs of density δ on which the predicate is very hard — this intuition is made precise by a result of Impagliazzo [Imp95]. Due to the high hardness, the evaluation of the predicate in a random point of this set will be indistinguishable from a random bit. The NW generator constructed with a predicate P works by transforming an input seed s into a sequence of points $x_1, \ldots, x_m$ from the domain of P; the output of the generator is then $P(x_1)P(x_2)\cdots P(x_m)$. For a random seed, each of the points x_i is uniformly distributed, and so we expect to typically generate δm points from the hard set, so that the output of the generator looks like having δm bits of

[2]This has to be true for all but finitely many input lengths n.

[3]In fact the result of [BFNW93] was somewhat weaker, but it is easily extendable to yield this result.

[4]The techniques of Andreev et al. [ACR97] are a rare exception, but they yield weaker result than the ones of [IW97].

randomness, that is, it is indistinguishable from some other distribution having (Shannon) entropy δm. The generation of the points $x_1 \cdots x_m$ can be modified so that the number of points landing in the hard set is sharply concentrated around its expected value δm. The output of the modified generator is then indistinguishable from having high min-entropy. When our generator is composed with a sufficiently good extractor[5] (such as the one in [Tre98]) then the result is a pseudorandom generator. This is the first construction of a pseudorandom generator based on mild average-case hardness that does not rely on hardness amplification. It is also the first application of the notion of a pseudoentropy generator to the construction of PRG in the Nisan–Wigderson sense.

Remark 1 While in this paper we analyze for the first time the Nisan-Wigderson generator under a *weaker* assumption than the one originally considered in [NW94], there has also been some work exploring the effect of *stronger* assumptions on the predicate. Impagliazzo and Wigderson [IW98] show that if the predicate has certain additional properties (such as "downward self-reducibility") then one needs only a *uniform* hardness assumption on the predicate (rather circuit-complexity assumption). Arvind and Köbler [AK97] and Klivans and van Melkebeek [KvM98] show that if the predicate is hard on average for *nondeterministic* circuits, then the output of the generator is indistinguishable from uniform for nondeterministic adversaries. Therefore it is possible to derandomize classes involving randomness and nondeterminism, such as AM. Trevisan [Tre98] shows that if the predicate is chosen randomly from a distribution having certain properties, then the output is *statistically* close to uniform. This yields the construction of extractors that we use in our generator.

The Connection with Polynomial Reconstruction. Our second result deals with the polynomial reconstruction problem and its connection to amplification of hardness. In the polynomial reconstruction problem we want to design an efficient randomized algorithm that, given oracle access to a function f that has δ-agreement[6] an unknown low-degree polynomial $p(\cdot)$, computes $p(\cdot)$ on every input with high probability over its internal coin tosses. This problem has been studied for its applications to program checking, average case hardness results for the permanent, and random self-reducibility of complete problems in high complexity classes [BF90, Lip89, GLR$^+$91, FF93, GS92, FL96, CPS99].

The applicability of polynomial reconstruction to hardness versus randomness results was demonstrated by Babai et al. [BFNW93]. They show that the existence of a polynomial reconstruction procedure implies that one can convert a worst-case hard predicate into one which is mildly average-case hard by encoding it as a polynomial. As in [NW94, Imp95, IW97] one can further amplify the hardness using XOR Lemmas. It is intuitively clear that a stronger reconstruction algorithm would imply a stronger hardness amplification result and it could be used in place of the XOR Lemma. Unfortunately, some calculations show that a polynomial reconstruction algorithm would have to work with $\delta = o(1)$ in order to be a viable surrogate of an XOR Lemma, but if we have access to a function f with δ-agreement to some polynomial $p(\cdot)$, where $\delta < 1/2$, then $p(\cdot)$ is not uniquely determined and the polynomial reconstruction problem is ill-posed. On the other hand even for very small values of δ, there is only a small number of polynomials that are δ-close to any given function f, and one can conceive a generalized reconstruction procedure that having oracle access to f outputs a small number of efficient programs so that every polynomial that is δ-close to f is computed by one of these programs. Finding a generalized reconstruction procedure of this form is a version of the "list decoding" problem for error-correcting codes (though we will not use the phrase "list decoding" in most of our exposition). Such a generalized reconstruction procedure has been given for the first time only very recently by Arora and Sudan [AS97]. The procedure of Arora and Sudan has a complicated analysis that relies on their difficult analysis of the low-degree test for the "highly noisy" case. In this paper we present a reconstruction procedure that works for an even wider range of parameters and has a much simpler proof. We also show that our reconstruction procedure is strong enough to imply the hardness amplification result of [IW97] (but even the weaker procedure of Arora and Sudan [AS97] would have sufficed). It has been pointed out to us that the connection between (generalized) polynomial reconstruction and hardness amplification has also been observed by Avi Wigderson [Wig98] and S. Ravi Kumar and D. Sivakumar [KS98].

2 Preliminaries

We write U_n for the uniform distribution on $\{0,1\}^n$. The *statistical difference* between two random variables X and Y on a universe U is defined to be $\max_{S \subset U} |\Pr[X \in S] - \Pr[Y \in S]|$.

Our main objects of study are pseudorandom generators:

Definition 2 *A function $G: \{0,1\}^d \to \{0,1\}^n$ is an (s, ε)* pseudorandom generator *if no circuit of size s can distinguish G from U_n with advantage greater than ε. That is, for every circuit C of size s,*

$$|\Pr[C(U_n) = 1] - \Pr[C(G(U_d)) = 1]| \leq \varepsilon.$$

We begin by recalling the Nisan–Wigderson construction of pseudorandom generators.

3 The Nisan–Wigderson generator

The combinatorial construction underlying the NW generator is a collection of sets with small intersections, called a *design*.

Lemma 3 (design [NW94]) *For every $\ell, m \in \mathbb{N}$, there exists a a family of sets $S_1, \ldots, S_m \subset \{1, \ldots, d\}$ such that*
1. $d = O(\ell^2/\log m)$, 2. For all i, $|S_i| = \ell$, and 3. For all $i \neq j$ $|S_i \cap S_j| \leq \log m$. Moreover, such a family can be found deterministically in time $\mathrm{poly}(m, 2^d)$

For concreteness, one can think of $m = 2^{\theta \ell}$ for some small constant $\theta > 0$, so that $d = O(\ell) = O(\log m)$. Given such a family of sets, the NW generator takes a uniformly distributed string of length d and produces m strings of length ℓ. That is, given parameters ℓ and m, we take the family of sets given by Lemma 3 and define $\mathrm{NW}_{\ell,m}: \{0,1\}^d \to (\{0,1\}^\ell)^m$ by

$$\mathrm{NW}_{\ell,m}(x) = (x_{S_1}, x_{S_2}, \ldots, x_{S_m}),$$

where x_{S_i} denotes the projection of x onto the coordinates specified by S_i.

The key property of this generator used in [NW94, IW97] is that the strings x_{S_i} behave as if they are independent when they are used as inputs to a hard function. Let $P: \{0,1\}^\ell \to \{0,1\}$ be any predicate. Then the NW pseudorandom generator using P is a function $\mathrm{NW\text{-}PRG}^P_{\ell,m}: \{0,1\}^d \to \{0,1\}^m$ given by

$$\mathrm{NW\text{-}PRG}^P_{\ell,m}(x) = P(x_1)P(x_2)\cdots P(x_m),$$

$$\text{where } (x_1, \ldots, x_m) = \mathrm{NW}_{\ell,m}(x)$$

[5] An extractor is an efficient algorithm that on input a distribution sampled from a distribution with high min-entropy has an output that is statistically close to uniform.

[6] We say that a function f has δ agreement with a function g if f and g are equal on a fraction at least δ of their domain.

The main theorem of [NW94] is that if P is taken to be a sufficiently hard (on average) predicate, $\text{NW-PRG}^P_{\ell,m}$ is a good pseudorandom generator.

Theorem 4 ([NW94]) *Suppose $P\colon \{0,1\}^\ell \to \{0,1\}$ is a predicate such that no circuit of size s can compute P correctly on more than a fraction $\frac{1}{2}+\frac{\varepsilon}{m}$ of the inputs. $\text{NW-PRG}_{\ell,m}$ is an $(s - O(m^2 \log m), \varepsilon)$ pseudorandom generator.*

The pseudorandom generators produced by this theorem can be spectacular, as the seed length $d = O(\ell^2/\log m)$ can be much smaller than (even logarithmic in) the number of output bits if P is sufficiently hard. The main drawback is that the hypothesis is also extremely strong (in that P must be very hard on average), and much work has been done to construct predicates that are strong enough for Theorem 4 based on weaker assumptions [BFNW93, Imp95, IW97, IW98]. In the next section, we analyze the quality of this generator when only a mildly hard predicate is used.

4 Pseudorandom generators via pseudoentropy

In this section, we show how to build a pseudorandom generator out of a mildly hard predicate in a different (and arguably more direct) way than [IW97]. Specifically, we show how to directly build a "pseudoentropy generator" from a mildly hard predicate and argue that applying an extractor to its output gives a pseudorandom generator.

4.1 Using a mildly hard predicate

Intuitively, the reason the NW pseudorandom generator works is that whenever x_i is a "hard instance" of P, $P(x_i)$ is indistinguishable from a random bit. If P is very hard as in the hypothesis of Theorem 4, then almost all inputs are hard instances. Thus, with high probability all the x_i's will be hard instances and the limited dependence of the x_i's guarantees that the $P(x_i)$'s will look simultaneously random.

Now suppose that P is instead only mildly hard, in the sense that no small circuit can compute correctly on more than a $1-\delta$ fraction of inputs, for some small but noticeable δ. Intuitively, this means that some δ fraction of the inputs are extremely hard for P. Thus, we'd expect that a δ fraction of the output bits of $\text{NW-PRG}^P_{\ell,m}$ are indistinguishable from random, so that we should get some crude pseudorandomness out of the generator. In fact, this intuition about hard instances can be made precise, using the following result of Impagliazzo [Imp95].

Theorem 5 (hardcore sets [Imp95]) *Suppose no circuit of size s can compute $P\colon \{0,1\}^\ell \to \{0,1\}$ on more than a $1-\delta$ fraction of the inputs in $\{0,1\}^\ell$. Then, for every $\varepsilon > 0$, there exists an ε-*hardcore set *$H \subset \{0,1\}^\ell$ such that $|H| = \delta \cdot 2^\ell$ and no circuit of size $s' = \Omega(\varepsilon^2\delta^2 s)$ can compute P correctly on more than a $\frac{1}{2}+\varepsilon$ fraction of the inputs in H.*

Using this theorem, we can prove something about the output of $\text{NW-PRG}^P_{\ell,m}$ when a mildly hard predicate P is used. Notice that if x is chosen uniformly at random, then each component $x_i = x_{S_i}$ of the output of $\text{NW}_{\ell,m}(x)$ is uniformly distributed in $\{0,1\}^\ell$. Hence, the *expected* number of x_i's that land in H is δm. Thus, the earlier intuition suggests that the output of $\text{NW-PRG}^P_{\ell,m}$ should have δm bits of pseudorandomness, and this is in fact true.

Theorem 6 *Suppose no circuit of size s can compute $P\colon \{0,1\}^\ell \to \{0,1\}$ on more than a $1-\delta$ fraction of the inputs in $\{0,1\}^\ell$. Then, for every $\varepsilon > 0$, there is a distribution D on $\{0,1\}^m$ of (Shannon) entropy*[7] *at least δm such that no circuit of size $s' = \Omega(\varepsilon^2/m^2)\cdot s - O(m^2\log m)$ can distinguish the output of $\text{NW-PRG}^P_{\ell,m}\colon \{0,1\}^d \to \{0,1\}^m$ from D with advantage greater than ε.*

Proof: Let H be a $(\varepsilon/\delta m)$-hardcore set for P, as given by Theorem 5. We will show that the following distribution satisfies the requirements of the theorem.

Distribution D: Choose x uniformly from $\{0,1\}^d$. Let $(x_1,\ldots,x_m) = \text{NW}(x)$. If $x_i \in H$, select $b_i \in \{0,1\}$ uniformly at random, and if $x_i \notin H$ let $b_i = P(x_i)$. Output $b_1 \cdots b_m$.[8]

First, we argue that the entropy of D is at least δm. Define $N(x_1,\ldots,x_m)$ to be the number of x_i's that are in H. Then for any $x \in \{0,1\}^d$, the entropy of $D|_x$ (i.e., D conditioned on x) is $N(\text{NW}(x)))$. By the definition of the Nisan-Wigderson generator, each x_i is (individually) uniformly distributed and therefore lands in H with probability δ. By linearity of expectations, the expectation of $N(\text{NW}(x))$ (over uniformly selected x) is δm. Thus, since conditioning reduces entropy (cf., [CT91, Th. 2.6.5]),

$$\begin{aligned} \mathrm{H}(D) &\geq \mathop{\mathrm{E}}_x[\mathrm{H}(D|_x)] \\ &= \mathop{\mathrm{E}}_x[N(\text{NW}(x))] \\ &= \delta m \end{aligned}$$

Now we show that D and $\text{NW-PRG}^P_{\ell,m}$ are computationally indistinguishable. Suppose that some circuit C distinguishes the output of $\text{NW-PRG}_{\ell,m}$ from D with advantage greater than ε. We will show that C must be of size at least $\Omega(\varepsilon^2/m^2)\cdot s - O(m^2 \log m)$. By complementing C if necessary, we have

$$\Pr\left[C(\text{NW-PRG}^P_{\ell,m}(U_d)) = 1\right] - \Pr\left[C(D) = 1\right] > \varepsilon.$$

For $x \in \{0,1\}^\ell$ and $r \in \{0,1\}$, define

$$Q(x,r) = \begin{cases} r & \text{if } x \in H \\ P(x) & \text{otherwise.} \end{cases}$$

Now consider "hybrids" $D_0,\ldots,D_m$ of D and $\text{NW-PRG}^P_{\ell,m}(U_d)$ defined as follows:

Distribution D_j: Choose x uniformly from $\{0,1\}^d$ and choose $r_1,\ldots,r_m$ uniformly from $\{0,1\}$. For $j = 1,\ldots,m$, let $p_j = P(x_{S_j})$ and $q_j = Q(x_{S_j}, r_j)$. Output $p_1 \cdots p_i q_{i+1} \cdots q_m$.

Thus, $D_0 = \text{NW-PRG}^P(U_d)$, and $D_m = D$. By the "hybrid argument" of [GM84] (cf. [Gol95, Sec. 3.2.3]), there is an i such that

$$\begin{aligned} \varepsilon/m &< \Pr[C(D_{i-1})=1] - \Pr[C(D_i)=1] \\ &= \delta\cdot\Pr[C(D_{i-1})=1 \mid x_{S_i}\in H] \\ &\quad + (1-\delta)\Pr[C(D_{i-1})=1 \mid x_{S_i}\notin H] \\ &\quad - \delta\cdot\Pr[C(D_i)=1 \mid x_{S_i}\in H] \\ &\quad - (1-\delta)\Pr[C(D_i)=1 \mid x_{S_i}\notin H] \\ &= \delta\cdot\Pr[C(D_{i-1})=1 \mid x_{S_i}\in H] \\ &\quad - \delta\cdot\Pr[C(D_i)=1 \mid x_{S_i}\in H], \end{aligned}$$

[7] Recall that the *(Shannon) entropy* of a distribution D is $\mathrm{H}(D) = \sum_\alpha \Pr[D=\alpha]\log(1/\Pr[D=\alpha])$.

[8] A similar "bit-flipping" technique is used in [HILL98] to prove their construction of a false entropy generator.

where the last equality is because D_{i-1} and D_i are identical conditioned on $x_{S_i} \notin H$. Expanding and using the fact that $q_i = Q(x_{S_i}, r_i) = r_i$ when $x_{S_i} \in H$, we have

$$\Pr_{x,\overline{r}}[C(p_1 \cdots p_{i-1} r_i q_{i+1} \cdots q_m) = 1 \mid x_{S_i} \in H] - \Pr_{x,\overline{r}}[C(p_1 \cdots p_{i-1} p_i q_{i+1} \cdots q_m) = 1 \mid x_{S_i} \in H] > \frac{\varepsilon}{\delta m},$$

where x is chosen uniformly in $\{0,1\}^d$ and $r_i, \ldots, r_m$ are selected uniformly in $\{0,1\}$. Renaming r_i as b and using the standard transformation from distinguishers to predictors [Yao82] (cf. [Gol98, Sec. 3.3.3]), we see that

$$\Pr_{x,b,\overline{r}}[C(p_1 \cdots p_{i-1} b q_{i+1} \cdots q_m) \oplus b = p_i \mid x_{S_i} \in H] > \frac{1}{2} + \frac{\varepsilon}{\delta m}$$

Recall that p_j (resp., q_j) is defined to be $P(x_{S_j})$ (resp., $Q(x_{S_j}, r_j)$), where x_{S_j} is the projection of x onto the bits specified by S_j. Using an averaging argument we can fix $r_{i+1}, \ldots, r_m$, b, and all the bits of x outside S_i while preserving the advantage in predicting $p_i = P(x_{S_i})$. Renaming x_{S_i} as z, we now observe that z varies uniformly over H while p_j for $j < i$ and q_j for $j > i$ are now functions P_j of z that depend on only $|S_i \cap S_j| \le \log m$ bits of z. So, we have

$$\Pr_z[C(P_1(z) \cdots P_{i-1}(z) b P_{i+1}(z) \cdots P_m(z)) \oplus b = P(z)] > \frac{1}{2} + \frac{\varepsilon}{\delta m}.$$

Each P_j can be computed by a circuit of size $O(m \log m)$, since every function of $\log m$ bits can be computed by a circuit of that size. Incorporating these circuits and b into C, we obtain a circuit C' of size $\mathrm{size}(C) + O(m^2 \log m)$ such that

$$\Pr_z[C'(z) = P(z)] > \frac{1}{2} + \frac{\varepsilon}{\delta m}.$$

Now, since H is $(\varepsilon/\delta m)$-hardcore for P as in Theorem 5, C' must have size greater than $\Omega(\delta^2 \cdot (\varepsilon^2/\delta m)^2) \cdot s = \Omega(\varepsilon^2/m^2) \cdot s$, and hence C must have size greater than $\Omega(\varepsilon^2/m^2) \cdot s - O(m^2 \log m)$. ■

Thus, using a mildly hard predicate with the NW generator, we can obtain many bits of crude pseudorandomness. A natural next step would be to try to "extract" this crude pseudorandomness and obtain an output that is indistinguishable from the uniform distribution. Unfortunately, one cannot hope to extract uniformly distributed bits from a distribution that just has high Shannon entropy. Extraction is only possible from distributions that have high *min-entropy*. Recall that a distribution D on a finite set S is said to have min-entropy k if for all $x \in D$, $\Pr[X = x] \le 2^{-k}$.

The reason that we were only able to argue about Shannon entropy in Theorem 6 is that we could only say that δm x_i's land in H *on average*. To obtain a result about min-entropy, we would need to guarantee that many x_i's lie in H *with high probability*. Clearly, this would be the case if the x_i's were generated pairwise independently instead of via the NW generator. But we also need the special properties of the NW generator to make the current argument about indistinguishability work. Following [IW97], we resolve this dilemma by taking the XOR of the two generators to obtain a new generator with the randomness properties of each.[9] That is, we obtain $x_1, \ldots, x_m$ from a seed x using the NW generator, we obtain $y_1, \ldots, y_m$ pairwise independent from a seed y, and then use $z_1 = x_1 \oplus y_1, \ldots, z_m = x_m \oplus y_m$ as the inputs to the predicate P. As we will see in the next section, this gives a generator whose output is indistinguishable from some distribution with high min-entropy, as desired.

[9] [IW97] take the XOR of the NW generator with a generator coming from a random walk on an expander.

4.2 A pseudoentropy generator.

The following definition (following [HILL98]) formalizes the type of generator we obtain.

Definition 7 *A generator* $G\colon \{0,1\}^d \to \{0,1\}^m$ *is a* (k, s, ε) pseudoentropy generator *if there is a distribution D on $\{0,1\}^m$ of min-entropy k such that no circuit of size s can distinguish the output of G from D with advantage greater than ε.*

Remark 8 The above definition differs from that of [HILL98] in several ways. Most importantly, we require the output to be indistinguishable from having high min-entropy, whereas they only require that it be indistinguishable from having high Shannon entropy. They later convert to the Shannon entropy to min-entropy by taking many samples on independent seeds, but we cannot afford the extra randomness needed to do this. Other differences are that we ask for indistinguishability against circuits rather than uniform adversaries, that we do not require that G be computable in polynomial time, and that we do not explicitly ask that k be larger than d (though the notion is uninteresting otherwise).

Recall that we need a way of generating many pairwise independent strings from a short seed.

Lemma 9 ([CG89] (see also [Gol97a])) *For any $\ell \in \mathbb{N}$ and $m \le 2^\ell$, there is a generator $\mathrm{PI}_{\ell,m}\colon \{0,1\}^{3\ell} \to (\{0,1\}^\ell)^m$ such that for y selected uniformly at random, the random variables $\mathrm{PI}_{\ell,m}(y)_1, \ldots, \mathrm{PI}_{\ell,m}(y)_m$ are pairwise independent. Moreover $\mathrm{PI}_{\ell,m}$ is computable in time $\mathrm{poly}(\ell, m)$.*

Let $P\colon \{0,1\}^\ell \to \{0,1\}$ be any predicate, let m be any positive integer, and let d be the seed length of $\mathrm{NW}_{\ell,m}$. Then our pseudoentropy generator using P is a function $\mathrm{PE}^P_{\ell,m}\colon \{0,1\}^{d+3\ell} \to \{0,1\}^m$ given by

$$\mathrm{PE}^P_{\ell,m}(x,y) = P(x_1 \oplus y_1) P(x_2 \oplus y_2) \cdots P(x_m \oplus y_m),$$

where

$$(x_1, \ldots, x_m) = \mathrm{NW}_{\ell,m}(x) \quad \text{and} \quad (y_1, \ldots, y_m) = \mathrm{PI}_{\ell,m}(y)$$

The following theorem, whose proof can be found in the full version of the paper [STV98], confirms that this construction does in fact yield a pseudoentropy generator.

Theorem 10 *Suppose no circuit of size s can compute $P\colon \{0,1\}^\ell \to \{0,1\}$ on more than a $1 - \delta$ fraction of the inputs in $\{0,1\}^\ell$. Then, for any $m \le 2^\ell$, $\mathrm{PE}^P_{\ell,m}\colon \{0,1\}^{d+3\ell} \to \{0,1\}^m$ is a (k, s', ε) pseudoentropy generator, with*

$$\begin{aligned} \textit{seed length} = d + 3\ell &= O(\ell^2/\log m) \\ \textit{pseudoentropy} = k &= \delta m/2 \\ \textit{adversary size} = s' &= \Omega(1/\delta^2 m^4) \cdot s - O(m^2 \log m) \\ \textit{adversary's advantage} = \varepsilon &= O(1/\delta m) \end{aligned}$$

Moreover, $\mathrm{PE}^P_{\ell,m}$ is computable in time $\mathrm{poly}(m, 2^{\ell^2/\log m})$ with m oracle calls to P.

4.3 Extracting the randomness

The tool we will use to transform our pseudoentropy generator into a pseudorandom generator is an *extractor*.

Definition 11 *A function* $\text{EXT}\colon \{0,1\}^m \times \{0,1\}^d \to \{0,1\}^m$ *is a* (k,ε)-extractor *if for every for every distribution* D *on* $\{0,1\}^m$ *of min-entropy* k, $\text{EXT}(D, U_d)$ *has statistical difference at most* ε *from* U_n.

We will make use of the following recent construction of extractors:

Theorem 12 ([Tre98]) *For every* m, k, *and* ε *such that* $k \le m$, *there is a* (k,ε)*-extractor* $\text{EXT}\colon \{0,1\}^m \times \{0,1\}^d \to \{0,1\}^{\sqrt{k}}$ *such that*

$$d = O\left(\frac{\log^2(m/\varepsilon)}{\log k}\right)$$

and $\text{EXT}\colon \{0,1\}^m \times \{0,1\}^d \to \{0,1\}^{\sqrt{k}}$ *is computable in time* $\text{poly}(m,d)$.

The following lemma (proved in the full version of the paper [STV98]) confirms the intuition that applying an extractor to a distribution that is computationally indistinguishable from a distribution with high min-entropy should yield a distribution that is indistinguishable from uniform.

Lemma 13 *Suppose* $G\colon \{0,1\}^{d_1} \to \{0,1\}^m$ *is a* (k,s,ε_1) *pseudoentropy generator and* $\text{EXT}\colon \{0,1\}^m \times \{0,1\}^{d_2} \to \{0,1\}^n$ *is a* (k,ε_2)*-extractor computable by circuits of size* t. *Then* $G'\colon \{0,1\}^{d_1+d_2} \to \{0,1\}^n$ *defined by* $G'(u,v) = \text{EXT}(G(u),v)$ *is a* $(s-t, \varepsilon_1+\varepsilon_2)$ *pseudorandom generator.*

Summing up, we have the following theorem:

Theorem 14 *There is a universal constant* $\gamma > 0$ *such that the following holds. Let* $P\colon \{0,1\}^\ell \to \{0,1\}$ *be any predicate such that no circuit of size* s *can compute* P *correctly on more than a* $1-\delta$ *fraction of the inputs, where* $s \le 2^\ell$ *and* $\delta \ge s^{-\gamma}$. *Define* $n = s^\gamma$ *and* $m = 2n^2/\delta$ *and let* $\text{PE}^P_{\ell,m}\colon \{0,1\}^{d_1} \to \{0,1\}^m$ *be the* $(\delta m/2, \Omega(1/\delta^2 m^4)\cdot s - O(m^2 \log m), O(1/\delta m))$ *pseudoentropy generator of Theorem 10 and let* $\text{EXT}\colon \{0,1\}^m \times \{0,1\}^{d_2} \to \{0,1\}^n$ *be the* $(\delta m/2, 1/\delta m)$*-extractor of Theorem 12. Let* $\text{PE-PRG}^P\colon \{0,1\}^{d_1+d_2} \to \{0,1\}^n$ *be defined by*

$$\text{PE-PRG}^P(u,v) = \text{EXT}(\text{PE}^P_{\ell,m}(u), v).$$

Then, PE-PRG^P *is a* (s',ε) *pseudorandom generator with*

$$\begin{aligned} \textit{output length} = n &= s^\gamma \\ \textit{seed length} = d_1 + d_2 &= O\left(\frac{\ell^2}{\log s}\right) \\ \textit{adversary size} = s' &= \sqrt{s} \\ \textit{adversary's advantage} = \varepsilon &= O(1/n^2), \end{aligned}$$

Moreover, PE-PRG^P *can be evaluated in time* $2^{O(\ell^2/\log s)}$ *with* $O(n^2/\delta)$ *oracle calls to* P.

In particular, suppose P is a predicate in E such that no circuit of size $s = 2^{\theta\ell}$ can compute P correctly on more than a $1-\delta = 1 - 1/\text{poly}(\ell)$ fraction of the inputs. Then the output length is $n = 2^{\Omega(\ell)}$, the seed length is $O(\ell) = O(\log n)$, no circuit of size $s' = 2^{\Omega(\ell)}$ can distinguish the output from uniform, and the generator can be evaluated in time $\text{poly}(n)$, so the resulting pseudorandom generator is sufficiently strong to obtain $\mathsf{P} = \mathsf{BPP}$.

Proof: By Theorem 10,

$$d_1 = O\left(\frac{\ell^2}{\log m} + \ell\right) \le O\left(\frac{\ell^2}{\log s}\right).$$

By Theorem 12,

$$d_2 = O\left(\frac{\log^2\left(\frac{m}{1/\delta m}\right)}{\log(\delta m/2)}\right) = O(\log s) \le O\left(\frac{\ell^2}{\log s}\right),$$

and EXT is computable in time $t = \text{poly}(m, d_2) = \text{poly}(m)$. By Lemma 13, no circuit of size s' can distinguish the output of PE-PRG from uniform wth advantage greater than $O(1/\delta m) = O(1/n^2)$, where

$$s' = \Omega(1/\delta^2 m^4)\cdot s - O(m^2 \log m) - t \ge \Omega(s^{1-10\gamma}) - \text{poly}(s^\gamma)$$

By choosing γ sufficiently small, s' will always be at least $\sqrt{s}$. ■

Remark 15 As mentioned earlier, Håstad et al. [HILL98] introduced the notion of a pseudoentropy generator and showed that the crude pseudorandomness of such a generator can be extracted to yield a pseudorandom generator. Their work is in the "cryptographic" setting, in which the generators must be computable in time polynomial in the *seed length* and hence one can only hope for the output to be polynomially longer than the seed (rather than exponentially, as we obtain). Hence throughout their construction they can afford super-linear increases in seed length, whereas preserving the seed length up to linear factors is crucial for obtaining pseudorandom generators good enough for $\mathsf{P} = \mathsf{BPP}$. For example, they can afford to use randomness-inefficient extractors such as 2-universal hash functions, whereas we require extractors which use only a logarithmic number of truly random bits, which have only been constructed recently [Zuc96, Tre98].[10]

Remark 16 The output of the pseudoentropy generator $\text{PE}^P_{\ell,m}$ constructed in Theorem 10 is actually "nicer" than stated. Specifically, it is indistinguishable from a *oblivious bit-fixing source* — that is, a distribution on strings of length m in which $m-k$ bit positions are fixed and the other k bit positions vary uniformly and independently. Such sources were the focus of the "bit extraction problem" studied in [Vaz85, BBR85, CGH$^+$85, Fri92] and the term "oblivious bit-fixing source" was introduced in [CW89]. To see that the output of $\text{PE}^P_{\ell,m}$ is indistinguishable from an oblivious bit-fixing source, simply observe that the distribution D given in the proof of Theorem 10 is such a source.[11] Extracting from oblivious bit-fixing sources in which all but k bits are fixed is an easier task than extracting from a general source of min-entropy k, and already in [CW89] there are (implicitly) extractors sufficient for our purposes.

5 Algebraic amplification of hardness

Recall the main theorem of Nisan and Wigderson (Theorem 4) that states that given a hard predicate $P\colon \{0,1\}^\ell \to \{0,1\}$, one can get

[10] Indeed, the term "extractor" was not even present at the time of [HILL98] and the first constructions of randomness-efficient extractors used their Leftover Hash Lemma as a starting point.

[11] Actually, D is a *convex combination* of oblivious bit-fixing sources. Distribution X is said to be a convex combination of distributions $X_1, \ldots, X_t$ if there is a distribution on I on $\{1,\ldots,t\}$ such that X can be realized by choosing $i \in \{1,\ldots,t\}$ according to I, taking a sample x from X_i, and outputting x. It is easy to see that any extractor for oblivious bit-fixing sources also works for convex combinations of them.

a pseudorandom generator. The requirement of a hard predicate P in this theorem can be replaced with a requirement of a hard function (with many bits of output) using the hardcore predicate construction of Goldreich and Levin [GL89] (see also [Gol97b]). Given a function $g\colon \{0,1\}^k \to \{0,1\}^\ell$, the hardcore predicate for g is the function $\mathrm{GL}_g\colon \{0,1\}^{k+\ell} \to \{0,1\}$ given by

$$\mathrm{GL}_g(x,r) = \langle g(x), r\rangle, \text{ for } x \in \{0,1\}^k \text{ and } r \in \{0,1\}^\ell,$$

where $\langle u, v\rangle$ denotes the mod-2 inner product of $u = (u_1,\ldots,u_\ell)$ and $v = (v_1,\ldots,v_\ell)$ (i.e., $\sum_{i=1}^{\ell} u_i v_i \pmod 2$).

Theorem 17 ([GL89]) *There exists a constant c s.t. the following holds. If $g\colon \{0,1\}^k \to \{0,1\}^\ell$ is a function such that no circuit of size s can compute g correctly on more than an ε fraction of the inputs, then $\mathrm{GL}_g\colon \{0,1\}^{k+\ell} \to \{0,1\}$ is a predicate such that no circuit of size $s' = \Omega((\frac{\varepsilon}{k\ell})^c \cdot s)$ can compute P correctly on more than a fraction $\frac{1}{2} + 2\varepsilon$ of the inputs.*

Thus the two theorems above show that it suffices to construct hard *functions* to obtain pseudorandom generators. The approach of Impagliazzo and Wigderson is to start from a predicate P that is hard in the worst case (i.e., no small circuit computes it correctly on all inputs); then to use a low-degree extension of P to obtain a polynomial function $\hat{p}$ that is mildly hard on the average. They then apply two different XOR lemmas to obtain a functions that grow harder; eventually obtaining as hard a function as required in Theorem 17. We use an alternate approach for this sequence by showing directly that the function $\hat{p}$ above is very hard; as hard as required for the combination of Theorems 4 and 17. We start by specifying the properties of the low-degree extension; a proof of following standard lemma can be found in the full version of this paper [STV98].

Proposition 18 *For every δ, ℓ and predicate $P\colon \{0,1\}^\ell \to \{0,1\}$, there exists m, $d \in \mathbb{N}$, a field F, and a polynomial $\hat{p}\colon F^m \to F$ (the low-degree extension) of total degree d satisfying the following properties:*

1. *$m \log|F| \le 4\ell$, $\frac{d}{|F|} \le \delta$, and $|F| \le \mathrm{poly}(\ell/\delta)$.*
2. *If T_P and $T_{\hat{p}}$ denote the worst-case computation times (or circuit sizes) for P and $\hat{p}$ respectively, then*

   $$T_P \le T_{\hat{p}} \le \mathrm{poly}(2^\ell T_P)$$

   ($\hat{p}$ is harder than P but not too much harder, especially if P has time complexity $2^{O(\ell)}$.)

In the following section we prove the following theorem.

Theorem 19 *There exists a constant c s.t. if some function $f\colon F^m \to F$ computed by a circuit of size s agrees with a degree d polynomial $\hat{p}\colon F^m \to F$ on $\varepsilon \ge c\sqrt{d/|F|}$ fraction of the inputs, then there exists a circuit of size $s \cdot \mathrm{poly}(md\log|F|/\varepsilon)$ that computes $\hat{p}$ correctly everywhere.*

Putting together the above we get:

Theorem 20 *There exists a universal constant $\gamma > 0$ such that the following holds: Let $P\colon \{0,1\}^\ell \to \{0,1\}$ be a function that is not computed by any circuit of size s, where $s \le 2^\ell$. Let $m, d, F, \hat{p}$ be as guaranteed to exist by Proposition 18 for $\delta = s^{-7\gamma}$. Let $\ell' = 4\ell + \log|F|$ and $Q\colon \{0,1\}^{\ell'} \to \{0,1\}$ be given by $Q = \mathrm{GL}_{\hat{p}}$ (where $\hat{p}$ is now viewed as a function from 4ℓ bits to $\log|F|$ bits). Then, for $n = s^\gamma$, $\mathrm{NW\text{-}PRG}^Q_{\ell',n}\colon \{0,1\}^t \to \{0,1\}^n$ is an (s',ε) pseudorandom generator with*

$$\begin{aligned}
\textit{output length} = n &= s^\gamma\\
\textit{seed length} = t &= O\left(\frac{\ell^2}{\log s}\right)\\
\textit{adversary size} = s' &= \sqrt{s}\\
\textit{adversary's advantage} = \varepsilon &= O(1/n^2),
\end{aligned}$$

Moreover, $\mathrm{PE\text{-}PRG}^P$ can be evaluated in time $2^{O(\ell^2/\log s)}$ with access to the entire truth table of P.

5.1 The reconstruction procedure

First, a bit of terminology:

Definition 21 *A randomized procedure A is said to* compute *a function $f\colon X \to Y$ at a point $x \in X$ if $\Pr[A(x) = f(x)] \ge 3/4$, where the probability is taken over the internal coin tosses of A. We say that A has* agreement $\alpha \in [0,1]$ *with f if A computes f on an α fraction of the inputs in D. We say that A* computes *f if it has agreement 1 with f.*

Now we prove Theorem 19 by providing a uniform solution to the following "multivariate reconstruction problem".

Given: An oracle $f\colon F^m \to F$ and parameters $d \in \mathbb{N}$ and $\varepsilon \in \mathbb{R}$.
Goal: Reconstruct (an implicit representation for) every polynomial that has ε-agreement with the function f. Specifically, construct randomized oracle machines $M_1,\ldots,M_k$ such that for every polynomial $p\colon F^m \to F$ of degree d that has (relative) agreement ε with f, there exists $j \in [k]$ such that M_j^f computes p.

We will be interested in the running time of the "reconstruction procedure", i.e., the time taken to generate the machines $M_1,\ldots,M_k$, as well as the running times of the machines $M_1,\ldots,M_k$.

Theorem 22 *There exists a constant c such that the reconstruction problem above can be solved in time $\mathrm{poly}(md\log|F|/\varepsilon)$, with the running time of each of the oracle machines listed in the output being $\mathrm{poly}(md\log|F|/\varepsilon)$, provided $\varepsilon > c\sqrt{d/|F|}$.*

Remark 23
1. This theorem is a strengthening of a theorem due to [AS97]. In particular, the lower bound on ε here is smaller than that of [AS97], who obtain an unspecified polynomial in d and $\frac{1}{|F|}$. Furthermore, our proof is simpler and in particular does not require "low-degree testing."
2. The bound of $\Omega(\sqrt{d/|F|})$ is within a constant factor of the bound for the univariate case. The constant c above is not optimized in this writeup. But our methods can push it down to any constant greater than 2. For the univariate case, this constant is 1. No inherent reason is known for the gap.

Observe that Theorem 19 follows immediately from Theorem 22. We have formulated the multivariate reconstruction problem in such a way that an efficient solution to it (as in Theorem 22) immediately implies a hardness amplification result. It should be clear that the connection actually applies to any error-correcting code for which there exist efficient algorithms for encoding (as in Proposition 18) and "list-decoding" (that is, finding efficient oracle machines for every codeword that agrees with a given oracle f in sufficiently many places). The "rate" of the code determines how the input length of the function changes when we encode it (e.g. in Proposition 18, input length ℓ becomes $m\log|F| \le 4\ell$ and linear growth like this is needed to obtain the result of [IW97]). The amount of agreement for which the list-decoding works determines how hard-on-average the encoded function is. The running time of the oracle machines produced in the reconstruction procedure determines the circuit size for which the hardness amplification works.

We now move on to the proof of Theorem 22. Fix an oracle $f: F^m \to F$ and a degree d polynomial $p: F^m \to F$ with δ agreement with f. We observe that it suffices to reconstruct a (randomized) oracle machine M such that M^f has sufficiently high agreement with p. This is due to the existence of "self-correctors" of polynomials [BF90, Lip89, GLR$^+$91, GS92]. Specifically, we use the following theorem:

Theorem 24 ([GLR$^+$91]) *There exists a randomized oracle machine* Corr *taking as parameters integers d and m and a field F such that on access to an randomized oracle $M: F^m \to F$ with agreement $\frac{15}{16}$ with some degree d polynomial p,* Corr^M *computes p in time $\mathrm{poly}(d, m)$ provided $|F| \geq 2(d+1)$.*

As in the algorithms of [BF90, Lip89, GLR$^+$91], we use the properties of "lines" in the m-dimensional space F^m, defined below.

Definition 25 *The* line through $x, y \in F^m$, *denoted $l_{x,y}$, is the parametrized set of points $\{l_{x,y}(t) \overset{\mathrm{def}}{=} (1-t)x + ty \mid t \in F\}$. Given a function $f: F^m \to F$, f restricted to the line $l_{x,y}$ is the function $f|_{l_{x,y}}: F \to F$ given by $f|_{l_{x,y}}(t) = f(l_{x,y}(t))$.*

Notice that if f is a polynomial of total degree d, then $f|_{l_{x,y}}(t)$ is a univariate polynomial of degree at most d. Our strategy, to reconstruct the value of p at a point x, is to look at a random line going through x. On this line p turns into a univariate polynomial. Furthermore, the random line through the randomly chosen point x is a "pairwise independent" collection of points from F^m. Thus p and f will have agreement close to δ on this line as well. Thus the goal of finding $p(x)$ "reduces" to the goal of reconstructing p restricted to this line, i.e., a univariate reconstruction problem, a problem that has been addressed in [ALRS92, Sud97, GS98]. In particular, we use the following theorem.

Theorem 26 ([Sud97]) *Given a sequence of n distinct pairs $\{(t_i, v_i)\}_{i=1}^n$, $t_i, v_i \in F$ and integer parameters d, k, a list of all polynomials $g_1, \ldots, g_l$ satisfying $|\{i \in \{1,\ldots,n\} | g_j(t_i) = v_i\}| \geq k$, can be reconstructed in time $\mathrm{poly}(n, \log|F|)$ provided $k > \sqrt{2dn}$. Furthermore $l \leq \frac{2n}{k}$.*

We describe a family of reconstruction procedures, $\{M_{z,a}\}_{z \in F^m, a \in F}$, that will be used to construct the machines M_1, ..., M_k. To gain some intuition into the procedure below, it may be helpful to consider only the machines $M_{z,p(z)}$. The machines take as parameters a positive real number ε, integers d and m, and a field F.

- $M_{z,a}(x)$:
  1. (Explicitly) find a list of distinct (univariate) polynomials $g_1, \ldots, g_l$ such that this list includes all polynomials that have agreement at least $\varepsilon/2$ with $f|_{l_{z,x}}$ and does not include any polynomial with agreement less than $\varepsilon/4$.
  2. If there exists a unique index $i \in \{1, \ldots, l\}$ such that $g_i(0) = a$, then output $g_i(1)$, else output anything.

Remark 27 1. Step 1 above can be computed in time polynomial in $1/\varepsilon$, $\log|F|$ and d as follows: If F is small enough, then we let $t_1, \ldots, t_n$ be all the elements of F and invoke Theorem 26 on the set $\{(t_i, f(l_{z,x}(t_i)))\}_{i=1}^n$ with $k = \varepsilon n/2$. (Note that $k > \sqrt{2dn}$ as long as $\varepsilon > 2\sqrt{d/|F|}$, which is true by hypothesis.) If F is too large to do this, then set $n = \mathrm{poly}(d/\varepsilon)$ and pick $t_1, \ldots, t_n$ distinct at random from F and then invoking Theorem 26 on the set $\{(t_i, f(l_{z,x}(t_i)))\}_{i=1}^n$ with $k = \varepsilon n/4$. Since there are at most $4/\varepsilon$ polynomials with agreement at least $\varepsilon/2$ with $f|_{l_{z,x}}$ (by the "furthermore" part of Theorem 26), the choice of n guarantees that with high probability, all of these polynomials agree with $f|_{l_{z,x}}$ on at least $\varepsilon n/4$ of the t_i's. As the choice of n also guarantees that $k = (\varepsilon n/4) > \sqrt{2dn}$, Theorem 26 yields a list containing all polynomials with agreement at least $\varepsilon/2$. Now, we wish to discard all polynomials with agreement less than $\varepsilon/4$ — this can be accomplished by comparing each polynomial g obtained with $f|_{l_{z,x}}$ on a random sample of $\mathrm{poly}(1/\varepsilon)$ points from F and discarding it if it has agreement smaller than $\varepsilon/3$ on this sample.

2. The number of polynomials output in Step 1 above is at most $8/\varepsilon$ (by the "furthermore" part of Theorem 26.)

To shed some light on the steps above: We expect that $p|_{l_{z,x}}$ is one of the g_i's returned in Step (1) above. In Step (2) we try to find out which g_i to use by checking to see if there is a unique one which has $g_i(0) = a$ (recall that $p|_{l_{z,x}}(0) = p(z)$), and if so we use this polynomial to output $p(x) = p|_{l_{z,x}}(1) = g_i(1)$. This intuition is made precise in the Section 5.2. We now finish the description of the reconstruction procedure.

- **Reconstruction algorithm.**
  - Repeat the following $O(\log(1/\varepsilon))$ several times:
    1. Pick $z \in F^m$ at random.
    2. Pick $y \in F^m$ at random.
    3. Find a list of univariate polynomials $h_1, \ldots, h_l$ including all polynomials with agreement at least $\varepsilon/2$ with $f|_{l_{z,y}}$.[12]
    4. For every polynomial h_j, include the oracle machine $\mathsf{Corr}^{M_{z,h_j(0)}}$ in the output list.

5.2 Analysis of the polynomial reconstruction procedure

Now we show that the reconstruction algorithm runs in time run in time $\mathrm{poly}(\frac{md}{\varepsilon}\log|F|)$ and outputs a list of oracles that includes one for every polynomial p that has ε agreement with f. Theorem 22 follows immediately.

The claim about the running time is easily verified. To analyze the correctness, it suffices to show that in any iteration of Steps 1–4 in **Reconstruction Algorithm**, an oracle computing p is part of the output with, say, constant probability for any fixed polynomial p of degree d that has ε agreement with f. We show this in two parts. First we argue that for most choices of z, $M_{z,p(z)}$ is an oracle that computes p on 15/16 of all inputs (and thus $\mathsf{Corr}^{M_{z,p(z)}}$ computes p everywhere). Then we show that for most pairs (z, y), there exists j s.t. the polynomial h_j reconstructed in Step 3 satisfies $h_j(0) = p(z)$.

Lemma 28 *There exists a constant c s.t. for every d, F, ε satisfying $1 \geq \varepsilon \geq c\sqrt{d/|F|}$, it is the case that*

$$\Pr_x\left[M_{z,p(z)}(x) = p(x)\right] \geq 15/16,$$

with probability at least $\frac{1}{2}$ over the random choice of $z \in F^m$,

[12] This is done as in Remark 27, though here we do not care if the list contains extra polynomials with low agreement.

Proof: We first argue that when both x and z are picked at random, certain bad events are unlikely to happen. The next two claims describe these bad events and upper bound their probability.

Claim 29 *If* $\varepsilon \geq 16\sqrt{1/|F|}$*, then*

$$\Pr_{x,z}\left[\nexists i \in [l] \text{ s.t. } g_i = p|_{l_{z,x}}\right] \leq 1/64.$$

Proof: For the polynomial $p|_{l_{z,x}}$ not to be included in the output list it has to be the case that p and f do not have $\varepsilon/2$ agreement on the line $l_{z,x}$. But the line is a pairwise independent collection of $|F|$ points in F^m. The quantity of interest then is the probability that a random variable with expectation ε attains an average of at most $\varepsilon/2$ on $|F|$ samples. Using Chebychev's inequality, this probability may be bounded by $\frac{4}{\varepsilon|F|} \leq \frac{4\varepsilon}{16^2} \leq 164$. ■

Claim 30 *If* $\varepsilon \geq 32\sqrt{d/|F|}$*, then*

$$\Pr_{x,z}\left[\exists j \in [l] \text{ s.t. } g_j \neq p|_{l_{z,x}} \text{ and } p|_{l_{z,x}}(0) = g_j(0)\right] \leq \frac{ld}{|F|} \leq 1/64.$$

Proof: For convenience in this argument, assume that $M_{z,p(z)}$ finds *all* polynomials of agreement at least $\varepsilon/4$ with $f|_{l_{z,x}}$ rather than just a subset, as that is clearly the worst case for the claim. Now, instead of picking x and z at random and then letting $g_1, \ldots, g_l$ be all degree d polynomials with $\varepsilon/4$ agreement with $f|_{l_{z,x}}$, we first pick z', x' independently and uniformly at random from F^m; and let $g'_1, \ldots, g'_{l'}$ be all univariate degree d polynomials with $\varepsilon/4$ agreement with $f|_{l_{z',x'}}$. We now pick two distinct elements t_1, t_2 uniformly from F and let $z = l_{z',x'}(t_1)$ and $x = l_{z',x'}(t_2)$. Notice that we can express $z' = l_{z,x}((t_2 - t_1)^{-1} \cdot t_2)$ and $x' = l_{z,x}((t_2 - t_1)^{-1} \cdot (t_2 - 1))$. Thus the lines $l_{z,x}$ and $l_{z',x'}$ contain the same set of points and thus the polynomials $g_i(t) \stackrel{\text{def}}{=} g'_i(t_2 + t \cdot (t_1 - t_2))$ are exactly the set of polynomials with $\varepsilon/4$ agreement with $f|_{l_{z,x}}$. Thus the event "$p|_{l_{z,x}} \neq g_j$ and $p|_{l_{z,x}}(0) = g_j(0)$" is equivalent to the event "$p|_{l_{x',z'}} \neq g'_j$ and $p|_{l_{x',z'}}(t_1) = g'_j(t_1)$", where t_1 is being chosen at random. This probability is at most $\frac{d}{|F|}$ for any fixed j and thus the probability that there exists a j s.t $p|_{l_{x',z'}}(t_1) = g'_j(t_1)$ is at most $l \cdot \frac{d}{|F|}$. From $l \leq \frac{8}{\varepsilon}$ and $1 \geq \varepsilon \geq 32\sqrt{d/|F|}$, the claim follows. ■

Discounting for the two possible bad events considered in Claims 29 and 30, we find that with probability at least $1 - \frac{1}{32}$, there exists a polynomial g_i returned in Step 1 of $M_{z,p(z)}$ such that $g_i = p|_{l_{z,x}}$; furthermore, this is the unique polynomial such that $g_i(0) = p|_{l_{z,x}}(0) = p(z)$. Thus the output is $g_i(1) = p|_{l_{z,x}}(1) = p(x)$.

Thus with probability at least $31/32$, we find that for a random pair (z, x), $M_{z,p(z)}$ computes $p(x)$. An application of Markov's inequality now yields the desired result. ■

Lemma 31 *With probability at least* $1 - \frac{1}{64}$*, one of the polynomials reconstructed in any one execution of Step 3 of* **Reconstruction Algorithm** *is* $p|_{l_{z,y}}$*; and thus one of the oracles created in Step 4 is* $\mathsf{Corr}^{M_{z,p(z)}}$*, provided* $|F|$ *is large enough.*

Proof: As in Claim 29 we argue that p and f have at least $\varepsilon/2$ agreement on the line $l_{z,y}$ and then $p|_{l_{z,y}}$ is one of the polynomials output in this step. Thus one of the oracles created is $\mathsf{Corr}^{M_{z,a}}$ for $a = p|_{l_{z,y}}(0) = p(z)$. ■

Proof of Theorem 22: Fix any degree d polynomial p with ε agreement with f. Combining Lemmas 28 and 31 we find that with probability $31/64$, one of the oracles output by the reconstruction algorithm is $\mathsf{Corr}^{M_{z,p(z)}}$; and z is such that $M_{z,p(z)}$ computes $p(x)$ for at least $15/16$ fraction of x's in F^m; and thus (by Theorem 24) $\mathsf{Corr}^{M_{z,p(z)}}$ computes p on every input.

Repeating the loop $O(\log \frac{1}{\varepsilon})$ times ensures that every polynomial p with ε agreement with f is included in the output with high probability, using the well-known bound that there are only $O(1/\varepsilon)$ such polynomials (cf., [GRS98, Theorem 17]). ■

Acknowledgments

We thank Oded Goldreich, Amnon Ta-Shma, and Avi Wigderson for clarifying discussions and pointing us to some related work.

References

[ACR97] Alexander Andreev, Andrea Clementi, and José Rolim. Worst-case hardness suffices for derandomization: a new method for hardness-randomness trade-offs. In *Proceedings of ICALP'97*, pages 177–187. LNC 1256S, Springer-Verlag, 1997.

[AK97] V. Arvind and J. Köbler. On resource-bounded measure and pseudorandomness. In *Proceedings of the 17th Conference on Foundations of Software Technology and Theoretical Computer Science*, pages 235–249. LNCS 1346, Springer-Verlag, 1997.

[ALRS92] Sigal Ar, Richard J. Lipton, Ronitt Rubinfeld, and Madhu Sudan. Reconstructing algebraic functions from mixed data. In *33rd Annual Symposium on Foundations of Computer Science*, pages 503–512, Pittsburgh, Pennsylvania, 24–27 October 1992. IEEE.

[AS97] Sanjeev Arora and Madhu Sudan. Improved low degree testing and its applications. In *Proceedings of the Twenty-Ninth Annual ACM Symposium on Theory of Computing*, pages 485–495, El Paso, Texas, 4–6 May 1997.

[BBR85] Charles H. Bennett, Gilles Brassard, and Jean-Marc Robert. How to reduce your enemy's information (extended abstract). In Hugh C. Williams, editor, *Advances in Cryptology—CRYPTO '85*, volume 218 of *Lecture Notes in Computer Science*, pages 468–476. Springer-Verlag, 1986, 18–22 August 1985.

[BF90] Donald Beaver and Joan Feigenbaum. Hiding instances in multioracle queries. In *7th Annual Symposium on Theoretical Aspects of Computer Science*, volume 415 of *Lecture Notes in Computer Science*, pages 37–48, Rouen, France, 22–24 February 1990. Springer.

[BFNW93] László Babai, Lance Fortnow, Noam Nisan, and Avi Wigderson. BPP has subexponential time simulations unless EXPTIME has publishable proofs. *Computational Complexity*, 3(4):307–318, 1993.

[BM84] Manuel Blum and Silvio Micali. How to generate cryptographically strong sequences of pseudo-random bits. *SIAM Journal on Computing*, 13(4):850–864, November 1984.

[CG89] Benny Chor and Oded Goldreich. On the power of two-point based sampling. *Journal of Complexity*, 5(1):96–106, March 1989.

[CGH+85] Benny Chor, Oded Goldreich, Johan Hastad, Joel Friedman, Steven Rudich, and Roman Smolensky. The bit extraction problem or t-resilient functions (preliminary version). In *26th Annual Symposium on Foundations of Computer Science*, pages 396–407, Portland, Oregon, 21–23 October 1985. IEEE.

[CPS99] Jin-Yi Cai, A. Pavan, and D. Sivakumar. On the hardness of the permanent. In *16th International Symposium on Theoretical Aspects of Computer Science*, Lecture Notes in Computer

Science, Trier, Germany, March 4–6 1999. Springer-Verlag. To appear.

[CT91] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. Wiley Series in Telecommunications. John Wiley & Sons, Inc., 2nd edition, 1991.

[CW89] Aviad Cohen and Avi Wigderson. Dispersers, deterministic amplification, and weak random sources (extended abstract). In *30th Annual Symposium on Foundations of Computer Science*, pages 14–19, Research Triangle Park, North Carolina, 30 October–1 November 1989. IEEE.

[FF93] Joan Feigenbaum and Lance Fortnow. Random-self-reducibility of complete sets. *SIAM Journal on Computing*, 22(5):994–1005, October 1993.

[FL96] Uriel Feige and Carsten Lund. On the hardness of computing the permanent of random matrices. *Computational Complexity*, 6(2):101–132, 1996.

[Fri92] Joel Friedman. On the bit extraction problem. In *33rd Annual Symposium on Foundations of Computer Science*, pages 314–319, Pittsburgh, Pennsylvania, 24–27 October 1992. IEEE.

[GL89] Oded Goldreich and Leonid A. Levin. A hard-core predicate for all one-way functions. In *Proceedings of the Twenty First Annual ACM Symposium on Theory of Computing*, pages 25–32, Seattle, Washington, 15–17 May 1989.

[GLR$^+$91] Peter Gemmell, Richard Lipton, Ronitt Rubinfeld, Madhu Sudan, and Avi Wigderson. Self-testing/correcting for polynomials and for approximate functions. In *Proceedings of the Twenty Third Annual ACM Symposium on Theory of Computing*, pages 32–42, New Orleans, Louisiana, 6–8 May 1991.

[GM84] Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270–299, April 1984.

[GNW95] Oded Goldreich, Noam Nisan, and Avi Wigderson. On Yao's XOR lemma. Technical Report TR95–050, Electronic Colloquium on Computational Complexity, March 1995. `http://www.eccc.uni-trier.de/eccc`.

[Gol95] Oded Goldreich. *Foundations of Cryptography (Fragments of a Book)*. Weizmann Institute of Science, 1995. Available, along with revised version 1/98, from `http://www.wisdom.weizmann.ac.il/~oded`.

[Gol97a] Oded Goldreich. A computational perspective on sampling (survey). Available from `http://www.wisdom.weizmann.ac.il/~oded/`, May 1997.

[Gol97b] Oded Goldreich. Three XOR lemmas — an exposition. Available from `http://www.wisdom.weizmann.ac.il/~oded/`, November 1997.

[Gol98] Oded Goldreich. *Modern Cryptography, Probabilistic Proofs and Pseudorandomness*, June 1998. To be published by *Springer*.

[GRS98] Oded Goldreich, Ronitt Rubinfeld, and Madhu Sudan. Learning polynomials with queries — the highly noisy case. Technical Report TR98-060, Electronic Colloquium on Computational Complexity, 1998. Preliminary version in FOCS '95.

[GS92] Peter Gemmell and Madhu Sudan. Highly resilient correctors for polynomials. *Information Processing Letters*, 43(4):169–174, 28 September 1992.

[GS98] Venkatesan Guruswami and Madhu Sudan. Improved decoding of Reed-Solomon and algebraic-geometric codes. Technical Report 98–043, Electronic Colloquium on Computational Complexity, 1998. Preliminary version in FOCS '98.

[HILL98] J. Håstad, R. Impagliazzo, L. Levin, and M. Luby. A pseudorandom generator from any one-way function. To appear in *SIAM J. on Computing*, 1998.

[Imp95] Russell Impagliazzo. Hard-core distributions for somewhat hard problems. In *36th Annual Symposium on Foundations of Computer Science*, pages 538–545, Milwaukee, Wisconsin, 23–25 October 1995. IEEE.

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

[IW98] Russell Impagliazzo and Avi Wigderson. Randomness vs. time: De-randomization under a uniform assumption. In *36th Annual Symposium on Foundations of Computer Science*, Palo Alto, CA, November 8–11 1998. IEEE.

[KS98] S. Ravi Kumar and D. Sivakumar. Personal communication, October 1998.

[KvM98] Adam Klivans and Dieter van Melkebeek. Graph nonisomorphism has subexponential size proofs unless the polynomial-time hierarchy collapses. Technical Report TR-98-12, University of Chicago, Department of Computer Science, December 1998. Extended abstract in these proceedings.

[Lip89] Richard Lipton. New directions in testing. In *Proceedings of DIMACS Workshop on Distributed Computing and Cryptography*, 1989.

[NW94] Noam Nisan and Avi Wigderson. Hardness vs randomness. *Journal of Computer and System Sciences*, 49(2):149–167, October 1994.

[NZ96] Noam Nisan and David Zuckerman. Randomness is linear in space. *Journal of Computer and System Sciences*, 52(1):43–52, February 1996.

[STV98] Madhu Sudan, Luca Trevisan, and Salil Vadhan. Pseudorandom generators without the XOR lemma. Technical Report TR98-074, Electronic Colloquium on Computational Complexity, December 1998. `http://www.eccc.uni-trier.de/eccc`.

[Sud97] Madhu Sudan. Decoding of Reed Solomon codes beyond the error-correction bound. *Journal of Complexity*, 13(1):180–193, March 1997.

[Tre98] Luca Trevisan. Constructions of near-optimal extractors using pseudo-random generators. Technical Report TR98-055, Electronic Colloquium on Computational Complexity, 1998. Extended abstract in these proceedings.

[Vaz85] Umesh V. Vazirani. Towards a strong communication complexity theory or generating quasi-random sequences from two communicating slightly-random sources (extended abstract). In *Proceedings of the Seventeenth Annual ACM Symposium on Theory of Computing*, pages 366–378, Providence, Rhode Island, 6–8 May 1985.

[Wig98] Avi Wigderson. Personal communication, October 1998.

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

[Zuc96] David Zuckerman. Simulating BPP using a general weak random source. *Algorithmica*, 16(4/5):367–391, October/November 1996.