Multiplicative Pseudorandom Generators for Nondeterministic Circuits

Alon Dermer
Department of Computer Science, University of Haifa, Israel

Ronen Shaltiel
Department of Computer Science, University of Haifa, Israel

Abstract

The hardness vs. randomness paradigm aims to construct pseudorandom generators (PRGs) based on complexity theoretic hardness assumptions. A seminal result in this area is a PRG construction by [27, 24]. A sequence of works [25, 30, 36, 31] generalized the result of [27, 24] to nondeterministic circuits, and showed that if $\mathsf{E} = \mathsf{DTIME}(2^{O(n)})$ requires nondeterministic circuits of size $2^{\Omega(n)}$, then for every sufficiently large s, and every $\epsilon \geq \frac{1}{s}$, there is an ϵ-PRG $G : \{0,1\}^{r=O(\log s + \log \frac{1}{\epsilon})} \to \{0,1\}^s$ that runs in time $\mathsf{poly}(s)$, and fools size s nondeterministic circuits. In particular, for every size s nondeterministic circuit C,

$$\Pr[C(G(U_r)) = 1] \leq \Pr[C(U_s) = 1] + \epsilon.$$

Applebaum et al. [1] showed that "black-box techniques" cannot achieve such results for $\epsilon = s^{-\omega(1)}$. In order to circumvent this problem, Artemenko et al. [2] suggested a "multiplicative" version of PRGs, which requires that:

$$\Pr[C(G(U_r)) = 1] \leq 2 \cdot \Pr[C(U_s) = 1] + \epsilon.$$

This still gives that $\Pr[C(G(U_r)) = 1]$ is very small, if $\Pr[C(U_s) = 1]$ is very small, and is therefore suitable for applications that only require this consequence. [2] constructed such multiplicative PRGs for $\epsilon = s^{-\omega(1)}$ (based on very strong hardness assumptions).

In this paper, we give an optimal construction of multiplicative PRGs for nondeterministic circuits. More specifically, under the same hardness assumption used for (standard) PRGs for nondeterministic circuits, we show that for every $\epsilon \geq \frac{1}{2^s}$, there is a multiplicative PRG $G : \{0,1\}^{r=O(\log s + \log \frac{1}{\epsilon})} \to \{0,1\}^s$ that runs in time $\mathsf{poly}(s)$ and fools size s nondeterministic circuits.

This gives the optimal seed length under a hardness assumption that is necessary, and provides improvements in several applications of multiplicative PRGs. Our result improves upon the previous multiplicative PRG construction of [2], which uses a stronger hardness assumption against Σ_3-circuits, and where the seed length is the suboptimal $r = O(\log s) + O(\log \frac{1}{\epsilon})^2$. Our result also improves upon the recent multiplicative PRG of Shaltiel [28] that only achieves very small stretch (the output length in [28] is less than twice the seed length).

Our PRG construction borrows ideas from the recent "low stretch" PRG of Shaltiel [28], and the (standard) PRG construction of Shaltiel and Umans [30]. Loosely speaking, we aim to get the "multiplicativity" of the former, and the "large stretch" of the latter. While both approaches generalize the list-decoding results of Sudan, Trevisan and Vadhan [34], the two results are tailored to two very different parameter regimes, and we introduce several new ideas to make the two approaches co-exist.

2012 ACM Subject Classification Theory of computation → Pseudorandomness and derandomization

Keywords and phrases Hardness vsRandomness, Pseudorandomness, Nondeterministic Circuits

Digital Object Identifier 10.4230/LIPIcs.CCC.2026.18

Related Version *Extended Version*: https://eccc.weizmann.ac.il/report/2026/017/

41st Computational Complexity Conference (CCC 2026).
Editor: Dana Moshkovitz; Article No. 18; pp. 18:1–18:20

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

Funding *Alon Dermer*: The research was supported by ISF grant 1006/23.
Ronen Shaltiel: The research was supported by ISF grant 1006/23 and also co-funded by the European Union (ERC, NFITSC, 101097959). Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Research Council. Neither the European Union nor the granting authority can be held responsible for them.

Acknowledgements We want to thank anonymous referees for helpful comments.

1 Introduction

1.1 Pseudorandom Generators With Respect To a Relation

Pseudorandomness is a viewpoint that says that a distribution Z over $\{0,1\}^m$ is "similar" to the uniform distribution U_m, from the point of view of some function $C : \{0,1\}^m \to \{0,1\}$, if the quantities $p_1 = \Pr[C(U_m) = 1]$ and $p_2 = \Pr[C(Z) = 1]$ are "similar". Typically, this similarity is measured by choosing a parameter $0 < \epsilon \leq 1$ and using the standard (additive, double-sided) relation $\overset{ad}{\sim}_\epsilon$ on $[0,1]$ defined as follows:

$$p_1 \overset{ad}{\sim}_\epsilon p_2 \iff |p_2 - p_1| \leq \epsilon,$$

or in other words, Z is pseudorandom for C if:

$$|\Pr[C(Z) = 1] - \Pr[C(U_m) = 1]| \leq \epsilon.$$

In this paper, following [2, 28] we will also consider other relations (as we explain below in Section 1.3). In the definition below, we define pseudorandomness, and pseudorandom generators (PRGs) with respect to an arbitrary relation, and obtain the standard notion as a special case.

▶ **Definition 1** (Pseudorandomness with respect to a relation). *Let $\sim$ be a relation on $[0,1]$. Given a function $C : \{0,1\}^m \to \{0,1\}$, a distribution Z over $\{0,1\}^m$ is* pseudorandom for C with respect to $\sim$, *if*

$$\Pr[C(U_m) = 1] \sim \Pr[C(Z) = 1].$$

We will abbreviate "with respect to" as "w.r.t." for brevity. Given a class $\mathcal{C}$ of functions $C : \{0,1\}^m \to \{0,1\}$, we say that Z is pseudorandom for $\mathcal{C}$ w.r.t. $\sim$, *if it is pseudorandom for every C in the class $\mathcal{C}$ w.r.t. $\sim$.*

We say that Z is ϵ-pseudorandom for $\mathcal{C}$, *if it is pseudorandom for $\mathcal{C}$ w.r.t. $\overset{ad}{\sim}_\epsilon$.*[1]

▶ **Definition 2** (Pseudorandom generators w.r.t. a relation). *Let $\sim$ be a relation on $[0,1]$. A function $G : \{0,1\}^r \to \{0,1\}^m$ is a* PRG *for a class $\mathcal{C}$ w.r.t. $\sim$, if $G(U_r)$ is pseudorandom for $\mathcal{C}$ w.r.t. $\sim$. G is an* ϵ-PRG *for $\mathcal{C}$ if it is a PRG for $\mathcal{C}$ w.r.t. $\overset{ad}{\sim}_\epsilon$.*

[1] We remark that if the class $\mathcal{C}$ is closed under complement then the standard notion is also obtained when using the (one sided) relation

$$p_1 \overset{a}{\sim}_\epsilon p_2 \iff p_2 \leq p_1 + \epsilon,$$

in which the absolute value is removed.

1.2 PRGs for Nondeterministic Circuits from Hardness Assumptions

In this paper we will consider PRGs for the class of functions computable by either deterministic or nondeterministic circuits of a specified size s. We will consider PRGs in the "Nisan-Wigderson setting" in which the PRG will be allowed to run in time $p(s) > s$ for some polynomial $p(s)$. It is well known that such PRGs imply circuit lower bounds, and therefore, explicit constructions of such PRGs (e.g. [24]) require complexity theoretic hardness assumptions. We now discuss a well known (family of) hardness assumptions.

We say that *"E is hard for exponential size circuits of some type"*, if there exists a problem $L \in \mathsf{E} = \mathsf{DTIME}(2^{O(n)})$ and a constant $\beta > 0$, such that for every sufficiently large n, circuits of size $2^{\beta \cdot n}$ (of the specified type) fail to compute the characteristic function of L on inputs of length n. (See full version for a more formal definition).

The assumption that E is hard for exponential size (deterministic) circuits was used by the celebrated paper of Impagliazzo and Wigderson [24] to imply PRGs for (deterministic) circuits. Subsequent work [25, 30, 36, 31] shows how to obtain PRGs for nondeterministic circuits, assuming that E is hard for exponential size nondeterministic circuits.[2] These two results are stated below.

▶ **Theorem 3** ([24]). *If E is hard for exponential size (deterministic) circuits, then for every sufficiently large s, there is an ϵ-PRG, $G : \{0,1\}^{r=O(\log s)} \to \{0,1\}^s$ for size s (deterministic) circuits, with $\epsilon = \frac{1}{s}$. Furthermore, G is computable in time $\mathsf{poly}(s)$.*[3]

▶ **Theorem 4** ([30]). *If E is hard for exponential size nondeterministic circuits, then for every sufficiently large s, there is an ϵ-PRG, $G : \{0,1\}^{r=O(\log s)} \to \{0,1\}^s$ for size s nondeterministic circuits, with $\epsilon = \frac{1}{s}$. Furthermore, G is computable in time $\mathsf{poly}(s)$.*

We remark that in both theorems, it is known that the assumption is necessary for the conclusion. Note that both theorems obtain error $\epsilon = \frac{1}{s}$. It is natural to ask whether it is possible to obtain smaller error of $\epsilon = s^{-\omega(1)}$ (at the cost of increasing the seed length). Unfortunately, we have evidence that "black-box techniques" cannot produce such results, even when starting from significantly stronger assumptions.

▶ **Theorem 5** ((Informal) [1, 20]). *"Black box techniques" cannot give a version of Theorem 3 or Theorem 4 in which $\epsilon = s^{-\omega(1)}$ even if:*

- *The seed length r is increased from $r = O(\log s)$ to $r = s - 1$.*
- *The hardness assumption is strengthened to "E is hard for exponential size Σ_i-circuits", for an arbitrary constant i. Here a Σ_i-circuit is a circuit with special gates that compute a complete language in the complexity class Σ_i^{P}. See precise definition in full version.*
- *The PRG is only w.r.t. $\overset{a}{\sim}_\epsilon$ rather than w.r.t. $\overset{ad}{\sim}_\epsilon$.*

The reader is referred to [1, 20] for a precise formulation of these results. We stress that all proofs in the PRG literature use "black-box techniques".

We remark that "cryptographic PRGs" can do better, and achieve error that is "negligible" in s. However, cryptographic PRGs (even w.r.t. $\overset{a}{\sim}_\epsilon$) do not exist against nondeterministic circuits, because in the cryptographic setting, the adversary has the computational resources to run the PRG, and can easily use nondeterminism to break the PRG.

---

[2] The assumption that E is hard for exponential size nondeterministic circuits is standard, and was used in many results [4, 25, 26, 30, 9, 17, 21, 31, 32, 15, 1, 10, 2, 22, 14, 5, 12, 6, 7, 29, 28]. It can be viewed as a scaled, nonuniform version of the widely believed assumption that $\mathsf{EXP} \neq \mathsf{NP}$.

[3] Here, and in the remainder of the paper, we often set $m = s$, as a size s circuit can receive at most s input bits, and can choose to ignore some of them.

1.3 Multiplicative PRGs

A useful property of a δ-PRG, $G : \{0,1\}^r \to \{0,1\}^m$ with very low error $\delta = s^{-\Omega(1)}$ against size s nondeterministic circuits, is that for every nondeterministic circuit C of size s,

$$p_1 = \Pr[C(U_m) = 1] \leq \delta \;\Rightarrow\; p_2 = \Pr[C(G(U_r)) = 1] \leq 2\delta = O(\delta).$$

That is, if p_1 is guaranteed to be very small, then p_2 is also guaranteed to be very small. We will loosely refer to this property as "preserving small probabilities". Some applications of PRGs for nondeterministic circuits (that we will survey later on) require $\delta = s^{-\omega(1)}$, but only rely on "preserving small probabilities".

Motivated by such applications (and the negative results of Theorem 5) Artemenko et al. [2] and Shaltiel [28] defined a notion of "multiplicative PRGs" that implies "preserving small probabilities" (and can be constructed under hardness assumptions). More specifically, inspired by differential privacy, they consider PRGs w.r.t. the following "one-sided" relation (which combines multiplicative and additive terms).

$$p_1 \overset{m}{\sim}_{(\epsilon,\delta)} p_2 \iff p_2 \leq e^{\epsilon} \cdot p_1 + \delta.$$

▶ **Definition 6** (Multiplicative PRGs [28]). [4] *A function $G : \{0,1\}^r \to \{0,1\}^m$ is an (ϵ,δ)-multiplicative PRG for a class $\mathcal{C}$ if it is a PRG for $\mathcal{C}$ w.r.t. $\overset{m}{\sim}_{(\epsilon,\delta)}$.*

Note that even if we take $\epsilon = 1$, then for every $\delta > 0$, we have that a $(1,\delta)$-multiplicative PRG $G : \{0,1\}^r \to \{0,1\}^m$ "preserves small probabilities", because

$$\Pr[C(U_m) = 1] \leq \delta \;\Rightarrow\; \Pr[C(G(U_r)) = 1] \leq e^1 \cdot \Pr[C(U_m) = 1] + \delta = O(\delta).$$

Recent work [28, 8] shows that multiplicative PRGs (with this definition) are useful when constructing extractors for samplable distributions. For the applications considered in this paper, it will always suffice to take $\epsilon = 1$, and very small δ, and so, we advise the reader to focus on the parameter δ.

1.4 An Optimal Multiplicative PRG for Nondeterministic Circuits

The main result of this paper is a new construction of multiplicative PRGs for nondeterministic circuits.

▶ **Theorem 7** (Optimal multiplicative PRGs for nondeterministic circuits). *If* E *is hard for exponential size nondeterministic circuits, then for every sufficiently large s, and every $\frac{1}{2^s} \leq \delta \leq \frac{1}{s}$, there is an $(\frac{1}{s},\delta)$-multiplicative PRG,*

$$G : \{0,1\}^{r=O(\log \frac{1}{\delta})} \to \{0,1\}^s$$

for size s nondeterministic circuits. Furthermore, G is computable in time poly(s).

---

[4] We remark that [2] use a different relation, more specifically the "double sided" relation:

$$p_1 \overset{md}{\sim}_{(\epsilon,\delta)} p_2 \iff p_1 \overset{m}{\sim}_{(\epsilon,\delta)} p_2 \text{ and } p_2 \overset{m}{\sim}_{(\epsilon,\delta)} p_1.$$

However, as we will see, in applications we typically only need the one sided version of $\overset{m}{\sim}_{(\epsilon,\delta)}$.

1.4.1 Optimality of the multiplicative PRG

Our multiplicative PRG is optimal in the following features:

- *The hardness assumption is necessary*: The assumption that E is hard for exponential size nondeterministic circuits, easily *follows* from the conclusion of Theorem 7. In fact, it follows for much weaker parameters, see full version for details. Another way to view this is that we are able to get a multiplicative PRG using the same hardness assumption that is necessary and sufficient for standard PRGs in Theorem 4.[5]
- *The seed-length is optimal up to a constant*: For every $\frac{1}{2^s} \leq \delta \leq \frac{1}{s}$, we obtain seed length $r = O(\log \frac{1}{\delta})$. This is obviously best possible (except for the constant hidden in the $O(\cdot)$ notation) as a probability space defined using r random bits cannot have nonzero probabilities that are smaller than 2^{-r}.
- We achieve $\epsilon = \frac{1}{s}$ which by Theorem 5 is best possible for black-box techniques. While we do achieve this, we remark again that for many applications $\epsilon = 1$ suffices.

1.4.2 Comparison to previous multiplicative PRG constructions

Artemenko et al. [2] constructed a multiplicative PRG which is suboptimal precisely in the two features mentioned above. They rely on the stronger assumption that E is hard for exponential size Σ_3-circuits (rather than the necessary assumption that we rely on). Furthermore, the seed length that they use is quadratically larger. More specifically, they obtain $r = O(\log \frac{1}{\delta})^2$ rather than the optimal $r = O(\log \frac{1}{\delta})$.[6]

Shaltiel [28] constructed a multiplicative PRG that has similar parameters to our Theorem 7, with the difference that the output length m is much smaller. More specifically, there exists a constant $\alpha > 0$, such that the output length is $m = (1 + \alpha) \cdot r$. Note that this means that this PRG has small stretch (it cannot even double its seed length) whereas the PRG of Theorem 7 achieves the optimal output length $m = s$ (and a circuit of size s cannot receive more than s bits as input). In particular, the PRG of Theorem 7 achieves superlinear stretch for $\delta = 2^{-o(s)}$, polynomial stretch if $\delta \leq 2^{-s^{0.9}}$, and almost exponential stretch if $\delta = 2^{-\log^{O(1)} s}$.

The proof of Theorem 7 builds on ideas from the two previous constructions [2, 28] (as well as other ideas) as we explain in detail in Section 2.

1.4.3 Black-box impossibility for a version of Theorem 7 for deterministic circuits

Given the similarity between Theorem 3 and Theorem 4 (which are identical except for replacing "deterministic circuits" with "nondeterministic circuits") it is natural to ask whether we can prove a version of Theorem 7 in which nondeterministic circuits are replaced by deterministic circuits. The next theorem shows that (in a similar sense to Theorem 5) "black-box techniques" cannot prove such a result.

---

[5] It is easy to see that an (ϵ, δ)-multiplicative PRG for a class $\mathcal{C}$ that is closed under complement, is also an $O(\epsilon + \delta)$-PRG for $\mathcal{C}$. The class of nondeterministic circuits of size s is not known (and not expected) to be closed under complement. Therefore, the aforementioned implication does not apply. Nevertheless our PRG has the property that for every relevant δ in Theorem 7, the multiplicative PRG that we construct is a $\frac{1}{s}$-PRG for nondeterministic circuits. Note that by Theorem 5 this is the best we can hope for.

[6] The PRG of [2] is only stated for $\delta = 2^{-s^{\Omega(1)}}$, however, it seems to us that it can be extended to any $\frac{1}{2^s} \leq \delta \leq \frac{1}{s}$.

▶ **Theorem 8** ((Informal) Black-box impossibility for a deterministic version of Theorem 7)**.** *"Black box techniques" cannot give a version of Theorem 7 in which* $\epsilon = 1$*,* $\delta = s^{-\omega(1)}$*, and all occurrences of "nondeterministic circuits" are replaced by "deterministic circuits". Furthermore, this holds even if the seed length* r *is increased to* $r = \Omega(s)$*.*

This shows that if one wants a multiplicative PRG for deterministic circuits of size s, with $\delta = s^{-\omega(1)}$, at the moment, we only know how to give such a result as a consequence of Theorem 7, and require the assumption that E is hard for exponential size nondeterministic circuits.

The proof of Theorem 8 follows by carefully adapting the black-box impossibility result of Grinberg, Shaltiel and Viola [20], and is deferred to the full version.

1.5 Randomness Reduction in Explicit Constructions for coNP/poly Properties

Explicit construction problems have the following form: We are given a property (namely a language $\mathcal{P} \in \{0,1\}^*$) and a number m, and want to efficiently produce a string $z \in \mathcal{P}$ of length m. For many famous properties $\mathcal{P}$ it is known that $\Pr[U_m \in \mathcal{P}] \geq 1 - \mu(m)$, for an exponentially small $\mu(m)$. In many cases (like producing a rigid matrix, or a generator matrix for a linear code that meets the Gilbert-Varshamov bound) it is also known that $\mathcal{P} \in \mathsf{coNP}$.

This holds whenever the probabilistic argument showing that $\Pr[U_m \in \mathcal{P}] \geq 1 - \mu(m)$ works as follows: It uses a union bound over exponentially many "bad events" B_i, where for each individual "bad event" B_i, there is a deterministic (or even nondeterministic) circuit $C_i(z)$ of size $s = \mathsf{poly}(m)$, that checks whether z is in the "bad event" B_i. Note that in this case, the union of bad events is in NP, as checking whether a given z is in the union of "bad events", amounts to checking whether there *exists* an i such that $C_i(z)$ accepts. Consequently, $\mathcal{P}$ (which is the complement of the union of bad events) is in coNP.[7]

We are interested in designing a randomized polynomial time algorithm $\mathsf{Construct}$ that on input 1^m, produces a string of length m that has the required property w.h.p. Our focus is the tradeoff between randomness complexity (the number of random bits used) and the error (the probability that the obtained string is not in $\mathcal{P}$). Multiplicative PRGs $G : \{0,1\}^r \to \{0,1\}^m$ for nondeterministic circuits of size $\mathsf{poly}(m)$ immediately apply for this problem. More specifically, by defining $\mathsf{Construct}(1^m) = G(U_r)$ and using Theorem 7 we obtain the following result.

▶ **Theorem 9** (Randomized explicit construction with error δ, using $r = O(\log \frac{1}{\delta})$ random bits)**.** *Let* $\mathcal{P}$ *be a language in* $\mathsf{coNP/poly}$ *such that for every sufficiently large* m*,*

$$\Pr[U_m \in \mathcal{P}] \geq 1 - \delta(m),$$

for $0 < \delta(m) \leq \frac{1}{m}$*. If* E *is hard for exponential size nondeterministic circuits, then there exists a randomized polynomial time algorithm* $\mathsf{Construct}$ *such that for every sufficiently large* m*,* $\mathsf{Construct}(1^m)$ *produces a string of length* m*, using* $r = O(\log \frac{1}{\delta(m)})$ *random bits, and we have that:*

$$\Pr[\mathsf{Construct}(1^m) \in \mathcal{P}] \geq 1 - 3 \cdot \delta(m).$$

[7] To demonstrate this point, consider the problem of explicitly producing an m bit string z that encodes a generator matrix A_z for a binary code that meets the Gilbert-Varshamov bound. Here, for every "message" i, the "bad event" B_i checks whether the string $A_z \cdot i$ has low Hamming weight.

Theorem 9 gives the "correct" tradeoff between the amount of random bits r, and the error δ. This improves upon the previous work of Artemenko et al. [2] (that relied on the multiplicative PRG of that paper) and used a stronger hardness assumption (against Σ_4-circuits) and larger randomness of $r = O(\log \frac{1}{\delta(m)})^2$ rather than $r = O(\log \frac{1}{\delta(m)})$.

1.5.1 Reducing communication when agreeing on a string with a certain coNP/poly property

Let $\mathcal{P}$ be a language as in Theorem 9. Imagine a "cryptographic scenario" where two parties Alice and Bob want to agree on some string $z \in \mathcal{P}$ of a specified length m (for some later use). The obvious approach is for Alice to choose a random $z \leftarrow U_m$ and send it to Bob. In this approach, the communication used is m.

Assuming the hardness assumption, and using Theorem 9, Alice can choose a random $x \in U_r$ for $r = O(\log \frac{1}{\delta(m)})$, send it to Bob, and both players can apply Construct on their own. This reduces the communication from m to r. Typically in cryptographic scenarios one expects error that is negligible in m, and we can take any function $\delta = m^{-\omega(1)}$ and reduce the communication complexity from m to say $r = \log(\frac{1}{\delta(m)})$ which is only slightly larger than $\log m$. Furthermore, note that $\delta = m^{-\omega(1)}$ is precisely where standard PRGs are inapplicable, whereas multiplicative PRGs obtain the "correct" tradeoff.[8]

1.6 Multiplicative PRGs For Nonboolean Circuits

Dubrov and Ishai [16] suggested a natural generalization of PRGs, which extends the standard definition (in which the circuit C outputs one bit) to a setting where C outputs ℓ bits. Thinking ahead, in the definition below, we consider PRGs w.r.t. a relation, and the notion of [16] is obtained for the relation $\overset{ad}{\sim}_\epsilon$.

▶ **Definition 10** (nb-PRGs w.r.t. a relation). *A function $G : \{0,1\}^r \to \{0,1\}^m$ is a ℓ-nb-PRG for a class $\mathcal{C}$ w.r.t. a relation $\sim$ on $[0,1]$, if for every function $C : \{0,1\}^m \to \{0,1\}^\ell$ in the class $\mathcal{C}$, and for every (arbitrary) function $D : \{0,1\}^\ell \to \{0,1\}$,*

$$\Pr[D(C(U_m)) = 1] \sim \Pr[D(C(G(U_r))) = 1].$$

1.6.1 Motivation for nb-PRGs

It is easy to see that for the relation $\overset{ad}{\sim}_\epsilon$ and $\ell = 1$, this definition coincides with the standard definition. For sufficiently large ℓ, and $m > \ell$ (a good choice to keep in mind is $m = \ell^{O(1)}$) we may view a function $C : \{0,1\}^m \to \{0,1\}^\ell$ as a "sampling procedure" which uses m random bits to produce a sample from the distribution $P = C(U_m)$. It is natural to ask whether it is possible to produce a sample from a distribution P' that is ϵ-close to P in statistical distance, using fewer random bits.[9]

nb-PRGs (w.r.t. $\overset{ad}{\sim}_\epsilon$) were introduced by Dubrov and Ishai [16] specifically for this application. More specifically, the distribution $P' = G(U_r)$ is generated using only r random bits, and Definition 10 immediately implies that P and P' are ϵ-close in statistical distance.

[8] We remark that recent work of [11] gives a pseudo-deterministic randomized algorithm for constructing strings with properties $\mathcal{P} \in \mathsf{P}$, and this allows two parties to agree on such a string (w.h.p) with no communication.

[9] Note that here, we want P and P' to be statistically indistinguishable and not just computationally indistinguishable. The latter goal is easier to obtain, but in some settings it is crucial for the two distributions to be statistically indistinguishable, so that even computationally unbounded adversaries D cannot distinguish between them.

Consequently, an ℓ-nb-PRG $G : \{0,1\}^r \to \{0,1\}^s$ for (deterministic circuits) of size s (w.r.t $\overset{ad}{\sim}_\epsilon$) can be used to reduce the randomness complexity of every sampling procedure C that runs in time s, and produces a distribution $P = C(U_s)$ on ℓ-bit strings. More specifically, the randomness complexity is reduced from s to r, while producing a distribution P' that is ϵ-close to P in statistical distance.

Note that we cannot hope for $r < \ell$, because a possible procedure $C(z)$ is one that outputs the first ℓ bits of z, and samples the distribution $P = U_\ell$. It is impossible to sample a distribution P' that is close in statistical distance to U_ℓ using fewer than ℓ random bits.

1.6.2 Constructions of nb-PRGs for polynomial size circuits

Applebaum et al. [1] (improving upon [16, 3]) showed that for every sufficiently large s, and every $\ell \geq \log s$ there is an ℓ-nb-PRG $G : \{0,1\}^{r=O(\ell)} \to \{0,1\}^s$ for circuits of size s (w.r.t. $\overset{ad}{\sim}_{\frac{1}{s}}$). This was obtained assuming E is hard for exponential size nondeterministic circuits. It is not known whether this hardness assumption is necessary, and as far as we know it may be possible to replace "nondeterministic circuits" with "deterministic circuits" in the hardness assumption. This result does achieve optimal seed length of $r = O(\ell)$, and error $\epsilon = \frac{1}{s}$, which is best possible in light of Theorem 5, using the observation that every ℓ-nb-PRG is also a 1-nb-PRG which is a PRG.

1.6.3 Multiplicative nb-PRGs for polynomial size circuits

In some cryptographic applications of nb-PRGs presented in [16, 2] (that we survey in the full version) the desired statistical distance ϵ should be "negligible". Motivated by these applications Artemenko et al. [2] observed that these applications only require "preserving small probabilities", and it is sufficient to use "multiplicative nb-PRGs" (that is nb-PRGs w.r.t $\overset{m}{\sim}_{(1,\delta)}$ for $\delta = s^{-\omega(1)}$) for these applications. We will refer to such ℓ-nb-PRGs as "(ℓ, ϵ, δ)-multiplicative nb-PRGs", and define these explicitly below.

▶ **Definition 11** (Multiplicative nb-PRGs). *A function $G : \{0,1\}^r \to \{0,1\}^m$ is an (ℓ, ϵ, δ)-multiplicative nb-PRG for a class $\mathcal{C}$, if for every function $C : \{0,1\}^m \to \{0,1\}^\ell$ in the class $\mathcal{C}$, and for every (arbitrary) function $D : \{0,1\}^\ell \to \{0,1\}$,*

$$\Pr[D(C(G(U_r))) = 1] \leq e^\epsilon \cdot \Pr[D(C(U_m)) = 1] + \delta.$$

In this paper we give an improved construction of multiplicative nb-PRGs.

▶ **Theorem 12** (Improved multiplicative nb-PRGs). *If E is hard for exponential size nondeterministic circuits, then for every sufficiently large s, every $\ell \geq \log s$, and every $\frac{1}{2^s} \leq \delta \leq \frac{1}{s}$ there is an $(\ell, \frac{1}{s}, \delta)$-multiplicative nb-PRG,*

$$G : \{0,1\}^{r=O(\ell+\log\frac{1}{\delta})} \to \{0,1\}^s$$

for size s deterministic circuits. Furthermore, G is computable in time $\mathsf{poly}(s)$.

We remark that Theorem 12 also holds if one replaces "deterministic circuits" by "single valued nondeterministic circuits".

Theorem 12 achieves the optimal seed length of $r = O(\ell + \log\frac{1}{\delta})$, and in particular, the right dependence on both ℓ and δ (up to constants). The previous construction of Artemenko et al. [2] achieved seed length $r = \ell + O(\log(1/\delta))^2$ which has incorrect dependence on δ.

While we do not know whether the hardness assumption in Theorem 12 is necessary, the hardness assumption used is the same hardness assumption that is used for (standard) nb-PRGs. In contrast, the previous construction of Artemenko et al. [2] assumes the much stronger assumption that E is hard for exponential size Σ_6-circuits.

The proof of Theorem 12 appears in the full version and shows that a multiplicative PRG with correct dependence on δ (as we obtain in Theorem 7) immediately implies a multiplicative nb-PRG.

2 Technique

In this section, we give a detailed informal overview of the main ideas that we use. This informal overview is intended to help the reader to understand the later technical sections (which contain full definitions, statements and proofs and do not build on the informal explanation of this section).

The readers can skip to the technical section if they wish.

2.1 An Overview of the Structure of (Standard) PRG Constructions

Let us start by revisiting the high level structure of (standard) PRGs for deterministic circuits, and specifically, the one by [24, 34] that give the PRG that is stated in Theorem 3. In this setting, we are given a parameter s, and aim to construct a δ-PRG $G : \{0,1\}^{O(\log s)} \to \{0,1\}^s$ for (deterministic) circuits of size s, w.r.t. $\overset{ad}{\sim}_\delta$, for $\delta = \frac{1}{s}$, under a hardness assumption. Thinking ahead, we think of δ as a parameter, and will later consider the case where $\delta = s^{-\omega(1)}$.

2.1.1 Low degree extension

When we aim to construct a δ-PRG for size s circuits and $\delta = \frac{1}{s}$ we first choose a large constant c_0 and set $d = \frac{c_0}{\beta}$ where β is the constant from the harnmess assumption. We choose the input length ℓ for the hardness assumption to be $\ell = d \log s = O(\log s)$. Invoking the hardness assumption, we obtain a function $f : \{0,1\}^\ell \to \{0,1\}$ that cannot be computed by circuits of size $2^{\beta \cdot \ell} = s^{c_0}$, but can be computed in time $2^{O(\ell)} = \mathsf{poly}(s^{c_0})$. We then "extend" the function f into its "low degree extension" $\hat{f}$. This is a degree $\mathsf{poly}(s)$ multivariate polynomial $\hat{f} : \mathbb{F}_q^d \to \mathbb{F}_q$, where $q = \mathsf{poly}(\frac{s}{\delta})$, and is an extension of f, in the following sense: For some fixed (and pre-determined) subset $H \subseteq \mathbb{F}_q$ of size s, there is an efficiently computable bijective map $\phi : \{0,1\}^\ell \to H^d$ such that for every $x \in \{0,1\}^\ell$, we have that $\hat{f}(\phi(x)) = f(x)$. In the terminology of error-correcting codes, this amounts to encoding (the truth table of) f using the Reed-Muller code to obtain the codeword that is (the truth table of) $\hat{f}$.

This step can be viewed as (nonboolean) *hardness amplification*: More specifically, Sudan, Trevisan and Vadhan [34] gave a local list-decoding algorithm showing that a circuit $\hat{C}$ of size s such that $\Pr_{\hat{x} \leftarrow \mathbb{F}_q^d}[\hat{C}(\hat{x}) = \hat{f}(\hat{x})] \geq \delta$, can be used to obtain a circuit B of size $\mathsf{poly}(s, q)$ that computes f correctly. For $\delta = \frac{1}{s}$ (which is the setting in Theorem 3) we have that $q = \mathsf{poly}(s)$ and can choose c_0 to be sufficiently large so that $\mathsf{poly}(s, q) \leq s^{c_0}$, contradicting the hardness

of f. Consequently, we can conclude that for every circuit $\hat{C}$ of size s, $\Pr[\hat{C}(X) = \hat{f}(X)] \leq \delta$. This argument fails if $\delta = s^{-\omega(1)}$, as then $q = s^{\omega(1)}$ and the size of the obtained circuit B is $\mathsf{poly}(s, q) = s^{\omega(1)}$ and does not contradict the hardness assumption.[10]

2.1.2 Concatenating with a boolean code (or Goldreich-Levin)

The next step is to transform the nonboolean function $\hat{f} : \mathbb{F}_q^d \to \mathbb{F}_q$ to a boolean function $g : \mathbb{F}_q^d \times \mathbb{F}_q \to \{0, 1\}$. For this purpose it is helpful to assume w.l.o.g. that q is a power of 2 (so that $v \in \mathbb{F}_q$ can be thought of as a $\log q$ bits string). One then defines:

$$g(w, v) = \langle \hat{f}(w), v \rangle,$$

where the inner product is taken over $\mathbb{F}_2$. In coding theoretic terms, this can be viewed as code-concatenation, namely, encoding each symbol of the "codeword" $\hat{f}$ by the Hadamard code, so that (the truth table of) g is the encoding of f by the code which is the concatenation of Reed-Muller and Hadamard.

The overall process of obtaining g from f is often referred to as "boolean hardness amplification" as one can show that the function g has the property that for $\delta = \frac{1}{s}$, it follows that every circuit P of size s,

$$\Pr_{w \leftarrow \mathbb{F}_q^d, v \leftarrow \mathbb{F}_q}[P(w, v) = g(w, v)] \leq \delta,$$

by showing that otherwise, there exists a small circuit $\hat{C}$ which breaks the (nonboolean hardness) of $\hat{f}$.

We stress that unlike the case of nonboolean hardness amplification, there are black-box limitations by Applebaum et al. [1] showing that assuming hardness against nondeterministic circuit (or even Σ_i-circuits for large i) does not help to obtain *boolean* hardness amplification for $\delta = s^{-\omega(1)}$. This is closely related to the limitations mentioned in Theorem 5, and is the essence of why we cannot expect (standard) PRGs with error $\delta = s^{-\omega(1)}$.

2.1.3 PRGs with one bit-stretch

At this point (at least for $\delta = \frac{1}{s}$) one can construct a (standard) PRG with one bit stretch by:

$$G(x) = x, g(x)$$

Note that the seed length of G is the input length of g which is $n = (d+1) \cdot \log q = O(\log q) = O(\log s + \log \frac{1}{\delta})$, which is indeed $O(\log s)$ for $\delta = \frac{1}{s}$. The correctness of this PRG G follows by first showing that for a uniform $x \leftarrow U_n$, if a circuit D of size s distinguishes $(x, g(x))$ from (x, U_1) (w.r.t. $\overset{ad}{\sim}_\delta$) then by a "distinguisher to predictor" argument, there exists a circuit P of roughly the same size of D, such that $\Pr[P(x) = g(x)] \geq \frac{1}{2} + \delta$, which is a contradiction to the boolean hardness amplification.

[10] We remark that Trevisan and Vadhan [35] showed how to "speed up the local decoding using nondeterminism" and obtain a *nondeterministic* C of size $\mathsf{poly}(s, \log q)$. This allows taking $\delta = s^{-\omega(1)}$ (in fact even exponentially small in s) and still obtain nonboolean hardness amplification, under a hardness assumption against nondeterministic circuits. While this result will not be useful for us directly as stated, we will later use a similar approach to "speed up" a size $\mathsf{poly}(s, q)$ deterministic circuit, and replace it by a size $\mathsf{poly}(s, \log q)$ nondeterministic circuit.

2.1.4 PRGs with large stretch

At this point, the original proof of Theorem 3 uses the Nisan-Wigderson generator [27] and "combinatorial designs" to take a seed x of length $r = O(\frac{n^2}{\log s})$ and stretch it to s strings $x_1, \ldots, x_s \in \{0,1\}^n$ such that the construction $G(x) = g(x_1), \ldots, g(x_s)$ is a PRG. Note that for $\delta = \frac{1}{s}$ this PRG indeed has seed length $r = O(\log s)$

2.2 Multiplicative PRG: The Construction

We now aim to prove Theorem 7 and construct a $(\frac{1}{s}, \delta)$-multiplicative PRG $G : \{0,1\}^{O(\log \frac{1}{\delta})} \to \{0,1\}^s$ for size s nondeterministic circuits, under the assumption that E is hard for exponential size nondeterministic circuits. The main thing to remember is that now δ can be much smaller than $\frac{1}{s}$, and a good choice to keep in mind is $\delta = 2^{-\sqrt{s}}$. Our plan is to follow the steps in Section 2.1. More specifically, given s and $\frac{1}{2^s} \leq \delta \leq \frac{1}{s}$, we set $q = \mathsf{poly}(s/\delta) = \mathsf{poly}(1/\delta)$, exactly as done earlier. However, in contrast to the standard setting in which $\delta = \frac{1}{s}$ and $q = \mathsf{poly}(s)$, we will be dealing with $q = s^{\omega(1)}$ which may be *exponential* in s.

We proceed with the first two steps described in the previous section (low-degree extension and Goldreich-Levin) for this modified choice of δ and q and obtain the function $g : \mathbb{F}_q^d \times \mathbb{F}_q \to \{0,1\}$, defined by $g(w,v) = \langle \hat{f}(w), v \rangle$ over $\mathbb{F}_2$. Recall that the input length of g (in bits) is $n = O(\log s + \log \frac{1}{\delta}) = O(\log \frac{1}{\delta})$.

2.2.1 "Standard approach" fails even for one bit stretch

Note that unlike the previous section, we do not expect to show that $G(x) = (x, g(x))$ is a PRG w.r.t. $\overset{a}{\sim}_\delta$, as by Theorem 5, black-box techniques (like we use here) cannot give such a result. Nevertheless, it was recently shown by Shaltiel [28] that G is a PRG w.r.t. $\overset{m}{\sim}_{(\frac{1}{s},\delta)}$. Our goal is to construct a multiplicative PRG with much larger stretch, that outputs s bits.

2.2.2 Using the Shaltiel-Umans PRG construction

Extending the number of random bits using the Nisan-Wigderson generator [27] (as done in the previous section) gives a seed length that does not have the correct dependence on δ. This is because, as explained in the previous section, using the Nisan-Wigderson generator (and combinatorial designs) one can at best get a seed length of $r = O(\frac{n^2}{\log s}) = \frac{O(\log \frac{1}{\delta})^2}{\log s} \approx (\log \frac{1}{\delta})^2$, for δ that is exponentially small in s. This limitation in the seed length of the Nisan-Wigderson generator is a focus of some earlier work (see [23, 30] for a discussion) and Shaltiel and Umans [30] gave a different PRG construction (in the standard setting) that avoids this type of loss.

Our plan is to use the approach of [30] and adjust it to the multiplicative setting. The Shaltiel-Umans PRG [30] relies on a $d \times d$ matrix A over $\mathbb{F}_q$ with some specific properties that we will soon explain. For every $0 \leq j \leq d-1$, they define $G_j : \mathbb{F}_q^d \times \mathbb{F}_q \to \{0,1\}^s$ as follows:

$$G_j(w,v) = g\left(\left(A^{s^j}\right)^0 \cdot w, v\right), \ldots, g\left(\left(A^{s^j}\right)^{s-1} \cdot w, v\right).$$

Note that the seed length of each G_j is $(d+1)\log q = O(\log \frac{1}{\delta})$. The final PRG $G : \left(\{0,1\}^{(d+1)\log q}\right)^d \to \{0,1\}^s$ is given by:

$$G(x_0, \ldots, x_{d-1}) = \bigoplus_{j=0}^{d-1} G_j(x_j)$$

and note that its seed length is $O(d \cdot \log \frac{1}{\delta}) = O(\log \frac{1}{\delta})$ as required.

2.2.3 Adjusting for an exponentially large field size

We will use this construction. A difficulty is that the construction and analysis of [30] are tailored to a parameter regime in which $q = \mathsf{poly}(s)$. This is because several components in the construction and analysis run in time $\mathsf{poly}(q)$. While, this is acceptable in the original construction where $q = \mathsf{poly}(s)$, it is a deal breaker in our case in which q may be exponential in s.[11]

Jumping ahead, we mention that a lot of the modifications that we make in the construction and analysis of [30] are required to reduce the running time of certain procedures from $\mathsf{poly}(q)$ to $\mathsf{poly}(s)$. The first such example is in the choice of the matrix A.

2.2.4 A modified property for the matrix A

As the matrix A is used by the PRG construction, that is required to run in time $\mathsf{poly}(s)$, we need to use a matrix A that can be found in time $\mathsf{poly}(s)$. The PRG construction of [30] relies on a matrix that we only know to find in time $\mathsf{poly}(q)$. As we cannot afford this, we will use a matrix A with a modified property, that can be found in time $\mathsf{poly}(s)$. We now elaborate on this issue.

The property of the $d \times d$ matrix A over $\mathbb{F}_q$ that is used in the analysis of [30] is that it is regular, and for every $w \in \mathbb{F}_q^d$ such that $w \neq 0$,

$$\left\{A^i \cdot w : 1 \leq i \leq q^d - 1\right\} = \mathbb{F}_q^d \setminus \{0\}.$$

In addition, the PRG construction of [30] also relies on the fact that (by appropriately choosing the relationship between s and q) the matrix $B = A^{\frac{q^d-1}{s^d-1}}$ satisfies that for every $w \in H^d$ such that $w \neq 0$, $\left\{B^i \cdot w : 1 \leq i \leq s^d - 1\right\} = H^d \setminus \{0\}$. We will refer to this property as the "additional property".

In [30] it was shown that a matrix A with the original property can be found in time $\mathsf{poly}(q^d) = \mathsf{poly}(q)$, but we do not know how to find it in time $\mathsf{poly}(s)$. Because of this reason (as well as additional reasons that we will explain later on) we will use a matrix A that only has the "additional property". Namely, in our construction we will use a matrix A that satisfies that A is regular, and for every $w \in H^d$ such that $w \neq 0$,

$$\left\{A^i \cdot w : 1 \leq i \leq s^d - 1\right\} = H^d \setminus \{0\}.$$

Recall that H is of size s, and using similar arguments to the ones used in [30], it follows that a matrix A that satisfies only the modified property can be produced in time $\mathsf{poly}(s^d, \log q) = \mathsf{poly}(s)$. The precise statement appears in the full version. With this choice, computing G indeed takes time $\mathsf{poly}(s)$ as required. Jumping ahead, we will need to modify the proof of [30] to work with this modified property of A.

[11] We remark that Umans [36] gave a modified PRG construction which uses a more complicated low degree extension, but avoids the need to xor the outputs of d "candidate PRGs". This yields reduced seed length, as one avoids the need to choose d independent seeds. Note however that in our setting, $d = c_0/\beta$ is a constant, and so this saving is insignificant. In any case, it seems to us that extending the approach of [36] to the multiplicative setting, is more difficult than extending the approach of [30]. Loosely speaking, this is because there are more technical issues to handle when adjusting the modified low degree extension of Umans [36] to exponentially large field size. It seems to us that some of the ideas used in this paper to adjust for exponentially large can be extended to the framework of Umans [36], but we did not pursue this direction, as at the moment, there is no clear motivation to do so.

2.2.5 How to find a matrix A with the modified property in time $\mathsf{poly}(s)$

For completeness we briefly survey the algorithm of [30] for producing a matrix A. In [30], the matrix A with the "original property" is obtained by identifying the vector space $\mathbb{F}_q^d$ with the extension field $\mathbb{F}_{q^d}$ (in the natural way). The multiplicative group of this field is cyclic, and a generator α for this group can be found by exhaustive search in time $\mathsf{poly}(q^d)$. The function $x \to \alpha \cdot x$ over $\mathbb{F}_q^d$ is an invertible $\mathbb{F}_q$-linear map, and therefore, can be represented by an invertible $d \times d$ matrix A over $\mathbb{F}_q$. This indeed gives that A is regular, and $\left\{A^i \cdot v : 1 \leq i \leq q^d - 1\right\} = \mathbb{F}_q^d \setminus \{0\}$.

Shaltiel and Umans [30] were able to argue that $B = A^{\frac{q^d-1}{s^d-1}}$ satisfies the "additional property" by carefully selecting the parameters s and q, to guarantee that there is a subfield of size s^d in $\mathbb{F}_{q^d}$ that coincides with (the set) H^d. By the same argument, one can use brute force search to find a generator of the multiplicative group of the *subfield*, rather than the *big field*. As this subfield is only of size s^d (rather than q^d), the time of the brute force search for the generator can be reduced from $\mathsf{poly}(q^d)$ to $\mathsf{poly}(s^d, \log q) = \mathsf{poly}(s)$.

2.3 Multiplicative PRG: The Analysis

We will now outline the proof of Theorem 7. We assume for the purpose of contradiction that G is not a multiplicative PRG with the required parameters, and therefore, there exists a size s nondeterministic circuit $D : \{0,1\}^s \to \{0,1\}$, that "distinguishes" a pseudorandom output from a uniform output, meaning that $\Pr[D(U_s) = 1] \overset{m}{\not\sim}_{(\frac{1}{s},\delta)} \Pr[D(G(U_r)) = 1]$.

Our goal is to contradict the hardness assumption by constructing a nondeterministic circuit B of size s^{c_0} for f. On a high level, we will try to imitate the analysis of Shaltiel and Umans [30] (that applies in the standard case) by also using ideas from the approach of Shaltiel [28] (that shows that $G(x) = x, g(x)$ is a multiplicative PRG). We will have to deal with the fact that q is no longer polynomial in s, and that we use a matrix A with the modified property.

The first step in the [30] analysis is to argue that as G is the xor of d "candidates" $G_0, \ldots, G_{d-1}$, a nondeterministic "distinguisher" D for G, yields d nondeterministic distinguishers $D_0, \ldots, D_{d-1}$ for $G_0, \ldots, G_{d-1}$. This step can easily be adjusted to our multiplicative setting (see full version for details).

The second step in [30] (as in many PRG constructions) is to use a hybrid argument [18] to argue that for every $0 \leq j \leq d-1$, the circuit D_j distinguishes between two distributions which only differ in the last bit. This hybrid argument can be easily adapted to the multiplicative setting (see full version for details). To make the notations simpler, we concentrate only on $j = 0$, and then the first $s-1$ bits of $G_0(w, v)$ are given by:

$$\mathsf{Prv}(w, v) = \langle \hat{f}(A^0 \cdot w), v\rangle, \ldots, \langle \hat{f}(A^{s-2} \cdot w), v\rangle,$$

whereas the last output bit of $G_0(w, v)$ is $\mathsf{Nxt}(w, v) = \langle \hat{f}(A^{s-1} \cdot w), v\rangle$.

For a uniformly chosen seed (w, v) for G_0, this "multiplicative hybrid argument" gives that D_0 distinguishes $(\mathsf{Prv}(w, v), U_1)$ from $(\mathsf{Prv}(w, v), \mathsf{Nxt}(w, v))$ w.r.t. $\overset{m}{\sim}_{(\frac{1}{s},\delta)}$. More formally, we have that:

$$\Pr[D_0\left(\mathsf{Prv}(w, v), U_1\right) = 1] \overset{m}{\not\sim}_{(\frac{1}{s},\delta)} \Pr[D_0\left(\mathsf{Prv}(w, v), \mathsf{Nxt}(w, v)\right) = 1].^{12}$$

[12] Here, we are slightly cheating as (just like in the case of the standard hybrid argument) there are small quantitative losses the multiplicative hybrid argument incurs, and both $\frac{1}{s}$ and δ should be divided by s (See full version for details). However, this quantitative loss is insignificant for our purposes, and we ignore it in this high level overview.

2.3.1 The multiplicative setting does not have a "distinguisher to predictor" argument

So far, the analysis we sketched closely follows the proof of [30] by finding "multiplicative analogs" for the steps in the original proof. Now, however, we arrive at a step for which there does not seem to be a multiplicative analog. In the standard setting, one can use a "distinguisher to predictor" argument to transform the distinguisher D_0 into a predictor $P_0(\mathsf{Prv}(w,v))$ which predicts $\mathsf{Nxt}(w,v)$ correctly with probability roughly $\frac{1}{2}+\delta$.

In our setting, $\delta = s^{-\omega(1)}$ and such a predictor is not useful. Loosely speaking, had we been able to use such a predictor to contradict the hardness of f, we would have broken the limitations of Theorem 5. Instead, we will need to proceed without using a "distinguisher to predictor" step, and instead we will try to imitate the next steps of the analysis of [30] using a distinguisher D_0 rather than a predictor. Loosely speaking, following [28], we will try to circumvent this difficulty using *nondeterminism*. Before we explain how to do this, let us briefly survey how the original proof of [30] proceeds after obtaining a predictor P_0.

2.3.2 How [30] use the predictor

Note that for every w, $(\langle \hat{f}(w), v\rangle)_{v\in\mathbb{F}_q}$ is the Hadamard encoding of $\hat{f}(w)$. Loosely speaking, by using list decoding techniques, [30] obtain a (nonboolean) predictor P_0' such that:

$$P_0'\left(\hat{f}(A^0\cdot w),\ldots,\hat{f}(A^{s-2}\cdot w)\right) = \hat{f}(A^{s-1}\cdot w)$$

with probability roughly δ. Loosely speaking, Shaltiel and Umans [30] show how to "error-correct" P_0' to an errorless predictor P_0^* that succeeds with probability one. We will explain some of this argument later.

Once the nonboolean predictor is errorless, one can obtain a circuit B that computes f as follows: The circuit B will be hardwired with some $w_0 \in \mathbb{F}_q^d$, and the evaluations $\hat{f}(A^0\cdot w_0),\ldots,\hat{f}(A^{s-2}\cdot w_0)$. When given $x\in\{0,1\}^\ell$, the circuit B will compute a number i^* such that $A^{i^*}\cdot w_0 = \phi(x)$ (where ϕ is the map $\phi:\{0,1\}^\ell \to H^d$ of the low degree extension) and then we know that $f(x) = \hat{f}(\phi(x)) = \hat{f}(A^{i^*}\cdot w_0)$.

Note that B can use P_0^* to predict the "next element" $\hat{f}(A^{s-1}\cdot w_0)$ from "previous elements" $\hat{f}(A^0\cdot w_0),\ldots,\hat{f}(A^{s-2}\cdot w_0)$, and then continue to predict $\hat{f}(A^s\cdot w_0)$ from $\hat{f}(A^1\cdot w_0),\ldots,\hat{f}(A^{s-1}\cdot w_0)$ that are now available to it. By using this process iteratively, one can reach i^* and compute $\hat{f}(A^{i^*}\cdot w_0) = f(x)$.

It should be noted that as described above, this iterative process takes too much time. This is because i^* could be very large. Using a matrix A with the original property, it could be that $i^* = q^d - 1$ and this could take q^d steps. With the modified property the number of steps can be reduced to s^d (assuming we can arrange that $w_0 \in H^d$ so that the starting point is in H^d). We will later explain how to modify the approach of [30] so that we can arrange that $w_0 \in H^d$. The reduced number of steps follows because by the modified property of A we have that $i^* \leq s^d$.

However, even with this reduction in the number of steps, this takes too much time, as $s^d = 2^\ell$, and it is trivial to obtain a size 2^ℓ circuit B for a function f on ℓ bits. This issue is the reason that [30] use many "generators" $G_0,\ldots,G_{d-1}$ in their construction. Loosely speaking, when using t generators, [30] shows (by a complicated argument that we will not explain in this high level overview) how to reduce the number of steps from i^* to $(i^*)^{1/t}$. This gives that in our setup, using a matrix with modified property, so that $i^* \leq s^d$, by using $t = d$ candidates, we can reduce the number of steps from s^d to s, and the obtained circuit B will end up being of size s^{c_0} that indeed contradicts the hardness of f.

2.3.3 Learning the next curve

As we explained above, the proof of [30] uses error-correcting techniques to error-correct a "noisy predictor" into an "errorless predictor". We now give an overview of this argument (and later explain how to imitate it using a distinguisher instead of a predictor). The main technical step in this process considers the following probabilistic setting: Let r be a large constant, and consider a uniformly chosen degree r polynomial $C : \mathbb{F}_q \to \mathbb{F}_q^d$ (which we will refer to as a curve). Loosely speaking, the approach of [30] is to design an errorless predictor P_0^* which predicts all the evaluations of the "next curve"

$$\left(\hat{f}(A^{s-1} \cdot C(t))\right)_{t \in \mathbb{F}_q}$$

from evaluations on previous curves, namely from:

$$\left(\hat{f}(A^0 \cdot C(t)), \ldots, \hat{f}(A^{s-2} \cdot C(t))\right)_{t \in \mathbb{F}_q}.$$

Note that the evaluations on the next curve, are evaluations of the univariate polynomial

$$\hat{p}(t) = \hat{f}(A^{s-1} \cdot C(t)).$$

This polynomial is a nonzero polynomial with low degree (as $\hat{f}, C$ are low degree, and A is a regular matrix). For every $t \in \mathbb{F}_q$, one can use the "noisy predictor" P_0' for $w = C(t)$, using the previous evaluations, and obtain a prediction for $\hat{p}(t)$. For every $t \in \mathbb{F}_q$, $C(t)$ is uniform over $\mathbb{F}_q^d$, and this means that we expect a δ-fraction of the q predictions to be correct. At this point, [30] use Sudan's list-decoding algorithm for the Reed-Solomon code [33] to obtain a list of $L = \mathsf{poly}(1/\delta)$ polynomials such that one of them is the correct polynomial $\hat{p}$. For the sake of explaining the argument, let us cheat and assume that $L = 1$, and one can "uniquely decode" to the correct polynomial.[13]

2.3.4 Arranging that a random curve intersects H^d

Recall that earlier, we promised to show that "errorless prediction" can start from a point w_0 that is in H^d rather than $\mathbb{F}_q^d$. This does not follow from the argument of [30]. Loosely speaking, this is because the random curve C is unlikely to intersect the set H^d, and so it may be the case that no "starting point" is in H^d.

One of the modifications that we introduce in this paper (which already applies in the original setting of [30]) is a modified probability space for the analysis of [30]. In this modified probability space, we insist that the random curve C intersects the set H^d. More specifically, rather than choosing the curve C uniformly at random, we choose it *conditioned* on the event that $\{C(0) = w_0\}$ for some $w_0 \in H^d$. Fortunately, it is possible to carry out the analysis of [30] with this modification. Loosely speaking, this is because even in the modified probability space, the random variables $(C(t))_{t \leftarrow \mathbb{F}_q \setminus \{0\}}$ are distributed like $q-1$ elements that are $(r-1)$-wise independent.

[13] Shaltiel and Umans are able to achieve $L = 1$, by considering a significantly more complicated random experiment in which two curves C^1 and C^2 are chosen at random, so that each one is uniform, but the two curves are "interleaved" (meaning that they are correlated in a specially designed way that allows list-decoding to imply unique decoding). We will not explain this argument in this high level overview. Our full proof repeats this argument (see full version for details) with some changes that we explain below.

2.3.5 Using a distinguisher instead of a predictor

We now return to the multiplicative case. Recall that we stopped after obtaining a distinguisher D_0 such that

$$\Pr[D_0\left(\mathsf{Prv}(w,v),U_1\right)=1] \overset{m}{\not\sim}_{(\frac{1}{s},\delta)} \Pr[D_0\left(\mathsf{Prv}(w,v),\mathsf{Nxt}(w,v)\right)=1].$$

As we explained earlier, in our setting when δ is very small, and we cannot use a "distinguisher to predictor" argument. Instead, we will have to imitate the proof of [30] using the distinguisher D_0.

A key idea is that while we don't have a predictor, we are in a setting where we are allowed to use nondeterminism. Inspired by [6, 28], our high level idea is to "learn the next curve" in two steps: We first use nondeterminism to guess the coefficients of the "correct polynomial" $\hat{p}(t)=\hat{f}(A^{s-1}\cdot C(t))$. We then design an efficient test such that the correct polynomial $\hat{p}$ passes the test, but no other low-degree polynomial does. This means that using nondeterminism, we can reduce the task of "learning the next curve" to the task of "testing the next curve".

Note that in the framework, the verification step of "testing the next curve" can *itself* be *nondeterministic*, as even then, the entire "guess and test" computation can be implemented by a small nondeterministic circuit. This additional freedom of using nondeterminism twice will be crucial in the argument.

More specifically, we will be able to show that there exists a number $\gamma>0$ (which can be exponentially small in s) such that for a random $t,v\leftarrow\mathbb{F}_q$, the correct polynomial $\hat{p}$ satisfies:

$$\Pr[D_0(\mathsf{Prv}(C(t),v),\langle\hat{p}(t),v\rangle)=1]\geq\gamma.$$

On the other hand, for every low degree polynomial $p\neq\hat{p}$,

$$\Pr[D_0(\mathsf{Prv}(C(t),v),\langle p(t),v\rangle)=1]\leq e^{-\frac{1}{s}}\cdot\gamma.$$

Loosely speaking, this "multiplicative distance" between γ and $e^{-\frac{1}{s}}\cdot\gamma$ is inherited from the fact that the distinguisher D_0 distinguishes w.r.t. a multiplicative relation. We will not go into precise details of this proof that builds on ideas from [6, 28] that are adapted to the setting considered in [30]. In fact, in the actual proof, we can only show that the second inequality holds for p which agrees with $\hat{p}$ on r choices of $t\in\mathbb{F}_q$. In this high level overview, we will cheat and ignore this technicality (which introduces significant complications).

In order to test whether a given low degree polynomial p is the correct polynomial $\hat{p}$, we will prepare the size $\mathsf{poly}(s)$ nondeterministic circuit:

$$B(t,v)=D_0(\mathsf{Prv}(C(t),v),\langle p(t),v\rangle).$$

All that is left is to design a test that given a size $\mathsf{poly}(s)$ circuit B, distinguishes the case that B accepts at least a γ-fraction of its inputs, from the case that B accepts at most $e^{-\frac{1}{s}}\cdot\gamma$-fraction of its inputs.

2.3.6 Using the Goldwasser-Sipser "set lower bound" protocol

We now explain how to distinguish the two cases. To gain some intuition, note that we do not have time to go over all inputs to B as their number is larger than q, and is not polynomial in s. It is also not feasible to distinguish the two cases by taking a sample of $\mathsf{poly}(s)$ random inputs to B, because the "additive distance" $\gamma-e^{-\frac{1}{s}}\cdot\gamma$ may be exponentially small in s, and a $\mathsf{poly}(s)$ size sample cannot be used to distinguish.

Fortunately, distinguishing between these two cases is precisely what nondeterministic circuits are good at. Following [28], we use the "set lower bound" of Goldwasser and Sipser [19] for this purpose. The Goldwasser-Sipser "set lower bound" is typically stated as an AM protocol, where the input is a size s deterministic circuit B, and Merlin claims that B accepts a γ-fraction of the inputs (for some given $\gamma > 0$). We can adapt this protocol to our setting by observing that:

- The Goldwasser-Sipser AM protocol works not just for deterministic circuits, but also for nondeterministic circuits. (This essentially follows because AM with a constant number of messages can be collapsed to two messages).
- $\mathsf{AM} \subseteq \mathsf{NP/poly}$, and so, the Goldwasser-Sipser AM protocol can be implemented by a nondeterministic circuit of size $\mathsf{poly}(s)$.
- The error of the Goldwasser-Sipser protocol is *multiplicative*. More specifically, even when γ is exponentially small, when given a size $\mathsf{poly}(s)$ circuit B, the Goldwasser-Sipser protocol can distinguish the case that B accepts at least a γ-fraction of the inputs, from the case that B accepts at most a $e^{-\frac{1}{s}} \cdot \gamma$-fraction of its inputs.[14]

Note that in the argument above we use nondeterminism three times: We use nondeterminism as D_0 is a nondeterministic circuit. Nevertheless, even if we were constructing a multiplicative PRG for deterministic circuits and D_0 is deterministic, we need nondeterminism to guess the correct polynomial $\hat{p}$, and then we need additional nondeterminism to run the Goldwasser-Sipser protocol.

Organization of this paper

Due to space constraints, this version is an extended abstract and does not contain full statements and proof. The reader is referred to [13] for a full version of this paper.

3 Conclusion and Open Problems

In this paper we give an optimal construction of multiplicative PRGs for nondeterministic circuits, as well as multiplicative nb-PRGs. Both are constructed under the hardness assumption that E is hard for exponential size nondeterministic circuits.

A natural open problem is to find more applications of multiplicative PRGs. In addition to the applications mentioned in this paper, recent work [28, 8] utilizes multiplicative PRGs as building blocks in explicit constructions of extractors for samplable distributions, as well as other types of "hard to sample functions". It is natural to ask whether the multiplicative PRGs of this paper can provide improvement in these applications.

Theorem 5 shows that we cannot expect to obtain (standard) ϵ-PRGs with error $\epsilon = s^{-\omega(1)}$ under assumptions of the form: there exists an i, such that E is hard for exponential size Σ_i-circuits. Is there a way to bypass these limitation by introducing another type of plausible assumption? We remark that it is possible to bypass this limitations if one is willing to

[14] Loosely speaking, it is this property of nondeterministic circuits that avoids black-box limitations and enables us to obtain multiplicative PRGs with $\delta = s^{-\omega(1)}$ under a hardness assumption against nondeterministic circuits. Loosely speaking, the black-box impossibility results of [1] follow by showing that size $\mathsf{poly}(s)$ nondeterministic circuits (or even Σ_i-circuits for any large i) cannot distinguish the case where a circuit B of size s accepts $\frac{1}{2} + s^{-\omega(1)}$ of its inputs from the case that it accepts $\frac{1}{2}$ of its inputs. This is essentially why we are not able to use predictors in our proof. However, we can use "multiplicative" distinguishers, as this leads to the need to distinguish between a circuit that accepts at least γ-fraction of inputs, from a circuit that accepts at most $e^{-\frac{1}{s}} \cdot \gamma$ fraction of inputs, and we've just seen that nondeterministic circuits *can* perform this task.

assume the very strong assumption that E is hard for exponential size PSPACE-circuits (as also mentioned in [2]). However, maybe there's a completely different line of plausible assumptions that can bypass such limitations.

A related open problem is that we don't know whether (standard) nb-PRGs for deterministic circuits require a hardness assumption against nondeterministic circuits. This assumption is used by current constructions [1], but unlike other cases mentioned in this paper, there are no black-box impossibility results that prevent such an nb-PRG from being based on the assumption that E is hard for exponential size deterministic circuits.

References

1 B. Applebaum, S. Artemenko, R. Shaltiel, and G. Yang. Incompressible functions, relative-error extractors, and the power of nondeterministic reductions. In *30th Conference on Computational Complexity*, pages 582–600, 2015. doi:10.4230/LIPIcs.CCC.2015.582.

2 S. Artemenko, R. Impagliazzo, V. Kabanets, and R. Shaltiel. Pseudorandomness when the odds are against you. In *31st Conference on Computational Complexity, CCC*, volume 50, pages 9:1–9:35, 2016. doi:10.4230/LIPIcs.CCC.2016.9.

3 S. Artemenko and R. Shaltiel. Pseudorandom generators with optimal seed length for non-boolean poly-size circuits. In *Symposium on Theory of Computing, STOC*, pages 99–108, 2014. doi:10.1145/2591796.2591846.

4 V. Arvind and J. Köbler. New lowness results for ZPP^{NP} and other complexity classes. *J. Comput. Syst. Sci.*, 65(2):257–277, 2002. doi:10.1006/jcss.2002.1835.

5 M. Ball, D. Dachman-Soled, and J. Loss. (nondeterministic) hardness vs. non-malleability. In *Advances in Cryptology - CRYPTO 2022 - 42nd Annual International Cryptology Conference*, volume 13507, pages 148–177, 2022. doi:10.1007/978-3-031-15802-5_6.

6 M. Ball, E. Goldin, D. Dachman-Soled, and S. Mutreja. Extracting randomness from samplable distributions, revisited. In *64th IEEE Annual Symposium on Foundations of Computer Science, FOCS*, pages 1505–1514, 2023. doi:10.1109/FOCS57990.2023.00092.

7 M. Ball, R. Shaltiel, and J. Silbak. Non-malleable codes with optimal rate for poly-size circuits. In *Advances in Cryptology - EUROCRYPT*, volume 14654 of *Lecture Notes in Computer Science*, pages 33–54, 2024. doi:10.1007/978-3-031-58737-5_2.

8 M. Ball, R. Shaltiel, and J. Silbak. Extractor for samplable distributions with low min-entropy. *To appear in STOC*, 2025.

9 B. Barak, S. J. Ong, and S. P. Vadhan. Derandomization in cryptography. *SIAM J. Comput.*, 37(2):380–400, 2007. doi:10.1137/050641958.

10 N. Bitansky and V. Vaikuntanathan. A note on perfect correctness by derandomization. In *Advances in Cryptology - EUROCRYPT 2017 - 36th Annual International Conference on the Theory and Applications of Cryptographic Techniques*, volume 10211, pages 592–606, 2017. doi:10.1007/978-3-319-56614-6_20.

11 L. Chen, Z. Lu, I. C. Oliveira, H. Ren, and R. Santhanam. Polynomial-time pseudodeterministic construction of primes. In *64th IEEE Annual Symposium on Foundations of Computer Science, FOCS*, pages 1261–1270. IEEE, 2023. doi:10.1109/FOCS57990.2023.00074.

12 L. Chen and R. Tell. When arthur has neither random coins nor time to spare: Superfast derandomization of proof systems. *Electron. Colloquium Comput. Complex.*, TR22-057, 2022. URL: https://eccc.weizmann.ac.il/report/2022/057.

13 A. Dermer and R. Shaltiel. Multiplicative pseudorandom generators for nondeterministic circuits. *Electron. Colloquium Comput. Complex.*, TR26, 2026. URL: https://eccc.weizmann.ac.il/report/2026/017.

14 D. Doron, D. Moshkovitz, J. Oh, and D. Zuckerman. Nearly optimal pseudorandomness from hardness. *J. ACM*, 69(6):43:1–43:55, 2022. doi:10.1145/3555307.

15 Andrew Drucker. Nondeterministic direct product reductions and the success probability of SAT solvers. In *54th Annual IEEE Symposium on Foundations of Computer Science, FOCS*, pages 736–745, 2013. doi:10.1109/FOCS.2013.84.

16 B. Dubrov and Y. Ishai. On the randomness complexity of efficient sampling. In *Proceedings of the 38th Annual ACM Symposium on Theory of Computing*, pages 711–720, 2006. doi:10.1145/1132516.1132615.

17 O. Goldreich and A. Wigderson. Derandomization that is rarely wrong from short advice that is typically good. In *APPROX-RANDOM*, pages 209–223, 2002. URL: http://link.springer.de/link/service/series/0558/bibs/2483/24830209.htm.

18 S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270–299, April 1984. Preliminary version appeared in STOC' 82. doi:10.1016/0022-0000(84)90070-9.

19 S. Goldwasser and M. Sipser. Private coins versus public coins in interactive proof systems. In *Proceedings of the 18th Annual ACM Symposium on Theory of Computing*, pages 59–68, 1986. doi:10.1145/12130.12137.

20 A. Grinberg, R. Shaltiel, and E. Viola. Indistinguishability by adaptive procedures with advice, and lower bounds on hardness amplification proofs. In *59th IEEE Annual Symposium on Foundations of Computer Science, FOCS*, pages 956–966. IEEE Computer Society, 2018. doi:10.1109/FOCS.2018.00094.

21 Dan Gutfreund, Ronen Shaltiel, and Amnon Ta-Shma. Uniform hardness versus randomness tradeoffs for arthur-merlin games. *Computational Complexity*, 12(3-4):85–130, 2003. doi:10.1007/s00037-003-0178-7.

22 P. Hubácek, M. Naor, and E. Yogev. The journey from NP to TFNP hardness. In *8th Innovations in Theoretical Computer Science Conference, ITCS*, volume 67, pages 60:1–60:21, 2017. doi:10.4230/LIPIcs.ITCS.2017.60.

23 R. Impagliazzo, R. Shaltiel, and A. Wigderson. Reducing the seed length in the nisan-wigderson generator. *Combinatorica*, 26(6):647–681, 2006. doi:10.1007/s00493-006-0036-8.

24 R. Impagliazzo and A. Wigderson. $P = BPP$ if E requires exponential circuits: Derandomizing the XOR lemma. In *STOC*, pages 220–229, 1997.

25 A. Klivans and D. van Melkebeek. Graph nonisomorphism has subexponential size proofs unless the polynomial-time hierarchy collapses. *SIAM J. Comput.*, 31(5):1501–1526, 2002. doi:10.1137/S0097539700389652.

26 P. Bro Miltersen and N. V. Vinodchandran. Derandomizing arthur-merlin games using hitting sets. *Computational Complexity*, 14(3):256–279, 2005. doi:10.1007/s00037-005-0197-7.

27 N. Nisan and A. Wigderson. Hardness vs. randomness. *JCSS: Journal of Computer and System Sciences*, 49, 1994.

28 R. Shaltiel. Multiplicative extractors for samplable distributions. In *40th Computational Complexity Conference, CCC*, volume 339 of *LIPIcs*, pages 22:1–22:22. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2025. doi:10.4230/LIPIcs.CCC.2025.22.

29 R. Shaltiel and J. Silbak. Explicit codes for poly-size circuits and functions that are hard to sample on low entropy distributions. In *Proceedings of the 56th Annual ACM Symposium on Theory of Computing, STOC*, pages 2028–2038, 2024. doi:10.1145/3618260.3649735.

30 R. Shaltiel and C. Umans. Simple extractors for all min-entropies and a new pseudorandom generator. *J. ACM*, 52(2):172–216, 2005. doi:10.1145/1059513.1059516.

31 R. Shaltiel and C. Umans. Pseudorandomness for approximate counting and sampling. *Computational Complexity*, 15(4):298–341, 2006. doi:10.1007/s00037-007-0218-9.

32 R. Shaltiel and C. Umans. Low-end uniform hardness versus randomness tradeoffs for am. *SIAM J. Comput.*, 39(3):1006–1037, 2009. doi:10.1137/070698348.

33 M. Sudan. Decoding of Reed Solomon codes beyond the error-correction bound. *Journal of Complexity*, 13, 1997.

34 M. Sudan, L. Trevisan, and S. P. Vadhan. Pseudorandom generators without the xor lemma. *J. Comput. Syst. Sci.*, 62(2):236–266, 2001. doi:10.1006/JCSS.2000.1730.

35 L. Trevisan and S. P. Vadhan. Extracting randomness from samplable distributions. In *41st Annual Symposium on Foundations of Computer Science*, pages 32–42, 2000. doi:10.1109/SFCS.2000.892063.

36 C. Umans. Pseudo-random generators for all hardnesses. *Journal of Computer and System Sciences*, 67:419–440, 2003. doi:10.1016/S0022-0000(03)00046-1.