Electronic Colloquium on Computational Complexity, Report No. 17 (2026)

Multiplicative Pseudorandom Generators for Nondeterministic Circuits

Alon Dermer* Ronen Shaltiel†

February 12, 2026

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 [NW94, IW97]. A sequence of works [KvM02, SU05, Uma03, SU06] generalized the result of [NW94, IW97] to nondeterministic circuits. More specifically, they 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. [AASY15] showed that "black-box techniques" cannot achieve such results for $\epsilon = s^{-\omega(1)}$. In order to circumvent this problem, Artemenko et al. [AIKS16] 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. [AIKS16] 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 [AIKS16], 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 [Sha25] that only achieves very small stretch (the output length in [Sha25] is less than twice the seed length).

We also show that black-box techniques cannot give a version of our result where "nondeterministic" is replaced by "deterministic". This justifies the current situation where hardness for nondeterministic circuits is used even if one only wants low error multiplicative PRGs that fool *deterministic* circuits.

Our PRG construction borrows ideas from the recent "low stretch" PRG of Shaltiel [Sha25], and the (standard) PRG construction of Shaltiel and Umans [SU05]. 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 [STV01], the two results are tailored to two very different parameter regimes, and we introduce several new ideas to make the two approaches co-exist.

*University of Haifa, Email: alon.dermer@gmail.com. Alon Dermer was supported by ISF grant 1006/23.

†University of Haifa, Email: ronen@cs.haifa.ac.il. Ronen Shaltiel 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.

ISSN 1433-8092

Contents

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 [AIKS16, Sha25] 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 1.1 (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 1.2 (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. $\overset{ad}{\sim}_\epsilon$.*

1.2 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. [IW97]) 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 Section 3.3 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 [IW97] to imply PRGs for (deterministic) circuits. Subsequent work

[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.

[KvM02, SU05, Uma03, SU06] 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 1.3 ([IW97])**.** *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 1.4 ([SU05])**.** *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.

Informal Theorem 1.5 ([AASY15, GSV18])**.** *"Black box techniques" cannot give a version of Theorem 1.3 or Theorem 1.4 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 Definition 3.1.*
- *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 [AASY15, GSV18] 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.

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 1.5) Artemenko et al. [AIKS16] and Shaltiel [Sha25] 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).

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

[2] The assumption that E is hard for exponential size nondeterministic circuits is standard, and was used in many results [AK02, KvM02, MV05, SU05, BOV07, GW02, GST03, SU06, SU09, Dru13, AASY15, BV17, AIKS16, HNY17, DMOZ22, BDL22, CT22, BGDM23, BSS24, SS24, Sha25]. 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.

Definition 1.6 (Multiplicative PRGs [Sha25][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 [Sha25, BSS25] 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 1.7 (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 $\mathsf{poly}(s)$.

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 1.7. In fact, it follows for much weaker parameters, see Remark 4.4. 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 1.4.[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 1.5 is best possible for black-box techniques. While we do achieve this, we remark again that for many applications $\epsilon = 1$ suffices.

Comparison to previous multiplicative PRG constructions. Artemenko et al. [AIKS16] 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 [Sha25] constructed a multiplicative PRG that has similar parameters to our Theorem 1.7, with the difference that the output length m is much smaller. More specifically, there exists a constant $\alpha > 0$, such

[4] We remark that [AIKS16] 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 $\overset{m}{\sim}_{(\epsilon,\delta)}$.

[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 1.7, the multiplicative PRG that we construct is a $\frac{1}{s}$-PRG for nondeterministic circuits. Note that by Theorem 1.5 this is the best we can hope for.

[6] The PRG of [AIKS16] is only stated for $\delta = 2^{-s^{\Omega(1)}}$, however, it seems to us that it can be extended to any $\frac{1}{2^s} \leq \delta \leq \frac{1}{s}$.

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 1.7 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 1.7 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 1.7 builds on ideas from the two previous constructions [AIKS16, Sha25] (as well as other ideas) as we explain in detail in Section 2.

Black-box impossibility for a version of Theorem 1.7 for deterministic circuits. Given the similarity between Theorem 1.3 and Theorem 1.4 (which are identical except for replacing "deterministic circuits" with "nondeterministic circuits") it is natural to ask whether we can prove a version of Theorem 1.7 in which nondeterministic circuits are replaced by deterministic circuits. The next theorem shows that (in a similar sense to Theorem 1.5) "black-box techniques" cannot prove such a result.

Informal Theorem 1.8 (Black-box impossibility for a deterministic version of Theorem 1.7)**.** *"Black box techniques" cannot give a version of Theorem 1.7 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 1.7, and require the assumption that E is hard for exponential size nondeterministic circuits.

The proof of Theorem 1.8 follows by carefully adapting the black-box impossibility result of Grinberg, Shaltiel and Viola [GSV18], and is deferred to the full version.

1.5 Randomness Reduction in Explicit Constructions for $\mathsf{coNP/poly}$ 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 1.7 we obtain the following result.

Theorem 1.9 (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,*

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

[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.

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:*

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

Theorem 1.9 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. [AIKS16] (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 $r = O(\log \frac{1}{\delta(m)})$.

Reducing communication when agreeing on a string with a certain $\mathsf{coNP/poly}$ property. Let $\mathcal{P}$ be a language as in Theorem 1.9. 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 1.9, 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 $\mathsf{Construct}$ 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 [DI06] 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 [DI06] is obtained for the relation $\overset{ad}{\sim}_\epsilon$.

Definition 1.10 (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 $D : \{0,1\}^\ell \to \{0,1\}$,*

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

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 [DI06] specifically for this application. More specifically, the distribution $P' = G(U_r)$ is generated using only r random bits, and Definition 1.10 immediately implies that P and P' are ϵ-close in statistical distance. 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.

[8] We remark that recent work of [CLO+23] 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.

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.

Constructions of nb-PRGs for polynomial size circuits. Applebaum et al. [AASY15] (improving upon [DI06, AS14]) 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 1.5, using the observation that every ℓ-nb-PRG is also a 1-nb-PRG which is a PRG.

Multiplicative nb-PRGs for polynomial size circuits. In some cryptographic applications of nb-PRGs presented in [DI06, AIKS16] (that we survey in Section 6.2) the desired statistical distance ϵ should be "negligible". Motivated by these applications Artemenko et al. [AIKS16] 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 1.11 (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 1.12 (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 1.12 also holds if one replaces "deterministic circuits" by "single valued nondeterministic circuits". See Remark 6.2.

Theorem 1.12 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. [AIKS16] 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 1.12 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. [AIKS16] assumes the much stronger assumption that E is hard for exponential size Σ_6-circuits.

The proof of Theorem 1.12 appears in Section 6 and shows that a multiplicative PRG with correct dependence on δ (as we obtain in Theorem 1.7) 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 [IW97, STV01] that give the PRG that is stated in Theorem 1.3. 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 $\delta = s^{-\omega(1)}$.

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 [STV01] 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 1.3) 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]

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,

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

[10] We remark that Trevisan and Vadhan [TV00] 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.

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. [AASY15] 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 1.5, and is the essence of why we cannot expect (standard) PRGs with error $\delta = s^{-\omega(1)}$.

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.

PRGs with large stretch. At this point, the original proof of Theorem 1.3 uses the Nisan-Wigderson generator [NW94] 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 1.7 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 $n = O(\log s + \log \frac{1}{\delta}) = O(\log \frac{1}{\delta})$.

"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 1.5, black-box techniques (like we use here) cannot give such a result. Nevertheless, it was recently shown by Shaltiel [Sha25] 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.

Using the Shaltiel-Umans PRG construction. Extending the number of random bits using the Nisan-Wigderson generator [NW94] (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 [ISW06, SU05] for a discussion) and Shaltiel and Umans [SU05] gave a different PRG construction (in the standard setting) that avoids this type of loss.

Our plan is to use the approach of [SU05] and adjust it to the multiplicative setting. The Shaltiel-Umans PRG [SU05] 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.

Adjusting for an exponentially large field size. We will use this construction. A difficulty is that the construction and analysis of [SU05] 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.

Jumping ahead, we mention that a lot of the modifications that we make in the construction and analysis of [SU05] 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.

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 [SU05] 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 [SU05] 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 [SU05] 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 [SU05] 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 [SU05], 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 Lemma 4.1 (see also Remark 4.2). With this choice, computing G indeed takes time $\mathsf{poly}(s)$ as required. Jumping ahead, we will need to modify the proof of [SU05] to work with this modified property of A.

How to find a matrix A with the modified property in time $\mathsf{poly}(s)$. For completeness we briefly survey the algorithm of [SU05] for producing a matrix A. In [SU05], 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 [SU05] 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 1.7. 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 [SU05] (that applies in the standard case) by also using ideas from the approach of Shaltiel [Sha25] (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 [SU05] 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 Lemma 3.5).

The second step in [SU05] (as in many PRG constructions) is to use a hybrid argument [GM84] 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 Lemma 3.4). 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:

$$\Pr[D_0\left(\mathsf{Prv}(w, v), U_1\right) = 1] \overset{m}{\not\sim}_{(\frac{1}{s},\delta)} \Pr[D_0\left(\mathsf{Prv}(w, v), \mathsf{Nxt}(w, v)\right) = 1].^{11}$$

The multiplicative setting does not have a "distinguisher to predictor" argument. So far, the analysis we sketched closely follows the proof of [SU05] 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 1.5. 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 [SU05] using a distinguisher D_0 rather than a predictor. Loosely speaking,

[11] 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 Lemma 3.4 for details). However, this quantitative loss is insignificant for our purposes, and we ignore it in this high level overview.

following [Sha25], 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 [SU05] proceeds after obtaining a predictor P_0.

How [SU05] 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, [SU05] 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 [SU05] 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 [SU05] 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 [SU05] use many "generators" $G_0, \ldots, G_{d-1}$ in their construction. Loosely speaking, when using t generators, [SU05] 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.

Learning the next curve. As we explained above, the proof of [SU05] 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 [SU05] 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, [SU05] use Sudan's list-decoding algorithm for the Reed-Solomon code [Sud97] 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.[12]

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 [SU05]. 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 [SU05]) is a modified probability space for the analysis of [SU05]. In this modified probability space (referred to as the "basic probability space" in Section 4.2) 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 [SU05] 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.

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 [SU05] 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 [BGDM23, Sha25], 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 [BGDM23, Sha25] that are adapted to the setting considered in [SU05]. In fact, in

[12] 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 Section 4.2 for details) with some changes that we explain below.

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.

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 [Sha25], we use the "set lower bound" of Goldwasser and Sipser [GS86] 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).
- $\mathsf{AM} \subseteq \mathsf{NP}/\mathsf{poly}$, 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.[13]

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

In Section 3 we give some preliminaries. In Section 4 we present our main construction and prove Theorem 1.7. In Section 5 we prove Theorem 1.9. In Section 6 we present our results on nonboolean PRGs, and their applications. Finally in Section 7 we conclude and present some open problems.

[13] 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 [AASY15] 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.

3 Preliminaries

3.1 Probabilistic notation

For a distribution D, we use the notation $X \leftarrow D$ to denote the experiment in which X is chosen according to D. For a set A, we use $X \leftarrow A$ to denote the experiment in which X is chosen uniformly from the set A.

The notation $X_1, \ldots, X_t \leftarrow A$ denotes the experiment in which t variables are chosen from A independently and with replacement. The notation $X_1, \ldots, X_t \leftarrow A$ denotes the experiment in which t variables are chosen from A independently and with replacement.

We use the notation $X_1, \ldots, X_t \underset{\text{distinct}}{\leftarrow} A$ to denote the experiment in which t distinct elements are chosen from A independently, but without replacement.

We often also identify a distribution X, with the random variable X chosen from this distributions. For a random variable X and an event A we use $(X|A)$ to denote the distribution which chooses an element according to X, conditioned on A. We use U_n to be the uniform distribution on n bit strings.

3.2 Definition of Circuits of Various Types

We formally define the circuit types that will be used in this paper.

Definition 3.1 (randomized circuits, nondeterministic circuits, oracle circuits and Σ_i-circuits)**.** *A* randomized *circuit C has additional wires that are instantiated with uniform and independent bits.*

A nondeterministic *circuit C has additional "nondeterministic input wires". We say that the circuit C evaluates to 1 on x iff there exists an assignment to the nondeterministic input wires that makes C output 1 on x.*

An oracle circuit $C^{(\cdot)}$ is a circuit which in addition to the standard gates uses an additional gate (which may have large fan in). When instantiated with a specific boolean function A, C^A is the circuit in which the additional gate is A. Given a boolean function $A(x)$, an A-circuit is a circuit that is allowed to use A gates (in addition to the standard gates). An $A_{||}$-circuit is a circuit that makes nonadaptive queries to its oracle A. (Namely, on every path from input to output, there is at most a single A gate).

An NP-circuit is a SAT-circuit (where SAT is the satisfiability function) a Σ_i-circuit is an A-circuit where A is the canonical Σ_i^P-complete language. The size of all circuits is the total number of wires and gates.[14]

3.3 Hardness Assumptions

We will rely on assumptions of the following form, introduced by Impagliazzo and Wigderson [IW97]

Definition 3.2 (E is hard for exponential size circuits)**.** *We say that "E is hard for exponential size circuits of type X" if there exist constants $0 < \beta < B$, and a language L in $\mathsf{E} = \mathsf{DTIME}(2^{B \cdot n})$, such that for every sufficiently large n, the characteristic function of L on inputs of length n is hard for circuits of size $2^{\beta n}$ of type X.*

3.4 Properties of Pseudorandomness w.r.t. Multiplicative Relations

Recall that in Section 1 we defined pseudorandomness w.r.t. an arbitrary relation, as well as several specific relations. For completeness, we repeat the definition of the various relations defined in the Section 1. In the relations below "a" stands for additive, "m" stands for multiplicative, and "d" stands for double-sided.

[14] An alternative approach to define these circuit classes is using the Karp-Lipton notation for Turing machines with advice. For $s \geq n$, a size $s^{\Theta(1)}$ deterministic circuit is equivalent to $\mathsf{DTIME}(s^{\Theta(1)})/s^{\Theta(1)}$, a size $s^{\Theta(1)}$ nondeterministic circuit is equivalent to $\mathsf{NTIME}(s^{\Theta(1)})/s^{\Theta(1)}$, a size $s^{\Theta(1)}$ NP-circuit is equivalent to $\mathsf{DTIME}^{\mathsf{NP}}(s^{\Theta(1)})/s^{\Theta(1)}$, and a size $s^{\Theta(1)}$ Σ_i-circuit is equivalent to $\mathsf{DTIME}^{\Sigma_i^P}(s^{\Theta(1)})/s^{\Theta(1)}$.

Definition 3.3 (Definitions of relations from Section 1)**.** *Given numbers $p_1, p_2, \epsilon, \delta \in [0,1]$, we define the following relations:*

$$\begin{aligned} p_1 \overset{ad}{\sim}_{\epsilon} p_2 &\iff |p_2 - p_1| \leq \epsilon. \\ p_1 \overset{a}{\sim}_{\epsilon} p_2 &\iff p_2 \leq p_1 + \epsilon. \\ p_1 \overset{m}{\sim}_{(\epsilon,\delta)} p_2 &\iff p_2 \leq e^{\epsilon} \cdot p_1 + \delta. \\ p_1 \overset{md}{\sim}_{(\epsilon,\delta)} p_2 &\iff p_1 \overset{m}{\sim}_{(\epsilon,\delta)} p_2 \text{ and } p_2 \overset{m}{\sim}_{(\epsilon,\delta)} p_1. \end{aligned}$$

Note that while some of these relations (e.g. $\overset{m}{\sim}_{(\epsilon,\delta)}$) are interesting for $\epsilon > 1$, in this paper we will always have that $0 \leq \epsilon \leq 1$ so that $1 + \epsilon \leq e^{\epsilon} \leq 1 + 3\epsilon$, and $1 - \epsilon \leq e^{-\epsilon} \leq 1 - \frac{\epsilon}{3}$. We will use these inequalities throughout this paper.

Below, we list several properties of pseudorandomness w.r.t. $\overset{m}{\sim}_{(\epsilon,\delta)}$ that can be viewed as generalizations of analogous properties for "standard pseudorandomness" w.r.t. $\overset{a}{\sim}_{\epsilon}$.

3.4.1 A Multiplicative Hybrid Argument

A useful property of standard pseudorandomness is that it allows using the "hybrid argument" of Goldwasser and Micali [GM84]. Below, we state and prove a generalized version for multiplicative pseudorandomness.

Lemma 3.4 (Multiplicative hybrid argument)**.** *Let $\epsilon < \frac{1}{4}$. Let $Z = (Z_1, \ldots, Z_n)$ be some distribution over $(\{0,1\}^k)^n$, and let $R = (R_1, \ldots, R_n)$ be the uniform distribution over $(\{0,1\}^k)^n$. If Z is not pseudorandom for $D : (\{0,1\}^k)^n \to \{0,1\}$, with respect to $\overset{m}{\sim}_{(\epsilon,\delta)}$, then there exists $i \in [n]$ such that*

$$\Pr[D(Z_1, \ldots, Z_{i-1}, R_i, R_{i+1}, \ldots, R_n) = 1] \overset{m}{\not\sim}_{(\epsilon',\delta')} \Pr[D(Z_1, \ldots, Z_{i-1}, Z_i, R_{i+1}, \ldots, R_n) = 1],$$

where $\epsilon' = \frac{\epsilon}{n}$ and $\delta' = \frac{\delta}{3n}$.

Proof. For every $0 \leq i \leq n$ we consider the hybrid distribution

$$H_i = (Z_1, \ldots, Z_{i-1}, Z_i, R_{i+1}, \ldots, R_n),$$

and define $p_i = \Pr[D(H_i) = 1]$. We have that $p_0 \overset{m}{\not\sim}_{(\epsilon,\delta)} p_n$, which says that $p_n > e^{\epsilon} \cdot p_0 + \delta$. We will show that there exists $i \in [n]$ such that $p_i > e^{\epsilon'} \cdot p_{i-1} + \delta'$. If there does not exist such an i, then for every $i \in [n]$, $p_i \leq e^{\epsilon'} \cdot p_{i-1} + \delta'$, which gives that:

$$p_n \leq e^{\epsilon'} \cdot p_{n-1} + \delta' \leq e^{\epsilon'} \cdot (e^{\epsilon'} \cdot p_{n-2} + \delta') + \delta' \leq \ldots \leq e^{n \cdot \epsilon'} \cdot p_0 + \delta' \cdot \sum_{0 \leq j \leq n-1} e^{j \cdot \epsilon'}.$$

By the formula for geometric sums

$$\sum_{0 \leq j \leq n-1} e^{j \cdot \epsilon'} = \frac{e^{\epsilon' \cdot n} - 1}{e^{\epsilon'} - 1} \leq \frac{1 + 3 \cdot \epsilon' \cdot n - 1}{1 + \epsilon' - 1} = 3n.$$

Overall, we get that

$$p_n \leq e^{n \cdot \epsilon'} \cdot p_0 + 3 \cdot \delta' n = e^{\epsilon} \cdot p_0 + \delta,$$

which is a contradiction. □

3.4.2 A Multiplicative XOR-Generator Lemma

A property of standard pseudorandomness that was used in [SU05] is that a distinguisher for the xor of several candidate generators, yields distinguishers of similar complexity for every one of the candidate generators. Below, we state a generalization of this observation for the case of multiplicative pseudorandomness.

Lemma 3.5. *Let $G_0, \ldots, G_{d-1}$ be functions from ℓ bits to n bits, and let $G : \{0,1\}^{d\ell} \to \{0,1\}^n$ be defined by $G(x_0, \ldots, x_{d-1}) = G_0(x_0) \oplus \ldots \oplus G_{d-1}(x_{d-1})$. If G is not a PRG for nondeterministic circuits of size s with respect to $\overset{m}{\sim}_{(\epsilon,\delta)}$, then there exist nondeterministic circuits $D_0, \ldots, D_{d-1}$ of size $s' = s + n$ such that for every $0 \leq i \leq d-1$, G_i is not a PRG for D_i with respect to $\overset{m}{\sim}_{(\epsilon,\delta)}$.*

Proof. By assumption, there exists a nondeterministic circuit D of size s, such that for $p_1 = \Pr[D(U_n) = 1]$, and $p_2 = \Pr[D(G(U_{d\ell})) = 1]$, we have that $p_1 \overset{m}{\not\sim}_{(\epsilon,\delta)} p_2$, meaning that $p_2 > e^{\epsilon} p_1 + \delta$. By definition:

$$p_2 = \Pr_{X_0 \leftarrow \{0,1\}^\ell, \ldots, X_{d-1} \leftarrow \{0,1\}^\ell}[D(G_0(X_0) \oplus \ldots \oplus G_{d-1}(X_{d-1})) = 1].$$

Fix some $0 \leq i \leq d-1$. By an averaging argument, there exist values $x_0, \ldots, x_{i-1}, x_{i+1}, \ldots, x_{d-1} \in \{0,1\}^\ell$ such that

$$p_2 \leq \Pr_{X_i \leftarrow \{0,1\}^\ell}[D(G_0(x_0) \oplus \ldots G_{i-1}(x_{i-1}) \oplus G_i(X_i) \oplus G_{i+1}(x_{i+1}) \oplus \ldots \oplus G_{d-1}(x_{d-1})) = 1].$$

Let $z = G_0(x_0) \oplus \ldots G_{i-1}(x_{i-1}) \oplus G_{i+1}(x_{i+1}) \oplus \ldots \oplus G_{d-1}(x_{d-1})$, and define $D_i(x) = D(x \oplus z)$. Note that D is a nondeterministic circuit of size $s + n$. We have that $\Pr[D_i(G_i(U_\ell)) = 1] \geq p_2$ and $\Pr[D_i(U_n) = 1] = \Pr[D(U_n) = 1] = p_1$, and therefore G_i is not a PRG for D_i with respect to $\overset{m}{\sim}_{(\epsilon,\delta)}$. □

3.4.3 Averaging Arguments for the Multiplicative Relation

In the standard setup of pseudorandomness (that is w.r.t. $\overset{a}{\sim}_\epsilon$) we have that if a randomized circuit D distinguishes a distribution W_1 for W_2, in the sense that $\Pr[D(W_1) = 1] \overset{a}{\not\sim}_\epsilon \Pr[D(W_2) = 1]$, then there exists a fixing to the random coins of D, such that the non-randomized circuit obtained by employing this fixing also distinguishes the two distributions.

The next simple lemma from [Sha25] states that this property also holds for pseudorandomness w.r.t. $\overset{m}{\sim}_{(\epsilon,\delta)}$.

Lemma 3.6 ([Sha25])**.** *Let W_1, W_2 be two distributions. Let D be a randomized nondeterministic circuit. If*

$$\Pr[D(W_1) = 1] \overset{m}{\not\sim}_{(\epsilon,\delta)} \Pr[D(W_2) = 1],$$

then there exists a fixing to the random coins of D such that the obtained (non-randomized) nondeterministic circuit D' satisfies

$$\Pr[D'(W_1) = 1] \overset{m}{\not\sim}_{(\epsilon,\delta)} \Pr[D'(W_2) = 1].$$

3.5 Seeded Extractors and the Leftover Hash Lemma

We use the following standard definition of seeded extractors.

Definition 3.7 (Seeded extractors)**.** *A function $\mathsf{SExt} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ is a (k,ϵ)-seeded extractor if for every distribution X over $\{0,1\}^n$, with $H_\infty(X) \geq k$, $\mathsf{SExt}(X, U_d)$ is ϵ-close to U_m.*

SExt is a strong (k,ϵ)-seeded extractor if the function $\mathsf{SExt}' : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^{d+m}$ defined by $\mathsf{SExt}'(x,y) = (y, \mathsf{SExt}(x,y))$ is a (k,ϵ)-seeded extractor.

We use the following result known as the "leftover hash lemma" by Impagliazzo, Levin and Luby [ILL89]

Theorem 3.8 (Leftover hash lemma [ILL89])**.** *For every integers $m \leq n$, and $\epsilon > 0$, there is a $(m + 2\log(1/\epsilon), \epsilon)$-strong extractor $\mathsf{SExt} : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^m$. Furthermore, SExt can be computed in time $\mathsf{poly}(n)$.*

We remark that in some sources this lemma is stated with $d = 2n$ rather than $d = n$, but the statement also holds for $d = n$ (as stated above).

3.6 The Low Degree Extension

Many results in complexity theory and derandomization rely on the low-degree extension. Loosely speaking, this is a technique to extend a given function $f : \{0,1\}^\ell \to \{0,1\}$ to a low-degree d-variate polynomial $\hat{f} : \mathbb{F}_q^d \to \mathbb{F}_q$. The standard precise statement is given below.

Lemma 3.9. *Let $f : \{0,1\}^\ell \to \{0,1\}$ be a function and $d \leq h \leq q$ be integers such that $h^d \geq 2^\ell$ and q is a power of 2. Given $H \subseteq \mathbb{F}_q$ of size h, and a one-to-one map $\phi : \{0,1\}^\ell \to H^d$, there is a degree $\hat{h} = h \cdot d$ polynomial $\hat{f} : \mathbb{F}_q^d \to \mathbb{F}_q$ such that for every $x \in \{0,1\}^\ell$, $f(x) = \hat{f}(\phi(x))$. Furthermore, $\hat{f}$ can be computed in time $\mathsf{poly}(2^\ell, \log q)$ given oracle access to f and ϕ.*

Proof. For all $j \in H$ define the function $g_j : \mathbb{F}_q \to \mathbb{F}_q$, such that for all $x \in \mathbb{F}_q$:

$$g_j(x) = \frac{\prod_{\substack{i \in H \\ i \neq j}} (x - i)}{\prod_{\substack{i \in H \\ i \neq j}} (j - i)}$$

Note that g_j is a polynomial of degree $h - 1$, $g_j(j) = 1$ and for all $i \in H$, such that $i \neq j$: $g_j(i) = 0$. Now, define the function $\hat{f} : \mathbb{F}_q^d \to \mathbb{F}_q$, such that for all $x = (x_1, \ldots, x_d) \in \mathbb{F}_q^d$:

$$\hat{f}(x_1, \ldots, x_d) = \sum_{y \in \{0,1\}^\ell} f(y) \cdot g_{\phi(y)_1}(x_1) \cdots g_{\phi(y)_d}(x_d)$$

This is a polynomial of degree at most $h \cdot d$ which satisfies $f(x) = \hat{f}(\phi(x))$ for all $x \in \{0,1\}^\ell$. By definition, $\hat{f}$ can be computed in time $\mathsf{poly}(2^\ell, \log q)$ given oracle access to f and ϕ. □

3.7 Sudan's List-Decoding Algorithm

We will rely on Sudan's celebrated list-decoding algorithm for the Reed-Solomon code [Sud97].

Theorem 3.10 (Sudan's list-decoding algorithm [Sud97])**.** *Let $\mathsf{prs}, \mathsf{agr}, \mathsf{deg}$ be integers. Given prs distinct pairs (x_i, y_i) in field F with $\mathsf{agr} > \sqrt{2 \cdot \mathsf{deg} \cdot \mathsf{prs}}$, there are at most $2\mathsf{prs}/\mathsf{agr}$ polynomials g of degree deg such that $g(x_i) = y_i$ for at least agr pairs. Furthermore, a list of all such polynomials can be computed in time $\mathsf{poly}(\mathsf{prs}, \log |F|)$.*

We remark that in this paper (as in the previous work [TV00, BGDM23]) we will rely only existence of small lists, and do not use the efficiency of list-decoding algorithm.

3.8 The Goldwasser-Sipser AM Protocol and Consequences

A classical result by Goldwasser and Sipser [GS86] shows that there is an AM protocol for showing that the fraction of accepting inputs of a given circuit is above some threshold. The same approach translates immediately to the case where the given circuit is nondeterministic (rather than deterministic). Below is a formal definition.

Definition 3.11 (The nondeterministic large set promise problem)**.** *Given $\lambda > 0$, we define a promise problem* $\mathsf{NondetLarge}_\lambda$ *over pairs (C, γ) where C is a nondeterministic circuit, and $0 \leq \gamma \leq 1$.*

- *The Yes instances are pairs (C, γ) such that C accepts at least a γ-fraction of its inputs.*
- *The No instances are pairs (C, γ) such that C accepts less than a $\gamma \cdot e^{-\lambda}$-fraction of its inputs.*

Note that a circuit C of size s can have at most s input bits. Throughout the paper we will always assume w.l.o.g. that C has s input bits (and may ignore some of them). We also note that because the number of possible inputs to C is at most 2^s, we can always assume that the number of bits needed to represent γ is at most s (which implies that the input to the promise problem is of length that is dominated by the length of the description of C, which is $O(s \log s)$).

Theorem 3.12 (Goldwasser and Sipser [GS86])**.** *For every integer s and $\lambda > 0$, there is a nondeterministic circuit A of size* $\mathsf{poly}(s, \frac{1}{\lambda})$ *which solves the promise problem* $\mathsf{NondetLarge}_\lambda$.

Theorem 3.12 is stated in a somewhat nonstandard way. The more standard formulation discusses deterministic circuits C, and gives an AM protocol that solves the promise problem. However, the same result immediately applies to nondeterministic circuits. This is because in the Goldwasser-Sipser AM protocol, Merlin sends inputs x to C on which $C(x) = 1$, and if C is nondeterministic, whenever Merlin sends an x, he can also supply a witness showing that $C(x) = 1$. This gives an AM-protocol with time $\mathsf{poly}(s, \frac{1}{\lambda})$ for $\mathsf{NondetLarge}_\lambda$, and the result in the theorem follows because one can transform an AM-protocol into a nondeterministic circuit, as in the proof that $\mathsf{AM} \subseteq \mathsf{NP/poly}$.

3.9 An r-wise Independent Tail Inequality

We need the following tail inequality by Bellare and Rompel [BR94].

Theorem 3.13 (r-wise independent tail inequality [BR94])**.** *Let $r > 4$ be an even integer. Suppose $X_1, X_2, \ldots, X_n$ are r-wise independent random variables taking values in $[0, 1]$. Let $X = \sum X_i$, $\mu = \mathbb{E}[X]$ and $A > 0$. Then:*

$$\Pr[|X - \mu| \geq A] \leq 8 \cdot \left(\frac{r\mu + r^2}{A^2} \right)^{r/2}.$$

In particular, if $r \leq n$, setting $A = \epsilon n$, for some $\epsilon > 0$, it follows that:

$$\Pr[|X - \mu| \geq \epsilon n] \leq 8 \cdot \left(\frac{2r}{\epsilon^2 n} \right)^{r/2}.$$

4 A Multiplicative PRG for Nondeterministic Circuits

In this section we prove Theorem 1.7 that yields a multiplicative PRG for nondeterministic circuits. In Section 4.1 we present the construction of the multiplicative PRG. The proof of correctness is presented In Section 4.2.

4.1 The Construction

An important ingredient in the construction is the following lemma, that is a special case of a more general lemma that is proven in [SU05, Lemma 4.18].

Lemma 4.1 (Traversing matrix [SU05])**.** *Let h,q and d be such that: h is a power of 2, q is a power of h, and d and $\log_h q$ are relatively prime, and let $\mathbb{F}_q$ be the field with q elements. There exists an invertible $d \times d$*

matrix A with entries in $\mathbb{F}_q$, and a set $H \subseteq F$ with $|H| = h$, such that A^{h^d-1} is the identity matrix, and for every $v \in H^d$, such that $v \neq 0$,

$$\left\{A^i \cdot v : 1 \leq i < h^d\right\} = H^d \setminus \{0\}.$$

Moreover, A can be found in time $\mathsf{poly}(h^d, \log q)$.

Remark 4.2 (On traversing matrices in [SU05])**.** *The paper of Shaltiel and Umans [SU05] contains several constructions, and in particular they construct both extractors and pseudorandom generators. The two different constructions use the same rationale, but rely on traversing matrices with different properties.*

Interestingly, the "traversing matrix" of Lemma 4.1 is not the one used in the PRG construction of [SU05]. The PRG construction uses a "traversing matrix" with additional properties, and this comes with the cost that we only know how to find such a matrix in time $\mathsf{poly}(q^d)$. As explained in Section 2.2, in our setting where q can be exponential in s, we cannot afford this time, and therefore cannot use the traversing matrix that was used in the PRG construction of [SU05].

Instead, we will use the matrix from Lemma 4.1, which has a modified property and is a (special case) of a traversing matrix that was used in an extractor construction in [SU05]. The advantage of this matrix is that (as stated in Lemma 4.1) it can *be found in time $\mathsf{poly}(h^d, \log q)$ and this time will be polynomial in s, even in our setting where q may be exponential in s.*

On the one hand, as explained in Section 2.3, we will need to modify the technical approach of [SU05] to accommodate for using a matrix with weaker properties. On the other hand, in our setup (where q is exponential in s) the use of the modified property, will also turn out to be an advantage in some of the steps in the approach of [SU05] (where the fact that the index i in Lemma 4.1 ranges over $[h^d]$, rather than $[q^d]$ will be crucial for the proof).

Summing up, the argument that we use in this paper gives a different proof even in the original setting of [SU05] (that is, when ignoring the additional complications needed to accommodate for "multiplicativeness"). This different approach and can potentially help in recent applications of the Shaltiel-Umans approach such as [CLO+23].

Our PRG construction is presented in Figure 1 and relies on several components from Section 3. It closely follows the high level overview given in Section 2.

Theorem 1.7 follows from the next theorem (that states the correctness of the construction) by choosing $\mathsf{m}_{\mathsf{ext}} = 1$ and $m = s$.

Theorem 4.3. *[Multiplicative PRG for nondeterministic circuit] If E is hard for exponential size nondeterministic circuits, then there exists a constant $a \geq 1$ such that for every sufficiently large s, and for every $\mathsf{m}_{\mathsf{ext}}$, m such that $\mathsf{m}_{\mathsf{ext}} \cdot m \leq s$, and $\frac{1}{2^s} \leq \delta \leq \frac{1}{s}$, the function $G : \{0,1\}^{a \cdot (\mathsf{m}_{\mathsf{ext}} + \log \frac{1}{\delta})} \to \{0,1\}^{m \cdot \mathsf{m}_{\mathsf{ext}}}$ defined in Figure 1 is a multiplicative $\overset{m}{\sim}_{(\frac{1}{s},\delta)}$-PRG for nondeterministic circuits of size s. Furthermore, G can be computed in time $\mathsf{poly}(s)$.*

Remark 4.4 (Regarding the optimality of the multiplicative PRG)**.** *As mentioned in Section 1, Theorem 1.7 is optimal in the following sense:*

- *The hardness assumption used is minimal. More specifically, the hardness assumption that E is hard for exponential size nondeterministic circuits* follows *from the conclusion of Theorem 1.7. This follows easily by adapting an argument of Impagliazzo, Shaltiel and Wigderson [ISW99] that shows that hard functions follow from (standard) PRG. We will now outline this argument and observe that it applies in our multiplicative setting.*

  Let $\delta = \frac{1}{s}$, and assume that for sufficiently large s, we have a $(\frac{1}{s}, \delta)$ multiplicative PRG

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

Figure 1: Construction of multiplicative PRG for nondeterministic circuits

Hardness assumption: We are assuming that E is hard for exponential size nondeterministic circuits. Namely, that there exist constants $0 < \beta < 1 < B$ and a function $f : \{0,1\}^* \to \{0,1\}$ such that:

Easiness: f is computable in time $2^{B\ell}$ on inputs of length ℓ.

Hardness: For every sufficiently large ℓ, nondeterministic circuits of size $2^{\beta\ell}$ fail to compute f on inputs of length ℓ.

Input parameters: We are given a sufficiently large integer s, and additional integer parameters $\mathsf{m_{ext}}$, m and $\delta > 0$, such that $\mathsf{m_{ext}} \cdot m \leq s$ and $\frac{1}{2^s} \leq \delta \leq \frac{1}{s}$.

Goal: Construct a $\overset{m}{\sim}_{(\frac{1}{s},\delta)}$-PRG for size s nondeterministic circuits, with output length $\mathsf{m_{out}} = \mathsf{m_{ext}} \cdot m$ and seed length $\mathsf{m_{ext}} + O(\log(1/\delta))$.

Construction:

Setting parameters for Traversing Matrix: Let c_0, c_q be sufficiently large universal constants that will be chosen in the proof. We set $h = s$, $d = \frac{c_0}{\beta}$, $\ell = d \log h$ and $q = \frac{2^{\mathsf{m_{ext}}}}{\delta^{c_q}}$. Let $\mathbb{F}_q$ be the field with q elements (we will be assuming that q is a power of 2). Let $H \subseteq \mathbb{F}_q$ be a set of size h from lemma 4.1 and let A be the invertible $d \times d$ matrix over $\mathbb{F}_q$ as constructed in lemma 4.1, and note that A can be found in time $\mathsf{poly}(h^d, \log q) = \mathsf{poly}(2^\ell)$ (here we should also verify that we can meet the conditions of Lemma 4.1, see Remark 4.5). We identify $\{0,1\}^\ell$ with H^d using the bijective function $\phi : \{0,1\}^\ell \to H^d$, such that for every $x \in \{0,1\}^\ell$, $\phi(x) = A^x \cdot w_0$ where w_0 is some nonzero vector in H^d, and we identify $x \in \{0,1\}^\ell$ with a number $0 \leq x < 2^\ell = h^d$). Note that ϕ can be computed in time $\mathsf{poly}(h^d, 2^\ell, \log q) = \mathsf{poly}(2^\ell)$.

Low degree extension: We define $\hat{f} : \mathbb{F}_q^d \to \mathbb{F}_q$ to be the "low degree extension" of f (a precise statement is given in Lemma 3.9). This is a polynomial of degree $\hat{h} = hd$ such that for every $x \in \{0,1\}^\ell$, $\hat{f}(\phi(x)) = f(x)$. We have that $\hat{f}$ is computable in time $\mathsf{poly}(2^\ell, 2^{B\ell}, \log q) = \mathsf{poly}(s)$.

Leftover hash lemma seeded extractor: Let c_ϵ be a sufficiently large universal constant that will be chosen in the proof, set $\epsilon = \frac{\delta}{c_\epsilon \cdot s \cdot 3m^2}$, and $\mathsf{SExt} : \{0,1\}^{\log q} \times \{0,1\}^{\log q} \to \{0,1\}^{\mathsf{m_{ext}}}$ be the $(\mathsf{m_{ext}} + 2\log(1/\epsilon), \epsilon)$-strong seeded extractor of the "Leftover hash lemma" (formally specified in Theorem 3.8). Note that by choosing c_q to be sufficiently large, we have that $\log q > \mathsf{m_{ext}} + 2\log(1/\epsilon)$.

PRG Construction: Let $g : \mathbb{F}_q^d \times \mathbb{F}_q \to \{0,1\}^{\mathsf{m_{ext}}}$ be defined by $g(w, v) = \mathsf{SExt}(\hat{f}(w), v)$.

For every $0 \leq j \leq d-1$ we define $G_j : \{0,1\}^{(d+1)\cdot\log q} \to \{0,1\}^{\mathsf{m_{ext}} \cdot m}$ as follows: Given a seed $x \in \{0,1\}^{(d+1)\log q}$ we interpret it as a pair $(w, v) \in \mathbb{F}_q^d \times \{0,1\}^{\log q}$ and define:

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

The final PRG $G : \{0,1\}^{(d+1)\cdot d\cdot\log q} \to \{0,1\}^{\mathsf{m_{ext}} \cdot m}$ is defined as follows: Given a seed $x = (x_0, \ldots, x_{d-1}) \in (\mathbb{F}_q^{d+1})^d$ define:

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

Note that by our choices, the seed length is $(d+1)\cdot d\cdot\log q = a\cdot(\mathsf{m_{ext}} + \log(1/\delta))$, for some constant a that depends only on β. Furthermore, G is computable in time $\mathsf{poly}(s)$ (where the exponent of the polynomial in s is a universal constant times $\frac{B}{\beta}$).

for nondeterministic circuits of size s, where G can be computed in time $\mathsf{poly}(s)$.

We define the function $f : \{0,1\}^{r+1} \to \{0,1\}$ by $f(z) = 1$ if and only if there exists $x \in \{0,1\}^r$

such that z is a prefix of $G(x)$. By construction $f : \{0,1\}^{O(\log s)} \to \{0,1\}$ can be computed in time $2^r \cdot \mathsf{poly}(s) = \mathsf{poly}(s)$. We claim that it cannot be computed by a size s nondeterministic circuit, as if there were such a circuit D, it would follow that: $\Pr[D(G(U_r)) = 1] = 1$ and $\Pr[D(U_s) = 1] \leq \frac{1}{2}$, contradicting the correctness of the multiplicative PRG.

We stress that while this argument shows that the conclusion of Theorem 1.7 implies the hardness assumption (when setting $\delta = \frac{1}{s}$), if one were to state Theorem 1.7 only for $\delta = s^{-\omega(1)}$, then the argument still works, but only shows that the function f can be computed in time $\mathsf{poly}(\frac{1}{\delta})$.

- *The seed length is minimal up to a constant factor. This is because for every $\frac{1}{2^s} \leq \delta \leq \frac{1}{s}$, given a $(1,\delta)$-PRG $G : \{0,1\}^r \to \{0,1\}^s$ for size s nondeterministic circuits, we can consider the (deterministic) circuit $D_{z_0}(z)$ that checks whether $z = z_0$, for $z_0 = G(0^r)$. By using nonuniformity to hardwire $G(0^r)$ to D_{z_0}, D is a size s circuit. However, it is easy to see that if $r = o(\log \frac{1}{\delta})$, then D is not fooled by the PRG. This is because $\Pr[D(U_s) = 1] = \frac{1}{2^s}$, and $\Pr[D(G(U_r)) = 1] = \frac{1}{2^r}$, and we get that:*

$$\frac{1}{2^r} = \Pr[D(G(U_r)) = 1] \leq e \cdot \Pr[D(U_s) = 1] + \delta = \frac{1}{2^s} + \delta = O(\delta),$$

which does not hold for $r = o(\log \frac{1}{\delta})$.

Remark 4.5 (Regarding the parameter choices in Figure 1)**.** *In Figure 1 we allow ourselves to write expressions like $\ell = d \log h$, and assume that ℓ is an integer, ignoring ceilings or floors. We use this convention throughout the paper. One place where more care should be made is when choosing q, h and d, and applying Lemma 4.1. This is because Lemma 4.1 has "divisibility requirements" and requires that h is a power of 2, q is a power of h, and d and $\log_h q$ are relatively prime. We need to justify that we can meet these conditions with the choices made in Figure 1.*

Having pointed this out, we observe that this can be achieved with no difficulty, by insisting that h is a power of 2, $q = h^c$ for c that is a power of 2, and choosing the constant d to be a power of 3. These choices can increase d and h by at most a factor of 3, and may cause q to be squared. These changes are insignificant, and using them, we meet the more intricate divisibility requirements of Lemma 4.1.

Another technicality that we ignore in Figure 1 is that the map ϕ cannot output the zero vector (and so formally it is not a bijection from $\{0,1\}^\ell$ to H^d). It is however a bijection from $\{0,\ldots,2^\ell - 2\}$ to $H^d \setminus 0$, and so one element is "missing". This can be solved by mapping $2^\ell - 1$ to the zero vector, and treating it separately. We ignore this technicality (which is immaterial in the argument).

The remainder of this section is devoted to the proof of Theorem 4.3.

4.2 Proof of Theorem 4.3

We now prove Theorem 4.3. The reader is referred to Section 2 for a high level overview of the proof.

Assume that G is not a PRG for nondeterministic circuits of size s with respect to $\overset{m}{\sim}_{(\frac{1}{s},\delta)}$. Our goal is to contradict the hardness assumption, and construct a nondeterministic circuit B of size $2^{\beta \cdot \ell}$ that computes f.

Throughout this proof we will consider a probability space with the following independently chosen random variables

$$W \leftarrow \mathbb{F}_q^d, V \leftarrow \mathbb{F}_q, R \leftarrow U_{\mathsf{m_{ext}}}.$$

Recall that G was defined by $G(x_0,\ldots,x_{d-1}) = G_0(x_0) \oplus \cdots \oplus G_{d-1}(x_{d-1})$. We have that G is not a PRG for nondeterministic circuits of size s with respect to $\overset{m}{\sim}_{(\frac{1}{s},\delta)}$. By Lemma 3.5 we conclude that for every $0 \leq j \leq d-1$, there exists a nondeterministic circuit D'_j of size $s + m \cdot \mathsf{m_{ext}} = O(s)$, such that G_j is not a PRG for D'_j with respect to $\overset{m}{\sim}_{(\frac{1}{s},\delta)}$.

4.2.1 Using a Multiplicative Hybrid Argument

We now use the multiplicative hybrid argument of Lemma 3.4 to get the following:

Claim 4.6. *For every $0 \leq j \leq d-1$ there exists an index $i_j^* \in \{0,\ldots,m-1\}$, and a non-deterministic circuit $D_j : (\{0,1\}^{\mathsf{m_{ext}}})^{i_j^*+1} \to \{0,1\}$ of size $s + \mathsf{m_{ext}} \cdot m = O(s)$ such that if we denote:*

- $Z_{j,i} = (G_j(W,V))_i = g\left(\left(A^{h^j}\right)^i \cdot W, V\right)$, *(namely the i'th block of $G_j(W,V)$).*
- $p_{1,j} = \Pr[D_j(Z_{j,0},\ldots,Z_{j,i_j^*-1},R)=1]$
- $p_{2,j} = \Pr[D_j(Z_{j,0},\ldots,Z_{j,i_j^*-1},Z_{j,i_j^*})=1]$

then for every $0 \leq j \leq d-1$, we have that $p_{1,j} \overset{m}{\not\sim}_{(\frac{1}{s\cdot m},\frac{\delta}{3m})} p_{2,j}$.

Proof of claim 4.6. We have that for every $0 \leq j < d$, the distribution $G_j(W,V) = (Z_{j,0},\ldots,Z_{j,m-1})$ is not pseudorandom for D'_j w.r.t. $\overset{m}{\sim}_{(\frac{1}{s},\delta)}$. This is precisely the setup considered in Lemma 3.4. More precisely, if we consider additional independent variables

$$R_0,\ldots,R_{m-1} \leftarrow \{0,1\}^{\mathsf{m_{ext}}},$$

then, by Lemma 3.4 we conclude that for every $0 \leq j \leq d-1$, there exists an $0 \leq i_j^* \leq m-1$ such that

$$\Pr[D'_j(Z_{j,0},\ldots,Z_{j,i_j^*-1},R_{i_j^*},\ldots,R_{m-1})=1] \overset{m}{\not\sim}_{(\epsilon',\delta')} \Pr[D'_j(Z_{j,0},\ldots,Z_{j,i_j^*-1},Z_{j,i_j^*},R_{i_j^*+1},\ldots,R_{m-1})=1],$$

where $\epsilon' = \frac{1}{sm}$ and $\delta' = \frac{\delta}{3m}$.

For every $0 \leq j \leq d-1$, we can think about $D'_j(x_0,\ldots,x_{d-1})$ as accepting two inputs $a = (x_0,\ldots,x_{i_j^*})$ and $b = (x_{i_j^*+1},\ldots,x_{m-1})$. We now plan to use Lemma 3.6. For this purpose, we define:

- $W_1^j = (Z_{j,0},\ldots,Z_{j,i_j^*-1},R)$.
- $W_2^j = (Z_{j,0},\ldots,Z_{j,i_j^*-1},Z_{j,i_j^*})$.

For every $0 \leq j \leq d-1$, we have that:

$$\Pr[D'_j(Z_{j,0},\ldots,Z_{j,i_j^*-1},R_{i_j^*},\ldots,R_{m-1})=1] = \Pr[D'_j(W_1^j,(R_{i_j^*+1},\ldots,R_{m-1}))=1].$$
$$\Pr[D'_j(Z_{j,0},\ldots,Z_{j,i_j^*-1},Z_{j,i_j^*},R_{i_j^*+1},\ldots,R_{m-1})=1] = \Pr[D'_j(W_2^j,(R_{i_j^*+1},\ldots,R_{m-1}))=1]$$

and we can think of the second input $(R_{i_j^*+1},\ldots,R_{m-1})$ as independent random coins tossed by D'_j. By Lemma 3.6 we conclude that for every j, there exists a fixing to $R_{i_j^*+1},\ldots,R_{m-1}$ that preserves the distinguishing advantage, and setting D_j to be the circuit obtained from D'_j by hardwiring this fixing, we get that for every $0 \leq j \leq d-1$ there exists a nondeterministic circuit D_j of size $O(s)$ such that:

$$\Pr[D_j(Z_{j,0},\ldots,Z_{j,i_j^*-1},R)=1] \overset{m}{\not\sim}_{(\epsilon',\delta')} \Pr[D_j(Z_{j,0},\ldots,Z_{j,i_j^*-1},Z_{j,i_j^*})=1].$$

□

4.2.2 Distinguishers That Rely On Previous Elements

Loosely speaking, Claim 4.6 says that for every $0 \leq j \leq d-1$, if we set $W' = \left(A^{h_j}\right)^{i_j^*} \cdot W$, then the circuit D_j from Claim 4.6 distinguishes $Z_{j,i_j^*} = g(W', V)$ from R, if it can be provided with the "previous elements" $Z_{j,0}, \ldots, Z_{j,i_j^*-1}$. In the next definition we set up notation to refer to these previous elements as a function of the "interesting element" W'.

Definition 4.7 (Previous elements)**.** *Throughout the proof we will use the following notation:*

- *Given $0 \leq i \leq m-1$, $0 \leq j < d$, and $w \in \mathbb{F}_q^d$ we define:*

$$\mathsf{Prv}_{i,j}(w) = \left(A^{h^j}\right)^{-i} \cdot w.$$

- *Given $0 \leq j < d$ and $w \in \mathbb{F}_q^d$, we define:*

$$\mathsf{Prv}_j(w) = \mathsf{Prv}_{m-1,j}(w), \mathsf{Prv}_{m-2,j}(w), \ldots, \mathsf{Prv}_{1,j}(w).$$

- *Given $0 \leq j < d$, $w \in \mathbb{F}_q^d$, and $v \in \mathbb{F}_q$ we define:*

$$\mathsf{Prv}_j(w, v) = g(\mathsf{Prv}_{m-1,j}(w), v), g(\mathsf{Prv}_{m-2,j}(w), v), \ldots, g(\mathsf{Prv}_{1,j}(w), v).$$

This notation is set up so that we can restate Claim 4.6 in the following form:

Claim 4.8. *For every $0 \leq j \leq d-1$ there exists a non-deterministic circuit D_j of size $O(s)$ such that:*

- $p_{1,j} = \Pr[D_j(\mathsf{Prv}_j(W, V), R) = 1]$.
- $p_{2,j} = \Pr[D_j(\mathsf{Prv}_j(W, V), g(W, V)) = 1]$.
- $p_{1,j} \overset{m}{\not\approx}_{(\frac{1}{s \cdot m}, \frac{\delta}{3m})} p_{2,j}$.

We remark that in Claim 4.8, for every j, the circuit D_j receives $m-1$ inputs in $\mathsf{Prv}_j(W, V)$, and not just $i_j^* - 1$ as in Claim 4.6. This is because the circuit D_j can ignore the $m - i_j^*$ inputs that it doesn't use.

4.2.3 The "Basic Probability Space" and Interleaved Curves

We need the following definition of a degree r curve.

Definition 4.9 (Degree r curve passing through given $r+1$ points)**.** *For distinct $r+1$ elements $t_0, \ldots, t_r \in \mathbb{F}_q$ and (not necessarily distinct) $y_0, \ldots, y_r \in \mathbb{F}_q^d$ we define $C_{\substack{t_0,\ldots,t_r \\ y_0,\ldots,y_r}} : \mathbb{F}_q \to \mathbb{F}_q^d$ to be the unique degree r polynomial such that for every $0 \leq i \leq r$, $C_{\substack{t_0,\ldots,t_r \\ y_0,\ldots,y_r}}(t_i) = y_i$.*

We now introduce a probability space which we will refer to as "the basic probability space". The purpose of this probability space is to select two random low degree curves C^1, C^2, so that later, we will be able to use the probabilistic method to argue that there exist two low degree curves C^1, C^2 with some very specific properties.

This argument imitates a similar argument from [SU05]. However, as mentioned in Section 2.3 the probability space that we consider here is slightly different than that used in [SU05]. See Remark 4.12 for details. The random experiment of the basic probability space is defined below.

Definition 4.10 (The basic probability space)**.** *Let $r = c_r \cdot d = \frac{c_r \cdot c_0}{\beta}$ for a sufficiently large universal constant c_r that will be chosen later in the proof and note that $q - 1 \geq (d+1) \cdot r$. We define the basic probability space as follows. Let:*

$$t_1^{(0)}, \ldots, t_r^{(0)},\ t_1^{(1)}, \ldots, t_r^{(1)},\ \ldots,\ t_1^{(d)}, \ldots, t_r^{(d)} \underset{\textit{distinct}}{\leftarrow} \mathbb{F}_q \setminus \{0\}$$

be distinct elements chosen randomly, and let

$$y_1^{(0)}, \ldots, y_r^{(0)},\ y_1^{(1)}, \ldots, y_r^{(1)},\ \ldots,\ y_1^{(d)}, \ldots, y_r^{(d)} \leftarrow \mathbb{F}_q^d$$

be chosen uniformly and independently at random.

We give some intuition for the choice of the basic probabiluty space. The selection of the random variables

$$t_1^{(0)}, \ldots, t_r^{(0)},\ t_1^{(1)}, \ldots, t_r^{(1)},\ \ldots,\ t_1^{(d)}, \ldots, t_r^{(d)}$$

and

$$y_1^{(0)}, \ldots, y_r^{(0)},\ y_1^{(1)}, \ldots, y_r^{(1)},\ \ldots,\ y_1^{(d)}, \ldots, y_r^{(d)}$$

above will be used to define two curves C^1, C^2 of degree $r' = (d+1) \cdot r$. As in [SU05] these curves will be "interleaved". This formally means that the two curves C^1, C^2 will intersect in a very specific way.

Definition 4.11 (Interleaved curves)**.**

- *For every $j \in \{0, \ldots, d\}$,*

$$A_j = \begin{cases} I & \text{if } j = 0 \\ A^{h^{j-1}} & \text{if } 1 \leq j \leq d \end{cases}$$

- *For all $k \in [r]$, $j \in \{0, \ldots, d\}$, and $u \in \{1, 2\}$, define*

$$y_k^{(j,u)} = \begin{cases} y_k^{(j)} & \text{if } u = 1 \\ A_j y_k^{(j)} & \text{if } u = 2 \end{cases}$$

For every $u \in \{1, 2\}$, define the degree $r' := (d+1) \cdot r$ curve:

$$C^u = C \begin{array}{cccccccc} 0, & t_1^{(0)}, & \ldots, & t_r^{(0)}, & \ldots, & t_1^{(d)} & \ldots, & t_r^{(d)} \\ w_0, & y_1^{(0,u)}, & \ldots, & y_r^{(0,u)}, & \ldots, & y_1^{(d,u)} & \ldots, & y_r^{(d,u)} \end{array}$$

Remark 4.12. *In [SU05] each of the individual curves C^1, C^2 is distributed like a uniform low degree curve (with careful correlations between them). In definition 4.11 each individual curves C^1, C^2 is not uniform. More specifically, we insist that each individual curve passes through w_0 for $t = 0$. This will be crucial in our application. Loosely speaking, this enables us to reduce the size of the circuit that computes f from depending on q to depending on h. This is crucial in our application where q is much larger than h.*

The two curves C^1, C^2 are arranged so that they have the following properties:

Claim 4.13 (Curves are interleaved)**.** *For every $0 \leq j < d$, and every $k \in [r]$:*

- $C^1(t_k^{(0)}) = C^2(t_k^{(0)})$.
- $C^1(t_k^{(j+1)}) = A^{-h^j} \cdot C^2(t_k^{(j+1)})$.

At a high level, Claim 4.13 says that C^1 and C^2 agree on r points. Similarly for every $0 < j < d$, if we shift the curve C^2 (by multiplying it with the regular matrix A^{-h^j}, then the two curves agree on r points.

We will be interested not only in the "original curves" C^i, C^2 but also by curves obtained by "shifting" each individual curve by an "offset" i. More specifically, note that as A is a regular matrix, for every i, $A^i \cdot C$ is also a degree r' curve.

Definition 4.14 (Shifting the curves)**.** *For every $0 \leq i \leq h^d - 1$, $1 \leq j \leq d$, $k \in [r]$, and $u \in \{1,2\}$ denote:*

- $y_k^{(i,j,u)} = A^i \cdot y_k^{(j,u)}$.
- $C_i^u = A^i \cdot C^u$.

Using the fact that A is a regular matrix, we immediately conclude that for every $0 \leq i \leq h^d - 1$, $C_i^1, C_i^2 : \mathbb{F}_q \to \mathbb{F}_q^d$ are degree r' polynomials. Furthermore, the curves C_i^1, C_i^2 are "interleaved" in the same sense, as the original curves.

Claim 4.15 (Shifted curves are interleaved)**.** *For every $0 \leq i \leq h^d - 1$, every $0 \leq j < d$, and every $k \in [r]$:*

- $C_i^1(t_k^{(0)}) = C_i^2(t_k^{(0)})$.
- $C_i^1(t_k^{(j+1)}) = A^{-h^j} \cdot C_i^2(t_k^{(j+1)})$.

We will also rely on the following properties of the selection of the curve in the basic probability space.

Lemma 4.16 (Sampling)**.** *For every $i \in \{0, \ldots, h^d - 1\}$, and $u \in \{1,2\}$, the degree r' random curve C_i^u is uniformly distributed over the family of all degree r' curves passing through $(0, w_0)$. Consequently, the $q-1$ random variables $\left(C_i^u(t)\right)_{t \in \mathbb{F}_q \setminus \{0\}}$ are r'-wise independent.*

Lemma 4.17 (Independence from interpolation locations)**.** *For every $i \in \{0, \ldots, h^d - 1\}$, and $u \in \{1,2\}$, the random curve C_i^u is independent of the interpolation points $\left(t_k^{(j)}\right)_{0 \leq j \leq d,\, 1 \leq k \leq r}$. In particular, conditioning on any value of the polynomial C_i^u, the "interpolation locations" $\left(t_k^{(j)}\right)_{0 \leq j \leq d,\, 1 \leq k \leq r}$ are distributed like $(d+1) \cdot r$ random distinct elements in $\mathbb{F}_q \setminus \{0\}$.*

Proof of Lemma 4.16 and Lemma 4.17. We prove both lemmas simultaneously. We start by proving Lemma 4.17. It is sufficient to show that for every $i \in \left\{0, \ldots, h^d - 1\right\}$ and $u \in \{1,2\}$, we have that for every fixing of $\left(t_k^{(j)}\right)_{0 \leq j \leq d,\, 1 \leq k \leq r}$, the curve C_i^u is uniformly distributed over degree r' curves that pass through the point $(0, w_0)$.

We begin by noting that for a random variable $y \leftarrow \mathbb{F}_q^d$, an invertible matrix $A \in \mathbb{F}_q^{d \times d}$ and an integer i, $A^i y$ is uniformly distributed as A^i is invertible. This means that for every $i \in \left\{0, \ldots, h^d - 1\right\}$ and $u \in \{1,2\}$, we can imagine that for fixed choice of distinct interpolation points $t_1, \ldots, t_{r'}$, the curve C_i^u was chosen by

$$C_i^u = C_{\substack{0, t_1, \ldots, t_{r'} \\ w_0, y_1, \ldots, y_{r'}}}$$

where $y_1, \ldots, y_{r'} \leftarrow \mathbb{F}_q^d$.

Fix some $i \in \left\{0, \ldots, h^d - 1\right\}$ and $u \in \{1,2\}$, the curve $C_i^u : \mathbb{F}_q \to \mathbb{F}_q^d$ can be thought of as d polynomials $C_0, \ldots, C_{j-1} : \mathbb{F}_q \to \mathbb{F}_q$, so that for every $t \in \mathbb{F}_q$, $C_i^u(t) = (C_0(t), \ldots, C_{d-1}(t))$.

We have that $C_i^u(0) = w_0$ which gives that for every $0 \leq j \leq d-1$, $C_j(0)$ is fixed. Hence, it suffices to prove that for every j, the remaining r' coefficients are i.i.d. over $\mathbb{F}_q$.

For every $0 \leq j \leq d-1$, we denote $C_j(t) = \sum_{0 \leq k \leq r'} a_k^{(j)} t^k$, where $\forall k : a_k^{(j)} \in \mathbb{F}_q$ and $y_k = (y_k^{(j)})_{k \in [r']}$. Note that we have already seen that for every j, $a_0^{(j)} = C_j(0) = w_0^{(j)}$. It follows that:

$$\begin{pmatrix} 0 & 0 & \cdots & 1 \\ t_1^{r'} & t_1^{r'-1} & \cdots & 1 \\ \vdots & \vdots & \ddots & \vdots \\ t_{r'}^{r'} & t_{r'}^{r'-1} & \cdots & 1 \end{pmatrix} \begin{pmatrix} a_{r'}^{(j)} \\ a_{r'-1}^{(j)} \\ \vdots \\ a_0^{(j)} \end{pmatrix} = \begin{pmatrix} w_0^{(j)} \\ y_1^{(j)} \\ \vdots \\ y_{r'}^{(j)} \end{pmatrix}$$

Where the first row corresponds to $t_0 = 0$. The matrix on the left-hand side is a Vandermonde matrix $\mathcal{V}_{0,t_1,\ldots,t_{r'}}$, and therefore invertible, thus:

$$\begin{pmatrix} a_{r'}^{(j)} \\ a_{r'-1}^{(j)} \\ \vdots \\ w_0^{(j)} \end{pmatrix} = (\mathcal{V}_{0,t_1,\ldots,t_{r'}})^{-1} \cdot \begin{pmatrix} w_0^{(j)} \\ y_1^{(j)} \\ \vdots \\ y_{r'}^{(j)} \end{pmatrix}$$

As w_0 is fixed, there is a bijective map between the values $(a_k^{(j)})_{k \in [r']}$ and $(y_k^{(j)})_{k \in [r']}$ for all $j \in [d]$, and overall a bijective map between $(y_k)_{k \in [r']}$ and $(a_k^{(j)})_{k \in [r'], j \in [d]}$ which means C_i^u is uniformly distributed among all degree r' curves passing through $(0, w_0)$.

We now prove Lemma 4.16. To prove that $(C_i^u(t))_{t \in \mathbb{F}_q \setminus \{0\}}$ is r'-wise independent, we will show that for any $t'_1, \ldots, t'_{r'} \in \mathbb{F}_q$, the random variable $(C_i^u(t'_k))_{k \in [r']}$ is uniformly distributed. In fact, we show the stronger statement that this random variable is uniformly distributed even for fixed $t_1, \ldots, t_{r'}$.

Note that if we add $t'_0 = 0$, for every $0 \leq j \leq d-1$, we have that:

$$\mathcal{V}_{0,t'_1,\ldots,t'_{r'}} \cdot (a_k^{(j)})_{0 \leq k \leq r'} = (C_j(t'_k))_{0 \leq k \leq n'}$$

We have seen that even for fixed $t_1, \ldots, t_{r'}$, we have that $(a_k^{(j)})_{k \in [r']}$ are random and independent elements, $\mathcal{V}_{0,t'_1,\ldots,t'_{r'}}$ is a bijection between $(a_k^{(j)})_{0 \leq k \leq r'}$ and $(C_i^u(t'_k))_{0 \leq k \leq r'}$, and $a_0^{(j)}$, $C_i^u(0)$ are fixed, hence there is a bijection between $(a_k^{(j)})_{k \in [r']}$ and $(C_j(t'_k))_{k \in [r']}$, and overall we have a bijection between $(a_k^{(j)})_{k \in [r'], j \in [d]}$ and $\left(C_j(t'_k)_{j \in [d]}\right)_{k \in [r']} = (C_i^u(t'_k))_{k \in [r']}$. It follows that $(C_i^u(t'_k))_{k \in [r']}$ is uniformly distributed. □

4.2.4 A Property That Identifies The Correct Polynomial

We start with the following definition.

Definition 4.18.

- *For every* $0 \leq j \leq d-1$, *we define* $\gamma_{1,j} = p_{1,j} + 4\epsilon$ *and* $\gamma_{2,j} = p_{2,j} - \epsilon$
- *For every* $0 \leq j \leq d-1$, *and every* $u \in \{1, 2\}$, *we define* $j' = j'(j, u)$, *defined by*

$$j' = \begin{cases} 0 & \text{if } u = 2 \\ j+1 & \text{if } u = 1 \end{cases}$$

Claim 4.19. *For every* $0 \leq j \leq d-1$, $\gamma_{2,j} > e^{\frac{1}{4s}} \cdot \gamma_{1,j}$.

Proof. Let $\eta = \frac{1}{s \cdot m}$ and $\delta' = \frac{\delta}{3m}$. Recall that

$$p_{j,2} > e^{\eta} \cdot p_{j,1} + \delta' > \max(\delta', e^{\eta} \cdot p_{j,1}).$$

Recall that in Figure 1 we defined $\epsilon = \frac{\delta}{c_\epsilon \cdot s \cdot 3m^2}$, for a sufficiently large constant c_ϵ that we are allowed to choose. We have that

$$\epsilon = \frac{\delta}{c_\epsilon \cdot s \cdot 3m^2} = \frac{\delta' \eta}{c_\epsilon} \leq \frac{p_{j,2} \cdot \eta}{c_\epsilon}.$$

Using that $\forall x \in [0,1]$, $1 + x \leq e^x \leq 1 + 3x$ and $1 - x \leq e^{-x} \leq 1 - x/3$, we get:

$$\frac{\gamma_{2,j}}{\gamma_{1,j}} = \frac{p_{j,2} - \epsilon}{p_{j,1} + 4\epsilon} > \frac{p_{j,2} - \frac{p_{j,2}\eta}{c_\epsilon}}{p_{j,2} \cdot e^{-\eta} + \frac{4p_{j,2}\eta}{c_\epsilon}} = \frac{p_{j,2} \cdot (1 - \frac{\eta}{c_\epsilon})}{p_{j,2} \cdot (e^{-\eta} + \frac{4\eta}{c_\epsilon})} \geq \frac{e^{-\frac{3\eta}{c_\epsilon}}}{1 - \frac{\eta}{3} + \frac{4\eta}{c_\epsilon}} \geq \frac{e^{-\frac{3\eta}{c_\epsilon}}}{e^{-(\frac{\eta}{3} - \frac{4\eta}{c_\epsilon})}} = e^{\frac{\eta}{3} - \frac{4\eta}{c_\epsilon} - \frac{3\eta}{c_\epsilon}} > e^{\frac{\eta}{4}}.$$

for $c_\epsilon \geq 100$. □

Recall that throughout the proof we consider a probability space with the following independently chosen random variables

$$W \leftarrow \mathbb{F}_q^d, V \leftarrow \mathbb{F}_q, R \leftarrow U_{\mathsf{m}_{\mathsf{ext}}}.$$

We will now add an additional independent random variable $T \leftarrow \mathbb{F}_q \setminus \{0\}$.

The next claim shows that there exists a fixing of the randomness in the basic probability space, such that for every $0 \leq j \leq d-1$, every $u \in \{1,2\}$, and every $i \in \{0, \ldots, h^d - 1\}$ the distinguishing circuit D_j can be used to specify a property that distinguishes the "correct polynomial" $\hat{p}_i^u = \hat{f} \circ C_i^u$ from any other degree $\hat{h} \cdot r'$ polynomial p such that for every $k \in [r]$, $p(t_k^{(j')}) = \hat{f}(y_k^{(i,j',u)})$. Here $j' = j'(j,u)$ is the function defined in Definition 4.18. Loosely speaking, as is the case in [SU05], the "reference points" in which we require p and $\hat{p}_i^u$ to agree on, are exactly the points in which the two curves are interleaved. The precise statement appears below.

Claim 4.20 (Existence of good curves)**.** *There exist distinct*

$$t_1^{(0)}, \ldots, t_r^{(0)},\ t_1^{(1)}, \ldots, t_r^{(1)},\ \ldots,\ t_1^{(d)}, \ldots, t_r^{(d)} \in \mathbb{F}_q \setminus \{0\}$$

and

$$y_1^{(0)}, \ldots, y_r^{(0)},\ y_1^{(1)}, \ldots, y_r^{(1)},\ \ldots,\ y_1^{(d)}, \ldots, y_r^{(d)} \in \mathbb{F}_q^d$$

such that for every $0 \leq j \leq d-1$, every $u \in \{1,2\}$, and every $i \in \{0, \ldots, h^d - 1\}$, the following holds.

***The correct polynomial passes*:** *For the degree $\hat{h} \cdot r'$ polynomial $\hat{p}_i^u : \mathbb{F}_q \to \mathbb{F}_q$ defined by $\hat{p}_i^u = \hat{f} \circ C_i^u$, we have that for every $k \in [r]$, $\hat{p}_i^u(t_k^{(j')}) = \hat{f}(y_k^{(i,j',u)})$, $\hat{p}_i^u(0) = \hat{f}(A^i \cdot w_0)$, and*

$$\Pr[D_j(\mathsf{Prv}_j(C_i^u(T), V), \mathsf{SExt}(\hat{p}_i^u(T), V)) = 1] \geq \gamma_{2,j}.$$

***No incorrect polynomial passes*:** *For every degree $\hat{h} \cdot r'$ polynomial $p : \mathbb{F}_q \to \mathbb{F}_q$ such that $p \neq \hat{p}_i^u$, that satisfies that for every $k \in [r]$, $p(t_k^{(j')}) = \hat{f}(y_k^{(i,j',u)})$, we have that*

$$\Pr[D_j(\mathsf{Prv}_j(C_i^u(T), V), \mathsf{SExt}(p(T), V)) = 1] \leq \gamma_{1,j}.$$

We show that good curves exist, by using the probabilistic method over the basic probability space. More specifically, Claim 4.21 that appears below, states that for every $0 \leq j \leq d-1$, every $u \in \{1,2\}$, and every $i \in \{0, \ldots, h^d - 1\}$, the two properties stated in Claim 4.20 hold over the choice of random interleaved curves in the basic probability space, with probability $1 - \frac{1}{5q^d}$. This means that Claim 4.20 follows from Claim 4.21

below, using a union bound over all choices of $0 \leq j \leq d-1$, $u \in \{1,2\}$, and $i \in \{0,\ldots,h^d-1\}$. This indeed follows as

$$(d+1) \cdot 2 \cdot h^d \cdot \frac{1}{5q^d} < 1,$$

which follows because d is constant,

$$q = \frac{2^{\mathsf{m}_{\mathsf{ext}}}}{\delta^{c_q}} \geq \frac{1}{\delta^{c_q}} \geq s^{c_q} = h^{c_q},$$

using the requirement that $\delta \leq \frac{1}{s}$, and we can choose the constant c_q to be sufficiently large.

4.2.5 Analyzing The Property in the Basic Probability Space

As explained above, Claim 4.20 will follow from the next claim (that analyzes the same property over a random choice in the basic probability space).

Claim 4.21. *For every $0 \leq j \leq d-1$, every $u \in \{1,2\}$, and every $i \in \{0,\ldots,h^d-1\}$, we have that except with probability at most $\frac{1}{5q^d}$ over the basic probability space, the following holds.*

***The correct polynomial passes*:** *For the degree $\hat{h} \cdot r'$ polynomial $\hat{p}_i^u : \mathbb{F}_q \to \mathbb{F}_q$ defined by $\hat{p}_i^u = \hat{f} \circ C_i^u$, we have that for every $k \in [r]$, $\hat{p}_i^u(t_k^{(j')}) = \hat{f}(y_k^{(i,j',u)})$, $\hat{p}_i^u(0) = \hat{f}(A^i \cdot w_0)$, and*

$$\Pr[D_j(\mathsf{Prv}_j(C_i^u(T),V), \mathsf{SExt}(\hat{p}_i^u(T),V)) = 1] \geq \gamma_{2,j}.$$

***No incorrect polynomial passes*:** *For every degree $\hat{h} \cdot r'$ polynomial $p : \mathbb{F}_q \to \mathbb{F}_q$ such that $p \neq \hat{p}_i^u$, that satisfies that for every $k \in [r]$, $p(t_k^{(j')}) = \hat{f}(y_k^{(i,j',u)})$, we have that*

$$\Pr[D_j(\mathsf{Prv}_j(C_i^u(T),V), \mathsf{SExt}(p(T),V)) = 1] \leq \gamma_{1,j}.$$

Proof of Claim 4.21. By Claim 4.8, for every $0 \leq j \leq d-1$ there exists a non-deterministic circuit D_j of size $O(s)$ such that:

- $p_{1,j} = \Pr[D_j(\mathsf{Prv}_j(W,V),R) = 1]$.
- $p_{2,j} = \Pr[D_j(\mathsf{Prv}_j(W,V), \mathsf{SExt}(\hat{f}(W),V)) = 1]$.

By a standard application of an r-wise independent tail inequality [BR94] we get that for every $i \in \{0,\ldots,h^d-1\}$ and $j \in \{0,\ldots,d-1\}$, the values $p_{1,j}$ and $p_{2,j}$ (which are probabilities over the choice $W \leftarrow \mathbb{F}_q^d$) are approximated by values $p_{i,1,j}^u, p_{i,2,j}^u$ (which are defined below by replacing W with $C_i^u(T)$ for $T \leftarrow \mathbb{F}_q \setminus \{0\}$). This is stated formally in the next claim.

Claim 4.22 (Sampling preserves $p_{1,j}$ and $p_{2,j}$)**.** *For every $i \in \{0,\ldots,h^d-1\}$, $j \in \{0,\ldots,d-1\}$ and $u \in \{1,2\}$, except for probability $\frac{1}{10q^d}$ over the basic probability space we have that:*

- $p_{i,1,j}^u = \Pr[D_j(\mathsf{Prv}_j(C_i^u(T),V),R) = 1] \leq p_{1,j} + \epsilon$, *and*
- $p_{i,2,j}^u = \Pr[D_j(\mathsf{Prv}_j(C_i^u(T),V), \mathsf{SExt}(\hat{f}(C_i^u(T)),V)) = 1] \geq p_{2,j} - \epsilon = \gamma_{2,j} > \gamma_{1,j}$.

The proof of Claim 4.22 follows by a straightforward application of the r-wise independent tail inequality of [BR94] (stated in Theorem 3.13).

Proof of Claim 4.22. For every $j \in \{0,\ldots,d-1\}$, we have that:

- $p_{1,j} = \Pr[D_j(\mathsf{Prv}_j(W,V),R) = 1]$

- $p_{2,j} = \Pr[D_j(\mathsf{Prv}_j(W,V), \mathsf{SExt}(\hat{f}(W),V)) = 1]$

Define $v_1(w) = \Pr[D_j(\mathsf{Prv}_j(w,V)), R) = 1]$ and $v_2(w) = \Pr[D_j(\mathsf{Prv}_j(w,V)), \mathsf{SExt}(\hat{f}(w),V)) = 1]$, so that $p_{1,j} = \mathbb{E}_{w \leftarrow \mathbb{F}_q^d}[v_1(w)]$ and $p_{2,j} = \mathbb{E}_{w \leftarrow \mathbb{F}_q^d}[v_2(w)]$. Recall that we are considering the basic probability space under which the conditions of Lemmas 4.16 and 4.17 are satisfied for the random curve C_i^v.

For every $t \in \mathbb{F}_q \setminus \{0\}$, every $i \in \{0, \ldots, h^d - 1\}$ and every $u \in \{1,2\}$ we define the random variable $R_{t,i}^u = C_i^u(t)$. By Lemma 4.16, we have that for every $i \in \{0, \ldots, h^d - 1\}$ and every $v \in \{1,2\}$, the random variables $(R_{t,i}^v)_{t \in \mathbb{F}_q \setminus \{0\}}$ are r'-wise independent according to and in particular r-wise independent. This means that we can apply the r-wise tail inequality from Theorem 3.13 to argue that for every $i \in \{0, \ldots, h^d - 1\}$ and every $v \in \{1,2\}$, $p_{1,j}$ and $p_{2,j}$ are with high probability approximated by $p_{i,1,j}^u$ and $p_{i,2,j}^u$. More specifically, for every $i \in \{0, \ldots, h^d - 1\}$ and every $v \in \{1,2\}$, Theorem 3.13 implies the probability that $|p_{1,j} - p_{i,1,j}^u| > \epsilon$ is at most

$$8 \cdot \left(\frac{2r}{\epsilon^2 (q-1)} \right)^{r/2} \leq \frac{1}{20q^d}$$

where the last inequality follows because we can choose the constants c_r, c_q in the definition of $r = c_r \cdot d$ and $q = \frac{2^{\mathsf{m_{ext}}}}{\delta^{cq}}$ to be sufficiently large so that $\frac{2r}{\epsilon^2 \cdot (q-1)} \leq \frac{1}{\sqrt{q}}$ and $r \geq 10d$. The same reasoning gives that the probability that $|p_2 - p_{x,2}| > \epsilon$ is at most $\frac{1}{20q^d}$, and the claim follows by a union bound over these two events. □

We continue with the proof of Claim 4.21. Fix some $i \in \{0, \ldots, h^d - 1\}, j \in \{0, \ldots, d-1\}$ and $u \in \{1,2\}$. By Claim 4.22 with probability $1 - \frac{1}{10q^d}$ over the basic probability space, we have that $p_{i,1,j}^u \leq p_{1,j} + \epsilon$ and $p_{i,2,j}^u \geq p_{2,j} - \epsilon$. Fix some specific choice of fixing $(t_k^{(e)})_{k \in [r], e \in \{0,\ldots,d\}}$ and $(y_k^{(e)})_{k \in [r], e \in \{0,\ldots,d\}}$ for the basic probability space which satisfies this condition. This fixing is done so that C_i^u (which is determined by $(t_k^{(e)})_{k \in [r], e \in \{0,\ldots,d\}}$ and $(y_k^{(e)})_{k \in [r], e \in \{0,\ldots,d\}}$) is fixed to a specific polynomial. We define:

$$\mathsf{List}_{j,i}^u = \left\{ p : \mathbb{F}_q \to \mathbb{F}_q \;\middle|\; p \text{ is a degree } \hat{h} \cdot r' \text{ polynomial, and } \Pr[D_j(\mathsf{Prv}_j(C_i^u(T),V), \mathsf{SExt}(p(T),V)) = 1] > \gamma_{1,j} \right\}$$

We have seen that $\hat{p}_i^u = \hat{f} \circ C_i^u \in \mathsf{List}_{j,i}^u$. For every polynomial $p \in \mathsf{List}_{j,i}^u$ we have that:

$$\Pr[D_j(\mathsf{Prv}_j(C_i^u(T),V), \mathsf{SExt}(p(T),V)) = 1] - \Pr[D_j(\mathsf{Prv}_j(C_i^u(T),V), R) = 1]$$

$$> \gamma_{1,j} - p_{i,1,j}^u > (p_{1,j} + 4\epsilon) - (p_{1,j} + \epsilon) = 3\epsilon.$$

As T is uniform over $\mathbb{F}_q \setminus \{0\}$ and independent of (V,R), by an averaging argument, it follows that there exist a subset $V_{j,i,p}^u \subseteq \mathbb{F}_q \setminus \{0\}$ of size $\epsilon(q-1)$ such that for every $t \in V_{j,i,p}^u$, if we denote $w_t = C_i^u(t)$ we have that:

$$\Pr[D_j(\mathsf{Prv}_j(w_t,V), \mathsf{SExt}(p(t),V)) = 1] - \Pr[D_j(\mathsf{Prv}_j(w_t,V), R) = 1] > 2\epsilon.$$

For every $t \in \mathbb{F}_q \setminus \{0\}$ we define:

$$\mathsf{List}_{j,i,t}^u = \{a \in \mathbb{F}_q : \Pr[D_j(\mathsf{Prv}_j(w_t,V), \mathsf{SExt}(a,V)) = 1] - \Pr[D_j(\mathsf{Prv}_j(w_t,V), R) = 1] > \epsilon\},$$

so that for $t \in V_{j,i,p}^u$, we have that $p(t) \in \mathsf{List}_{j,i,t}^u$.

As SExt is a (k,ϵ)-strong extractor for $k = \mathsf{m_{ext}} + 2\log(1/\epsilon)$, we have that for every $t \in \mathbb{F}_q \setminus \{0\}$, $|\mathsf{List}_{j,i,t}^u| \leq 2^k$ (as otherwise the uniform distribution on $\mathsf{List}_{j,i,t}^u$ violates the guarantee of strong extractors (see Definition 3.7) with respect to the distinguisher $D_{j,t}(y,z) = D_j(\mathsf{Prv}_j(w_t,y),z)$).

We now have the setup of the celebrated Reed-Solomon list-decoding algorithm of Sudan [Sud97] (stated formally in Theorem 3.10). More precisely, there are $\mathsf{prs} = (q-1) \cdot 2^k$ points (namely, all pairs (t,y) for

$t \in \mathbb{F}_q \setminus \{0\}$ and $y \in \mathsf{List}^u_{j,i,t}$) such that every degree $\mathsf{deg} = \hat{h} \cdot r'$ polynomial $p \in \mathsf{List}^u_{j,i}$, passes through $\mathsf{agr} = \epsilon \cdot (q-1)$ of the points. By Sudan's theorem, if $\mathsf{agr} > \sqrt{2 \cdot \mathsf{prs} \cdot \mathsf{deg}}$ then $|\mathsf{List}^u_{j,i}| \leq \frac{2\mathsf{prs}}{\mathsf{agr}} = \frac{2 \cdot 2^k}{\epsilon} = \frac{2^{\mathsf{m_{ext}}+1}}{\epsilon^3}$.[15] The requirement that $\mathsf{agr} > \sqrt{2 \cdot \mathsf{prs} \cdot \mathsf{deg}}$ translates to

$$q - 1 > \frac{2 \cdot 2^k \cdot \hat{h} \cdot r'}{\epsilon^2} = \frac{2 \cdot 2^{\mathsf{m_{ext}}} \cdot \hat{h} \cdot (d+1) \cdot r}{\epsilon^4}.$$

Recall that $\hat{h} = h \cdot d \leq s^2$, $r \leq s$, $\epsilon = \frac{\delta}{c_\epsilon \cdot s \cdot m^2}$, and we have that $m, s \leq \frac{1}{\delta}$. We can choose the constant c_q to be sufficiently large so that $q = \frac{2^{\mathsf{m_{ext}}}}{\delta^{c_q}}$ satisfies the requirement.

Using that the "interpolation points" and C^u_i are independent to trim the list. For every $i \in \{0, \ldots, h^d - 1\}$, $j \in \{0, \ldots, d-1\}$ and $u \in \{1, 2\}$, in the basic probability space (namely, when choosing $(t^{(e)}_k)_{k \in [r], e \in \{0,\ldots,d\}}$ and $(y^{(e)}_k)_{k \in [r], e \in \{0,\ldots,d\}}$), the quantities $p^u_{i,1,j}, p^u_{i,2,j}$, and the set $\mathsf{List}^u_{j,i}$ are random variables that depend on the choice of $(t^{(e)}_k)_{k \in [r], e \in \{0,\ldots,d\}}$ and $(y^{(e)}_k)_{k \in [r], e \in \{0,\ldots,d\}}$ in the basic probability space.

A crucial observation is that the random variables $p^u_{i,1,j}, p^u_{i,2,j}$ and $\mathsf{List}^u_{j,i}$, depend only on the "shape" of the curve C^u_i. More formally, $p^u_{i,1,j}, p^u_{i,2,j}$ and $\mathsf{List}^u_{j,i}$ are determined by the set $\{(t, C^u_i(t)) : t \in \mathbb{F}_q\}$ which is determined by the polynomial C^u_i. However, by lemma 4.17, for every specific fixing of the polynomial C^u_i, every choice of distinct values for $(t^{(e)}_k)_{k \in [r], e \in \{0,\ldots,d\}} \in \mathbb{F}_q \setminus \{0\}$ is still possible, and equally likely. This gives that the random variable C^u_i is independent of the random variable $(t^{(e)}_k)_{k \in [r], e \in \{0,\ldots,d\}}$.

Consider conditioning the probability space of choosing $(t^{(e)}_k)_{k \in [r], e \in \{0,\ldots,d\}}$ and $(y^{(e)}_k)_{k \in [r], e \in \{0,\ldots,d\}}$, on a specific fixing of C^u_i, such that $p^u_{i,1,j} \leq p_{j,1} + \epsilon$ and $p^u_{i,2,j} \geq p_{j,2} - \epsilon$, so that by the previous discussion, $|\mathsf{List}^u_{j,i}| \leq \frac{2^{\mathsf{m_{ext}}+1}}{\epsilon^3}$.

By Claim 4.22 such a fixing occurs with probability $1 - \frac{1}{10q^d}$. We've seen that having conditioned on a specific choice of C^u_i, the set $\mathsf{List}^u_{j,x,i}$ is fixed, and yet $(t^{(k)}_{i'})_{i' \in [r], k \in \{0,\ldots,d\}}$ are distributed like r' random distinct values in $\mathbb{F}_q \setminus \{0\}$. Recall that $j' = j'(j,u)$ defined in Definition 4.18 is defined by

$$j' = \begin{cases} 0 & \text{if } u = 2. \\ j+1 & \text{if } u = 1. \end{cases}$$

Therefore, $t^{(j')}_1, \ldots, t^{(j')}_r$ are distributed like r random distinct values in $\mathbb{F}_q \setminus \{0\}$. We also have that $\hat{p}^u_i \in \mathsf{List}^u_{j,i}$, and that for every $k \in [r]$, $\hat{p}^u_i(t^{(j')}_k) = \hat{f}(C^u_i(t^{(j')}_k)) = \hat{f}(y^{(j')}_k)$. We also have that:

$$\hat{p}^u_i(0) = \hat{f}(C^u_i(0)) = \hat{f}(A^i \cdot C^u(0)) = \hat{f}(A^i \cdot w_0).$$

Every $p \in \mathsf{List}^u_{j,i}$ that is different from $\hat{p}^u_i$ agrees with $\hat{p}^u_i$ in at most $\hat{h} \cdot r'$ elements. Therefore, the probability (in this conditioned probability space) that p and $\hat{p}^u_i$ agree on the (still random) $t^{(j')}_1, \ldots, t^{(j')}_r$ is at most $\left(\frac{\hat{h} \cdot r'}{q-1}\right)^r$. We will do a union bound against all $p \in \mathsf{List}^u_{j,i}$ such that $p \neq \hat{p}^u_i$, and there are at most $|\mathsf{List}^u_{j,i}| \leq \frac{2^{\mathsf{m_{ext}}+1}}{\epsilon^3}$ such polynomials. We obtain that the probability that there exists $p \in \mathsf{List}^u_{j,i}$ such that $p \neq \hat{p}^u_i$, and yet for every $k \in [r]$, $p(t^{(j')}_k) = \hat{p}^u_i(t^{(j')}_k)$, is at most

$$\frac{2^{\mathsf{m_{ext}}+1}}{\epsilon^3} \cdot \left(\frac{\hat{h} \cdot r'}{q-1}\right)^r \leq \frac{2^{\mathsf{m_{ext}}+1} \cdot c^3_\epsilon}{\delta^8} \cdot \left(\frac{2}{\delta^3 \cdot q}\right)^{c_r \cdot d} \leq \frac{2^{\mathsf{m_{ext}}+1} \cdot c^3_\epsilon}{\delta^6} \cdot \left(\frac{\delta}{2^{\mathsf{m_{ext}}}}\right)^{c_r \cdot d} \leq \frac{1}{10q^d},$$

[15] Note that here (similar to [TV00, BGDM23, Sha25] and in contrast to [STV01]) we only use combinatorial list-decoding, and do not rely on the efficiency of Sudan's algorithm, and we could have used a combinatorial list-decoding result like the Johnson bound.

where the inequalities above follow because $\epsilon = \frac{\delta}{c_\epsilon \cdot s \cdot m^2}$, $\delta \leq \frac{1}{s}$, $\hat{h} \cdot r' = hdr' = (d+1) \cdot s \cdot c_r \cdot d^2 \leq s^3 \leq \frac{1}{\delta^3}$, $m \leq s$, and then we can take $c_q \geq 5$, so that for $q = \frac{2^{\mathsf{m_{ext}}}}{\delta^{c_q}}$, we have that $\frac{2}{\delta^3 \cdot q} \leq \frac{\delta}{2^{\mathsf{m_{ext}}}}$. The final inequality follows for a sufficiently large constant c_r.

Overall, we have that except for probability $\frac{1}{10q^d} + \frac{1}{10q^d} = \frac{1}{5q^d}$ over the choice from the basic probability space, we have that:

$$\Pr[D_j(\mathsf{Prv}_j(C_i^u(T), V), \mathsf{SExt}(\hat{p}_i^u(T), V)) = 1] \geq \gamma_{2,j}.$$

and for every degree $\hat{h} \cdot r'$ polynomial $p \neq \hat{p}_i^u$ such that $\Pr[D_j(\mathsf{Prv}_j(C_i^u(T), V), \mathsf{SExt}(p(T), V)) = 1] > \gamma_{1,j}$, there exist $k \in [r]$ such that $p(t_k^{(j')}) \neq \hat{p}_i^u(t_k^{(j')}) = \hat{f}(y_k^{(j')})$, and this completes the proof of Claim 4.21. □

4.2.6 The Procedure "Test Next Curve"

We now introduce a nondeterministic procedure that will serve as a subroutine in the final non-deterministic circuit B of size $\mathsf{poly}(s)$, which will be designed to break the function f. The goal of this procedure is to test whether a given polynomial $p : \mathbb{F}_q \to \mathbb{F}_q$ is equal to $\hat{p}_i^u = \hat{f} \circ C_i^u$. The procedure will be based on the properties guaranteed in Claim 4.20.

We will use the fixed choices of the basic probability space that are guaranteed in Claim 4.20. More specifically, We fix distinct elements

$$t_1^{(0)}, \ldots, t_r^{(0)},\ t_1^{(1)}, \ldots, t_r^{(1)},\ \ldots,\ t_1^{(d)}, \ldots, t_r^{(d)} \in \mathbb{F}_q \setminus \{0\}$$

and elements

$$y_1^{(0)}, \ldots, y_r^{(0)},\ y_1^{(1)}, \ldots, y_r^{(1)},\ \ldots,\ y_1^{(d)}, \ldots, y_r^{(d)} \in \mathbb{F}_q^d$$

that are guaranteed in Claim 4.20. Loosely speaking, by Claim 4.20 we are guaranteed that for every $i \in \{0, \ldots, h^d - 1\}$, every $j \in \{0, \ldots, d-1\}$ and every $u \in \{1, 2\}$, C_i^u is a "good curve" on which the conclusion of Claim 4.20 applies. The precise description of the procedure TEST NEXT CURVE appears in Figure 2.

4.2.7 Analyzing the Procedure "Test Next Curve"

We want to show that the procedure TEST NEXT CURVE indeed performs the intended action, as explained in Figure 2. We start with the following claim, which explains the role of the circuit $D_{i,j}^u$ in relation to Claim 4.20.

Claim 4.23. *For every $i \in \{0, \ldots, h^d - 1\}$, every $j \in \{0, \ldots, d-1\}$, every $u \in \{1, 2\}$, every polynomial $p : \mathbb{F}_q \to \mathbb{F}_q$ of degree at most $\hat{h} \cdot r'$, and every collection of polynomials $o_1, \ldots, o_{m-1} : \mathbb{F}_q \to \mathbb{F}_q$ of degree at most $\hat{h} \cdot r'$:*

- *The circuit $D_{i,j}^u$ is a nondeterministic circuit of size $\mathsf{poly}(s, d)$.*
- *If $o_1, \ldots, o_{m-1}$ are the intended polynomials, namely if $o_1 = \hat{f} \circ C_{i-(m-1)h^j}^u,\ \ldots,\ o_{m-1} = \hat{f} \circ C_{i-h^j}^u$, then,*

$$D_{i,j}^u(t, v) = D_j(\mathsf{Prv}_j(C_i^u(t), v), \mathsf{SExt}(p(t), v))$$

Proof of Claim 4.23. The circuit $D_{i,j}^u$ operates as follows. On input (t, v), it first computes the values $\mathsf{SExt}(o_1(t), v), \ldots, \mathsf{SExt}(o_{m-1}(t), v), \mathsf{SExt}(p(t), v)$. Since SExt takes inputs of length $\log q = \mathsf{poly}(s)$, it follows from Theorem 3.8 that SExt can be computed by a circuit of size $\mathsf{poly}(s)$. The polynomials $o_1, \ldots, o_{m-1}$ are of degree $\hat{h} \cdot r' = \mathsf{poly}(s, d)$, and so, evaluating them at a given point takes time $\mathsf{poly}(s, d)$. The circuit $D_{i,j}^u$ then feeds these values as input to the circuit $D_j(\mathsf{SExt}(o_1(t), v), \ldots, \mathsf{SExt}(o_{m-1}(t), v), \mathsf{SExt}(p(t), v))$, which itself has size $\mathsf{poly}(s)$. Therefore, the overall circuit $D_{i,j}^u$ has size $\mathsf{poly}(s, d)$.

Figure 2: Procedure TEST NEXT CURVE$^u_{i,j}$

The procedure TEST NEXT CURVE is defined as follows:

Non-uniformity: TEST NEXT CURVE will be hardwired with the following "non-uniform" advice.

- Distinct elements $t_1^{(0)}, \ldots, t_r^{(0)}, t_1^{(1)}, \ldots, t_r^{(1)}, \ldots, t_1^{(d)}, \ldots, t_r^{(d)} \in \mathbb{F}_q \setminus \{0\}$ whose existence is guaranteed by Claim 4.20.
- Probabilities $(\gamma_{1,j})_{j\in\{0,\ldots,d-1\}}$, $(\gamma_{2,j})_{j\in\{0,\ldots,d-1\}}$ from Definition 4.18.
- Circuits $(D_j)_{j\in\{0,\ldots,d-1\}}$.

Parameters: TEST NEXT CURVE will receive the following parameters.

- Offset i: an integer in $[0, \ldots, h^d - 1]$
- Stride j: an integer in $[0, \ldots, d-1]$
- curve type $u \in \{1, 2\}$

Input: TEST NEXT CURVE will receive the following inputs.

- $m-1$ degree $\hat{h} \cdot r'$ polynomials $o_1, \ldots, o_{m-1} : \mathbb{F}_q \to \mathbb{F}_q$ represented by their coefficients. Note that by our choice of parameters $\hat{h}r' = \mathsf{poly}(s)$ for some fixed polynomial that does not depend on d.

  We will refer to $o_1, \ldots, o_{m-1}$ as "previous polynomials", and their intended values are the polynomials: $\hat{f} \circ C^u_{i-(m-1)\cdot h^j}, \ldots, \hat{f} \circ C^u_{i-1\cdot h^j}$.
- $\alpha_1, \ldots, \alpha_r \in \mathbb{F}_q$ which we will refer to as "reference points" and are intended to be the values $\hat{f}(y_1^{(i,j',u)}), \ldots, \hat{f}(y_r^{(i,j',u)})$. Here $j' = j'(j,u)$ is the function from Definition 4.18.
- A polynomial $p : \mathbb{F}_q \to \mathbb{F}_q$ of degree $\hat{h}r'$ which we will refer to as the "tested polynomial".

  Loosely speaking, when provided with the intended previous polynomials, and intended reference points, the goal of the procedure is to test whether the tested polynomial is the intended value $\hat{f} \circ C^u_i$.

Action: Given parameters i, j, u and inputs $(o_1, \ldots, o_{m-1})$, $(\alpha_1, \ldots, \alpha_r)$ and p, the procedure TEST NEXT CURVE$^u_{i,j}(o_1, \ldots, o_{m-1}; \alpha_1, \ldots, \alpha_r; p)$ is defined as follows:

- Verify that for every $k \in [r]$, $p(t_k^{(j')}) = \alpha_k$.
- Construct the nondeterministic circuit:

$$D^u_{i,j}(t, v) = D_j\left(\mathsf{SExt}(o_1(t), v), \ldots, \mathsf{SExt}(o_{m-1}(t), v), \mathsf{SExt}(p(t), v)\right)$$

  We will argue below that this circuit is of size $\mathsf{poly}(s, d)$.
- We will now use Theorem 3.12, taking $\lambda = \frac{1}{s}$ on input $(D^u_{i,j}, \gamma_{2,j})$. For these choices, Theorem 3.12 gives a nondeterministic circuit A of size $\mathsf{poly}(s, d)$ that solves the promise problem $\mathsf{NondetLarge}_\lambda$ on these inputs, and distinguishes the case where $\Pr[D^u_{i,j}(T, V) = 1] \geq \gamma_{2,j}$ from the case $\Pr[D^u_{i,j}(T, V) = 1] \leq \gamma_{2,j} \cdot e^{-\frac{1}{s}}$. By Claim 4.19 we have that $\gamma_{1,j} < \gamma_{2,j} \cdot e^{-\frac{1}{s}}$, implying that A distinguishes the case where $\Pr[D^u_{i,j}(T, V) = 1] \geq \gamma_{2,j}$ from the case $\Pr[D^u_{i,j}(T, V) = 1] \leq \gamma_{1,j}$. The procedure TEST NEXT CURVE which is allowed to be nonuniform and nondeterministic will use A to distinguish between these two cases and check if $A(D^u_{i,j}, \gamma_{2,j}) = 1$.
- If all verification steps pass, then the procedure outputs 1.

Complexity: We will argue below that for every i, j, u, the procedure TEST NEXT CURVE$^u_{i,j}$ can be implemented by a nondeterministic circuit of size $\mathsf{poly}(s, d)$.

For the second item, recall that

$$\mathsf{Prv}_j(C^u_i(t), v) = \mathsf{SExt}(\hat{f} \circ \left(A^{h^j}\right)^{-(m-1)} \cdot C^u_i(t), v), \ldots, \mathsf{SExt}(\hat{f} \circ \left(A^{h^j}\right)^{-1} C^u_i(t), v)$$

and for every $i' \in [m-1]$, $\left(A^{h^j}\right)^{i'} \cdot C_i^u = C^u_{i-i' \cdot h^j}$, hence

$$\mathsf{Prv}_j(C_i^u(t), v) = \mathsf{SExt}(\hat{f} \circ C^u_{i-(m-1)h^j}, v), \ldots, \mathsf{SExt}(\hat{f} \circ C^u_{i-h^j}, v)$$

and if $o_1 = \hat{f} \circ C^u_{i-(m-1)h^j},\ \ldots,\ o_{m-1} = \hat{f} \circ C^u_{i-h^j}$ then

$$D^u_{i,j}(t, v) = D_j(\mathsf{Prv}_j(C_i^u(t), v), \mathsf{SExt}(p(t), v))$$

□

In the next claim, we show that the procedure TEST NEXT CURVE$^u_{i,j}$ acts as intended. More specifically that it is a size $\mathsf{poly}(s, d)$ nondeterministic circuit such that when supplied with the intended previous polynomials and reference points, it accepts a given tested polynomial p if and only if it is the intended polynomial $\hat{p}^u_i = \hat{f} \circ C^u_i$. Loosely speaking, Claim 4.20 was specifically set up to guarantee the correctness of the procedure.

Claim 4.24 (Correctness of TEST NEXT CURVE)**.** *For every $i \in \{0, \ldots, h^d - 1\}$, every $j \in \{0, \ldots, d-1\}$ and every $u \in \{1, 2\}$:*

- TEST NEXT CURVE$^u_{i,j}$ *can be simulated by a nondeterministic circuit of size* $\mathsf{poly}(s, d)$.
- *Let*

  $$o_1 = \hat{f} \circ C^u_{i-(m-1)h^j}, \ldots, o_{m-1} = \hat{f} \circ C^u_{i-h^j}$$

  and let

  $$\alpha_1 = \hat{f}(y_1^{(i,j',u)}), \ldots, \alpha_r = \hat{f}(y_r^{(i,j',u)})$$

  be the intended values for the previous polynomials and reference points, then for every polynomial $p : \mathbb{F}_q \to \mathbb{F}_q$ of degree at most $\hat{h} \cdot r'$:

$$\text{TEST NEXT CURVE}^u_{i,j}(o_1, \ldots, o_{m-1}, \alpha_1, \ldots, \alpha_r, p) = \begin{cases} 1 & \text{if } p = \hat{f} \circ C^u_i \\ 0 & \text{if } p \neq \hat{f} \circ C^u_i \end{cases}$$

Proof of Claim 4.24. The procedure TEST NEXT CURVE$^u_{i,j}$ constructs a nondeterministic circuit $D^u_{i,j}(t, v)$ of size $\mathsf{poly}(s, d)$. It then invokes the nondeterministic circuit A from Theorem 3.12, which solves the promise problem $\mathsf{NondetLarge}_s$, on the input $(D^u_{i,j}, \gamma_{2,j})$. Since A itself has size $\mathsf{poly}(s, d)$, the overall complexity of TEST NEXT CURVE$^u_{i,j}$ is $\mathsf{poly}(s, d)$.
We now prove correctness. We show that

$$\text{TEST NEXT CURVE}^u_{i,j}(o_1, \ldots, o_m, \alpha_1, \ldots, \alpha_r, p) = \begin{cases} 1 & \text{if } p = \hat{f} \circ C^u_i, \\ 0 & \text{otherwise.} \end{cases}$$

First, if there exists $k \in [r]$ such that

$$p(t_k^{(j')}) \neq \hat{f} \circ C^u_i(t_k^{(j')}) = \hat{f}(y_k^{(i,j',u)}) = \alpha_k,$$

then necessarily $p \neq \hat{f} \circ C^u_i$, and the initial verification step fails.

Assume therefore that for all $k \in [r]$,

$$p(t_k^{(j')}) = \hat{f}(y_k^{(i,j',u)}) = \alpha_k.$$

By Claim 4.20, it follows that if $p \neq \hat{f} \circ C^u_i$, then

$$\Pr[D^u_{i,j}(T,V)=1] = \Pr[D_j(\mathsf{Prv}_j(C^u_i(T),V),\mathsf{SExt}(p(T),V))=1] \leq \gamma_{1,j} < e^{-1/s}\gamma_{2,j}.$$

On the other hand, if $p = \hat{f} \circ C^u_i$, then

$$\Pr[D^u_{i,j}(T,V)=1] \geq \gamma_{2,j}.$$

Thus, the pair $(D^u_{i,j}, \gamma_{2,j})$ satisfies the promise of Theorem 3.12. Consequently, the circuit A accepts if and only if $p = \hat{f} \circ C^u_i$. Therefore, all verification steps succeed exactly in this case, completing the proof. □

4.2.8 Learning the Successive Interleaved Curves

Recall that our goal is to construct a nondeterministic circuit B that computes f and contradicts the hardness assumption. In order to abstract the process of B, we introduce another procedure that will be used as a subroutine in circuit B. This procedure called "INTERLEAVE" will be analogous to a similar procedure with the same name from [SU05]. This procedure uses the same logic as in [SU05] but relies on the procedure TEST NEXT CURVE from the previous section, rather than on the "predictor" based approach of [SU05]. (See Section 2.3 for an explanation of the difference between the two approaches. The precise description of the procedure INTERLEAVE appears in Figure 3.

Claim 4.25. *Let*

$$o^1_1 = \hat{f} \circ C^1_{i-(m-1)h^j}, \ldots, o^1_{m-1} = \hat{f} \circ C^1_{i-h^j}$$

and let

$$o^2_1 = \hat{f} \circ C^2_{i-(m-1)h^j}, \ldots, o^2_{m-1} = \hat{f} \circ C^2_{i-h^j}$$

then,

$$\textsc{Interleave}_{i,j}(o^1_1,\ldots,o^1_m,o^2_1,\ldots,o^2_m,p^1,p^2) = \begin{cases} 1 & \text{if } p^1 = \hat{f} \circ C^1_i \text{ and } p^2 = \hat{f} \circ C^2_i \\ 0 & \text{otherwise} \end{cases}$$

Proof of Claim 4.25. We first make the observation that for every $k \in [r]$,

$$o^2_{m-1}(t^{(j+1)}_k) = \hat{f} \circ C^2_{i-h^j}(t^{(j+1)}_k) = \hat{f}(A^{-h^j} \cdot C^2_i(t^{(j+1)}_k))$$

$$= \hat{f}(C^1_i(t^{(j+1)}_k) = \hat{f}(A^i \cdot C^1(t^{(j+1)}_k)) = \hat{f}(A^i \cdot y^{(j+1,1)}) = \hat{f}(y^{(i,j+1,1)}) = \hat{f}(y^{(i,j'(j,2),1)})$$

The first inequality follows because by our assumption $o^2_{m-1} = \hat{f} \circ C^2_{i-h^j}$. The second inequality follows because $C^2_{i-h^j} = A^{-h^j} \circ C^2_i$. The third inequality follows from Claim 4.15 we have that $C^1_i(t^{(j+1)}_k) = A^{-h^j} \cdot C^2_i(t^{(j+1)}_k)$. The fourth inequality follows because $C^1_i = A^i \cdot C^1$. The fifth inequality follows because by the definition of C^1, we have that $C^1(t^{(j+1)}_k) = y^{(j+1,1)}$. The sixth inequality follows because $A^i \cdot y^{(j+1,1)} = y^{(i,j+1,1)}$. The final inequality follows because by Definition 4.18, the function $j' = j'(j,u)$, is defined by

$$j' = \begin{cases} 0 & \text{if } u = 2 \\ j+1 & \text{if } u = 1 \end{cases}$$

This gives that the evaluations

$$o^2_{m-1}(t^{(j+1)}_1), \ldots, o^2_{m-1}(t^{(j+1)}_r)$$

that are used as reference points in the first invocation of the procedure $\textsc{Test Next Curve}^1_{i,j}$ are the intended values

$$\hat{f}(y^{(i,j'(j,1),1)}_1), \ldots, \hat{f}(y^{(i,j'(j,1),1)}_r)$$

Figure 3: Procedure $\text{INTERLEAVE}_{i,j}$

The procedure INTERLEAVE is defined as follows:

Non-uniformity: INTERLEAVE will be hardwired with the following "non-uniform" advice.

- The non-uniformity required for the procedure TEST NEXT CURVE.
- Distinct elements $t_1^{(0)}, \ldots, t_r^{(0)}, t_1^{(1)}, \ldots, t_r^{(1)}, \ldots, t_1^{(d)}, \ldots, t_r^{(d)} \in \mathbb{F}_q \setminus \{0\}$ whose existence is guaranteed by Claim 4.20.

Parameters: INTERLEAVE will receive the following parameters.

- Stride j: an integer in $[0, \ldots, d-1]$
- Offset i: an integer in $[0, \ldots, h^d - 1]$

Input: INTERLEAVE will receive the following inputs.

- $m-1$ degree $\hat{h} \cdot r'$ polynomials $o_1^1, \ldots, o_{m-1}^1 : \mathbb{F}_q \to \mathbb{F}_q$: "previous polynomials" which are intended to be the polynomials: $\hat{f} \circ C^1_{i-(m-1)\cdot h^j}, \ldots, \hat{f} \circ C^1_{i-1\cdot h^j}$.
- $m-1$ degree $\hat{h} \cdot r'$ polynomials $o_1^2, \ldots, o_{m-1}^2 : \mathbb{F}_q \to \mathbb{F}_q$: "previous polynomials" which are intended to be the polynomials: $\hat{f} \circ C^2_{i-(m-1)\cdot h^j}, \ldots, \hat{f} \circ C^2_{i-1\cdot h^j}$.
- degree $\hat{h}r'$ "tested" polynomial $p^1 : \mathbb{F}_q \to \mathbb{F}_q$ which is intended to be $\hat{f} \circ C_i^1$.
- degree $\hat{h}r'$ "tested" polynomial $p^2 : \mathbb{F}_q \to \mathbb{F}_q$ which is intended to be $\hat{f} \circ C_i^2$.

  Loosely speaking, when provided with the intended previous polynomials, the goal of the procedure is to test whether the tested polynomials p^1, p^2 are the intended $\hat{f} \circ C_i^1, \hat{f} \circ C_i^2$.

Action: Given parameters i, j and inputs $o_1^1, \ldots, o_{m-1}^1$, $o_1^2, \ldots, o_{m-1}^2$, p^1, p^2, the procedure $\text{INTERLEAVE}_{i,j}(o_1^1, \ldots, o_{m-1}^1; o_1^2, \ldots, o_{m-1}^2; p^1; p^2)$ is defined as follows:

- Invoke procedure $\text{TEST NEXT CURVE}^1_{i,j}(o_1^1, \ldots, o_{m-1}^1, o_{m-1}^2(t_1^{j+1}), \ldots, o_{m-1}^2(t_r^{j+1}), p^1)$ and check whether its output equals 1.
- Invoke procedure $\text{TEST NEXT CURVE}^2_{i,j}(o_1^2, \ldots, o_{m-1}^2, p^1(t_1^0), \ldots, p^1(t_r^0), p^2)$ and check whether its output equals 1.
- If all verification steps pass, then the procedure will output 1.

Complexity: Note that for every i, j, the procedure $\text{INTERLEAVE}_{i,j}$ can be implemented by a nondeterministic circuit of size $\mathsf{poly}(s, d)$.

for the reference points. By claim 4.24, it follows that

$$\text{TEST NEXT CURVE}^1_{i,j}(o_1^1, \ldots, o_{m-1}^1, o_{m-1}^2(t_1^{(j+1)}), \ldots, o_{m-1}^2(t_r^{(j+1)}), p^1) = 1$$

if and only if $p^1 = \hat{f} \circ C_i^1$. Furthermore, if the first verification step succeeds, then $p^1 = \hat{f} \circ C_i^1$ and for every $k \in [r]$,

$$p^1(t_k^{(0)}) = \hat{f}(C_i^1(t_k^{(0)})) = \hat{f}(C_2^1(t_k^{(0)})) = \hat{f}(A^i \cdot C^2(t_k^{(0)})) = \hat{f}(A^i \cdot y_k^{(0,2)}) = \hat{f}(y_k^{(i,0,2)}) = \hat{f}(y_k^{(i,j'(0,2),2)})$$

The first inequality follows because $p^1 = \hat{f} \circ C_i^1$. The second inequality follows because by Claim 4.15, we have that $C_i^1(t_k^{(0)}) = C_i^2(t_k^{(0)})$. The third inequality follows because $C_i^2 = A^i \cdot C^2$. The fourth inequality follows because $C^2(t_k^{(0)}) = y_k^{(0,2)}$. The fifth inequality follows because $A^i \cdot y_k^{(0,2)} = y_k^{(i,0,2)}$. The final inequality follows because $j'(0,2) = 0$.

This gives that the evaluations

$$p^1(t_1^{(0)}), \ldots, p^1(t_r^{(0)})$$

that are used as reference points in the first invocation of the procedure TEST NEXT CURVE$^2_{i,j}$ are the intended values

$$\hat{f}(y_1^{(i,j'(j,2),2)}), \ldots, \hat{f}(y_r^{(i,j'(j,2),2)})$$

for the reference points. By claim 4.24, it follows that

$$\text{TEST NEXT CURVE}^2_{i,j}(o^2_1, \ldots, o^2_{m-1}, p^1(t_1^{(0)}), \ldots, p^1(t_r^{(0)}), p^2) = 1$$

if and only if $p^2 = \hat{f} \circ C^2_i$. Summing up, we conclude that both verification steps succeed if and only if $p^1 = \hat{f} \circ C^1_i$ and $p^2 = \hat{f} \circ C^2_i$. □

4.2.9 Obtaining a Nondeterministic Circuit that Contradicts the Hardness Assumption

We are finally ready to finish the proof and obtain a contradiction by showing that there exists a nondeterministic circuit B of size $2^{\beta \cdot \ell}$ that computes f, and contradicts the hardness assumption. We construct such a circuit B using the procedure INTERLEAVE from the previous section. This proof follows along the same lines of the original argument of Shaltiel and Umans [SU05] when adjusted to account to the modifications that we have made because we have to work with a distinguisher, rather than a predictor (as explained in Section 2.3).
The circuit B will be hardwired with:

- The nonuniformity required to run the procedure INTERLEAVE.
- The polynomials $(\hat{f} \circ C^u_i)_{u \in \{1,2\}, i \in \{-1,\ldots,-(m-1)\}}$. Note that these are degree $\hat{h} \cdot r'$ polynomials, and can be represented by their coefficients. (Here, we allow ourselves to use negative indices, but note that as $A^{h^d-1} = I$, we have that $C^u_{-i} = C^u_{h^d-1-i}$. and the use of negative indices is just for convenience).

We now describe how the nondeterministic B operates when given an input $x \in \{0,1\}^\ell$. It will be more convenient to think of x as an integer $0 \leq x \leq 2^\ell = h^d$. Recall that in the PRG construction that is presented in Figure 1, we set things up so that:

$$f(x) = \hat{f}(\phi(x)) = \hat{f}(A^x \cdot w_0).$$

The procedure INTERLEAVE allows us to achieve the following: Given an offset i and stride j, if we already correctly computed the *previous polynomials*

$$\hat{f} \circ C^1_{i-(m-1)\cdot h^j}, \ldots, \hat{f} \circ C^1_{i-1\cdot h^j} \text{ and } \hat{f} \circ C^2_{i-(m-1)\cdot h^j}, \ldots, \hat{f} \circ C^2_{i-1\cdot h^j},$$

then we can obtain the *current polynomials*

$$\hat{f} \circ C^1_i \text{ and } \hat{f} \circ C^2_i$$

by the following process: We guess coefficients to two degree $\hat{h} \cdot r'$ polynomials p^1, p^2 (that are supposed to be the current polynomials) and then call the procedure INTERLEAVE$_{i,j}$ on the previous polynomials and guessed polynomials. By Claim 4.25 we are guaranteed that INTERLEAVE$_{i,j}$ accepts if and only if the polynomials p^1, p^2 are the current polynomials. We will refer to such a step, as "learning the polynomials at offset i with stride j"

To make this less abstract, note that B is set up so that we can learn the polynomials at offset 0 with stride 0. Furthermore, note that once we have done that, we can learn the polynomials at offset 1 with stride 0 (as we now have access to the correctly computed previous polynomials at offset 1 and stride 0 ($m-2$ of them are given as nonuniformity, and the last one was obtained in the previous learning step).

We now describe the operation of B as a sequence of d phases, where each one is composed of h^2 learning steps. More specifically: We write x in base h, that is we express x as

$$x = \sum_{0 \le j \le d-1} a_j \cdot h^j,$$

where for every $0 \le j \le d-1$, $a_j \in \{0, \ldots, h-1\}$. We will construct B as follows, using d phases.

Phase 0: We perform h^2 steps of learning as follows. For every

$$i \in \left\{k : 0 \le k \le h^2 - 1\right\}$$

we learn the polynomials at offset i using stride 0. Note that initially, we have already correctly computed the previous polynomial at offset $i = 0$ and stride 0 (by the given nonuniformity). Furthermore, that for every $1 \le k \le h^2 - 1$ the previous polynomials that we require at offset $i = k$ and stride 0 are available to us at the end of the previous step.

Phase 1: We perform h^2 steps of learning as follows. For every

$$i \in \left\{a_0 + k \cdot h^1 : 0 \le k \le h^2 - 1\right\}$$

we learn the polynomials at offset i using stride 1. Note that initially, we have already correctly computed the previous polynomial at offset $i = a_0$ and stride 1 (these polynomials were obtained in the previous phase). Furthermore, that for every $1 \le k \le h^2 - 1$ the previous polynomials that we require at offset $i = a_0 + k \cdot h^1$ and stride 1 are available to us at the end of the previous step.

Phase 2: We perform h^2 steps of learning as follows. For every

$$i \in \left\{a_0 + a_1 \cdot h + k \cdot h^2 : 0 \le k \le h^2 - 1\right\}$$

we learn the polynomials at offset i using stride 2. Note that initially, we have already correctly computed the previous polynomial at offset $i = a_0 + a_1 \cdot h$ and stride 2 (these polynomials were obtained in the previous phase). Furthermore, that for every $1 \le k \le h^2 - 1$ the previous polynomials that we require at offset $i = a_0 + a_1 \cdot h + k \cdot h^2$ and stride 2 are available to us at the end of the previous step.

$\vdots$

Phase j: We perform h^2 steps of learning as follows. For every

$$i \in \left\{a_0 + a_1 \cdot h + \ldots + a_{j-1} \cdot h^{j-1} + k \cdot h^j : 0 \le k \le h^2 - 1\right\}$$

we learn the polynomials at offset i using stride j. Note that initially, we have already correctly computed the previous polynomial at offset $i = a_0 + a_1 \cdot h + a_{j-1} \cdot h^{j-1}$ and stride j (these polynomials were obtained in the previous phase). Furthermore, that for every $1 \le k \le h^2 - 1$ the previous polynomials that we require at offset $i = a_0 + a_1 \cdot h + \ldots + a_{j-1} \cdot h^{j-1} + k \cdot h^j$ and stride j are available to us at the end of the previous step.

$\vdots$

Phase $d-1$: We perform h^2 steps of learning as follows. For every

$$i \in \left\{a_0 + a_1 \cdot h + \ldots + a_{d-2} \cdot h^{d-1} + k \cdot h^{d-1} : 0 \le k \le h^2 - 1\right\}$$

we learn the polynomials at offset i using stride $d-1$. Note that initially, we have already correctly computed the previous polynomial at offset $i = a_0 + a_1 \cdot h + a_{d-2} \cdot h^{d-2}$ and stride $d-1$ (these polynomials were obtained in the previous phase). Furthermore, that for every $1 \le k \le h^2 - 1$ the previous polynomials that we require at offset $i = a_0 + a_1 \cdot h + \ldots + a_{d-2} \cdot h^{d-2} + k \cdot h^{d-1}$ and stride j are available to us at the end of the previous step.

Note that at this point, we have correctly learned that polynomials at offset $x = \sum_{0 \leq j \leq d-1} a_j \cdot h^j$ which is one of the offsets that we learned in the final stage.

Let us be more precise. The procedure that we described is a nondeterministic procedure that made d phases, where each one consists of h^2 steps. In each one of these steps we guessed two polynomials, and tested them using the procedure INTERLEAVE. This means that if on input x, all "verification steps" made by the $d \cdot h^2$ invocations of INTERLEAVE accepted, then we are guaranteed that the polynomial $p^1 : \mathbb{F}_q \to \mathbb{F}_q$ that we guessed and verified for offset x is the correct polynomial, meaning that $p^1 = \hat{f} \circ C^1_x$. In that case we have that:

$$p^1(0) = \hat{f} \circ C^1_x(0) = \hat{f}(C^1_x(0)) = \hat{f}(A^x \cdot C^1(0)) = \hat{f}(A^x \cdot w_0) = f(x).$$

This means that if all invocations of INTERLEAVE accepted, we can now output $p^1(0)$, and this is indeed $f(x)$. If any invocation of INTERLEAVE rejected, then we output zero.
Overall, we have that we constructed a nondterministic circuit B such that for every input x:

- If $f(x) = 1$, then there exist a nondeterministic guess that leads B to output one.
- If $f(x) = 0$, there does not exist a nondeterministic guess that leads B to output one. This is because in the case that all guesses are correct, B outputs $p^1(0) = f(x) = 0$.[16]

We are left with bounding the size of the nondeterministic circuit B. The computation of B is composed of d phases, where in each one there are h^2 steps, and each one invokes the procedure INTERLEAVE that is a size $\mathsf{poly}(s, d)$ nondeterministic circuit. Recall that $h = s$. It follows that the overall size of B is $\mathsf{poly}(s, d) = (s \cdot d)^c$ for some universal constant c. It remains to show that as a function of its input length ℓ, the circuit B has size that is upper-bounded by $2^{\beta \cdot \ell}$.

Recall, that we have not yet chosen the constant c_0 that was specified in Figure 1 and governs the choice of the constant d. We are allowed to choose c_0 to be any sufficiently large universal constant. We choose $c_0 = c + 1$. With this choice, for sufficiently large s, as c_0 and β are constants and $d = \frac{c_0}{\beta}$, we have that the size of B is at most

$$(s \cdot d)^c = s^c \cdot d^c = s^c \cdot \left(\frac{c_0}{\beta}\right)^c \leq s^{c+1} = s^{c_0}.$$

Using the choices in Figure 1 we have that:

$$s^{c_0} = h^{c_0} = 2^{\beta \cdot \ell},$$

where the last inequality follows because $\ell = d \log h$, and $d = \frac{c_0}{\beta}$, and together, this gives that $2^{\beta \cdot \ell} = s^{c_0}$.

Overall, this gives that B is a nondeterministic circuit of size $2^{\beta \cdot \ell}$ that computes f correctly. This is a contradiction to the hardness assumption that is stated in Figure 1. This completes the proof of Theorem 4.2.

5 Randomness Reduction in Explicit Constructions for coNP/poly Properties

In this section we prove Theorem 1.9 by observing that it directly follows from Theorem 1.7.

Proof of Theorem 1.9. Let $\mathcal{P}$ be a language in coNP/poly such that for every sufficiently large m,

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

for $0 < \delta(m) \leq \frac{1}{m}$. As $\mathcal{P} \in \mathsf{NP/poly}$, we have that there exists a constant $c \geq 1$ such that for every sufficiently large m, there is a nondeterministic circuit D of size $s = m^c$, such that for every $z \in \{0,1\}^m$, $D(z) = 1$ if and only if $z \notin \mathcal{P}$. Note that $\frac{1}{2^s} \leq \delta(m) \leq \frac{1}{s^{1/c}}$.

[16] Another way to think about this is that the circuit B that we construct is a single valued nondeterministic circuit that computes f. See e.g., [MV05] for a definition of single valued nondeterministic circuits.

By Theorem 1.7, the assumption that E is hard for exponential size circuits yields a $(\frac{1}{s}, \delta(m))$-multiplicative PRG $G : \{0,1\}^{r=O(\log \frac{1}{\delta(m)})} \to \{0,1\}^s$ for size s nondeterministic circuits. We define $\mathsf{Construct}$ to be the randomized algorithm that on input 1^m outputs the first m bits of $G(U_r)$. By the guarantee on G, this randomized algorithm runs in time $\mathsf{poly}(s) = \mathsf{poly}(m)$, and uses $r = O(\log \frac{1}{\delta(m)})$ random coins. It follows that for every sufficiently large m:

$$\begin{aligned}
\Pr[\mathsf{Construct}(1^m) \notin \mathcal{P}] &= \Pr[G(U_r) \notin \mathcal{P}] \\
&= \Pr[D(G(U_r)) = 1] \\
&\leq e^{\frac{1}{s}} \cdot \Pr[D(U_m) = 1] + \delta(m) \\
&\leq 2 \cdot \Pr[D(U_m) = 1] + \delta(m) \\
&\leq 2 \cdot \delta(m) + \delta(m) \\
&= 3 \cdot \delta(m).
\end{aligned}$$

□

6 Multiplicative PRGs for Nonboolean Circuits

In this section we discuss our results on multiplicative nb-PRGs. In Section 6.1 we prove Theorem 1.12. In Section 6.2 we discuss applications of multiplicative nb-PRGs.

6.1 Proof of Theorem 1.12

Theorem 1.12 is a direct consequence of Theorem 1.7 and the observation that an optimal multiplicative PRG (in the boolean case) is also an optimal multiplicative nb-PRG. This observation is stated precisely below.

Theorem 6.1 (Multiplicative nb-PRG from multiplicative-PRG)**.** *If $G : \{0,1\}^r \to \{0,1\}^m$ is an (ϵ, δ)-multiplicative PRG for circuits of size s, then for every $\ell \geq 1$, G is also an $(\ell, 4 \cdot \epsilon, \frac{4 \cdot \delta \cdot 2^\ell}{\epsilon})$-multiplicative nb-PRG for circuits of size $s' = s - O(\ell)$.*

We now note that Theorem 1.12 follows from Theorem 1.7 using Theorem 6.1. Indeed, assuming that E is hard for exponential size nondeterministic circuits, for every sufficiently large s, every $\ell \geq \log s$, and every $\frac{1}{2^s} \leq \delta \leq \frac{1}{s}$, we can set $\delta' = \frac{\delta}{4 \cdot s \cdot 2^\ell}$, and $s' = s^2$ and apply Theorem 1.7 to obtain a multiplicative $(\frac{1}{s'}, \delta')$ PRG $G : \{0,1\}^r \to \{0,1\}^s$ for

$$r = O(\log \frac{1}{\delta'}) = O(\ell + \log \frac{1}{\delta}).$$

We stress that in order to get the correct dependence on ℓ (namely, $r = O(\ell)$) in the multiplicative nb-PRG, it is imperative to have the correct dependence of $r = O(\log \frac{1}{\delta})$ in the multiplicative PRG.

Indeed, previous work by Artemenko et al. [AIKS16] constructed (boolean) multiplicative PRGs. However, that paper relies on a significantly stronger hardness assumption, and in addition only achieved $r = O(\log \frac{1}{\delta})^2$ for the boolean multiplicative PRG. As a consequence, Artemenko et al. [AIKS16] could not use the argument of Theorem 6.1 directly, as this would yield an nb-PRG with seed length $r \geq \ell^2$.

Instead, they used a significantly more complicated construction of multiplicative nb-PRGs from multiplicative nb-PRGs. One of the costs of this more complicated construction is that they end up needing to assume the assumption that E is hard for exponential size Σ_6-circuits.

Remark 6.2 (Multiplicative nb-PRG for nondeterministic circuits)**.** *When deriving Theorem 1.12 from Theorem 1.7 above, we did not use the fact that the PRG constructed in Theorem 1.7 fools nondeterministic circuits. The argument only requires that the (boolean) multiplicative PRG to fool deterministic circuits.*

It is natural to ask whether Theorem 6.1 can be extended to give a multiplicative nb-PRG that fools nondeterministic circuits *when starting from a (boolean) multiplicative PRG that fools* nondeterministic circuits.

Here, there is a definitional issue, as it is not clear what a nondetreministic nonboolean circuit is. (The concept of nondeterministic computation is defined only for procedures that have boolean output). Nevertheless, if one takes "single valued nondeterministic circuits" (see e.g. [MV05] for a definition) as the nonboolean generalization of nondeterministic circuit, then the proof of Theorem 6.1 gives a version where in the assumption one assumes a (boolean) PRG that fools nondeterministic circuits, and in the conclusion one obtains an nb-PRG that fools single valued nondeterministic circuits.

The proof of Theorem 6.1 is given below, and is very similar to an analogous argument that was given in [AIKS16].

Proof of Theorem 6.1. Let $C : \{0,1\}^m \to \{0,1\}^\ell$ be a size s' circuit. For every $v \in \{0,1\}^\ell$, we define the circuit $C_v : \{0,1\}^m \to \{0,1\}$ that given $z \in \{0,1\}^m$ answers one if and only if $C(z) = v$. By definition, for every $v \in \{0,1\}^\ell$, C_v is a size $s = s' + O(\ell)$ circuit. We define:

$$H = \left\{ v \in \{0,1\}^\ell : \Pr[C(U_m) = v] \geq \frac{\delta}{\epsilon} \right\}.$$

This gives that for every $v \in H$,

$$\begin{aligned}
\Pr[C(G(U_r)) = v] &= Pr[C_v(G(U_r)) = 1] \\
&\leq e^\epsilon \cdot \Pr[C_v(U_m) = 1] + \delta \\
&\leq (1 + 3\epsilon) \cdot \Pr[C_v(U_m) = 1] + \delta \\
&\leq (1 + 3\epsilon) \cdot \Pr[C(U_m) = v] + \epsilon \cdot \Pr[C(U_m) = v] \\
&= (1 + 4\epsilon) \cdot \Pr[C(U_m) = v] \\
&\leq e^{4\epsilon} \cdot \Pr[C(U_m) = v].
\end{aligned}$$

Similarly, for every $v \in \{0,1\}^\ell \setminus H$,

$$\begin{aligned}
\Pr[C(G(U_r)) = v] &= \Pr[C_v(G(U_r)) = 1] \\
&\leq e^\epsilon \cdot \Pr[C_v(U_m) = 1] + \delta \\
&\leq 3 \cdot \frac{\delta}{\epsilon} + \delta \\
&\leq 4 \cdot \frac{\delta}{\epsilon}.
\end{aligned}$$

For every function $D : \{0,1\}^\ell \to \{0,1\}$, we define

$$H_D = \left\{ v \in \{0,1\}^\ell : v \in H \text{ and } D(v) = 1 \right\}.$$

$$L_D = \left\{ v \in \{0,1\}^\ell : v \notin H \text{ and } D(v) = 1 \right\}.$$

Finally, we compute:

$$\begin{aligned}\Pr[D(C(G(U_r))) = 1] &= \sum_{v \in H_D} \Pr[C(G(U_r)) = v] + \sum_{v \in L_D} \Pr[C(G(U_r)) = v] \\ &\leq \sum_{v \in H_D} e^{4\epsilon} \cdot \Pr[C(U_m) = v] + \sum_{v \in L_D} 4 \cdot \frac{\delta}{\epsilon} \\ &\leq \sum_{v : D(v)=1} e^{4\epsilon} \cdot \Pr[C(U_m) = v] + \sum_{v \in \{0,1\}^\ell} 4 \cdot \frac{\delta}{\epsilon} \\ &\leq e^{4\epsilon} \cdot \Pr[D(C(U_m)) = 1] + 4 \cdot 2^\ell \cdot \frac{\delta}{\epsilon}.\end{aligned}$$

This gives that D is a $(\ell, 4\epsilon, \frac{4 \cdot 2^\ell \cdot \delta}{\epsilon})$-multiplicative nb-PRG. □

6.2 Applications of Multiplicative nb-PRGs

Multiplicative nb-PRGs are defined so that whenever one has a polynomial time algorithm Sample such that for every sufficiently large m, $P = \mathsf{Sample}(U_m)$ is a distribution on $\ell \ll m$ bits (a good choice to keep in mind is $\ell = m^\alpha$ for some constant $0 < \alpha < 1$) then the distribution $P' = \mathsf{Sample}(G(U_r))$ (obtained using the multiplicative nb-PRG that is the consequence of Theorem 1.12) can be sampled in time $\mathsf{poly}(m)$ using only $r = O(\ell + \log(1/\delta))$ random bits, and is similar to P, in the sense that it "preserves small probabilities". More specifically, that for any event $A \subseteq \{0,1\}^\ell$,

$$\Pr[P' \in A] \leq e^{\frac{1}{m}} \cdot \Pr[P \in A] + \delta \leq 2 \cdot \Pr[P \in A] + \delta.$$

The property that small probabilities are preserved is attractive in cryptographic applications, where one often wants that the probability that the adversary breaks the security is negligible.

nb-PRGs in cryptographic applications. An obvious difficulty when trying to apply PRGs from the Nisan-Wigderson setting in a cryptographic application is that such PRGs are *not cryptographic*, meaning that the time allowed to the adversary of the PRG is *smaller* than the time it takes to run the PRG. This is in contrast to *cryptographic* PRGs, in which the time allowed to the adversary of the PRG is *larger* than the time it takes to run the PRG. (Typically, the PRG runs in fixed polynomial time and is required to fool adversaries that run in arbitrary polynomial time).

It is this property that allows cryptographic PRGs to be used in cryptographic applications where honest parties can run cryptographic PRG, and expect to fool adversaries that run in larger time.

The notion of nb-PRG that we are considering is set up against an adversary that runs in two steps:

- At the first step, a computationally bounded $C : \{0,1\}^m \to \{0,1\}^\ell$ (which does not have sufficient time to run the PRG) receives a string $z \in \{0,1\}^m$.
- At the second step, an unbounded adversary $D : \{0,1\}^\ell \to \{0,1\}$ receives the string $C(z)$.

It is required that this "combination of procedures" does not distinguish a pseudorandom distribution from the uniform distribution (and in the multiplicative case, this is with respect to a multiplicative relation).

Past work on nb-PRGs [DI06, AS14, AIKS16] identified several settings in which this two step guarantee can be used in cryptographic applications. We will now briefly survey two such applications.

Reducing the input length of OWFs with large input length [AIKS16]. Consider a OWF (one way function) that has input length that is much larger than the output length. For concreteness, assume that the only OWF that we have is a function $f : \{0,1\}^m \to \{0,1\}^\ell$ where $m = \ell^{10}$. We would like to reduce the input

length of f (as this often reduces communication in cryptographic protocols) while "preserving the hardness". This can be done using a suitable multiplicative nb-PRG, $G : \{0,1\}^{r=O(\ell)} \to \{0,1\}^m$ by defining:

$$f'(s) = f(G(s)).$$

This gives a construction of a function $f' : \{0,1\}^{O(\ell} \to \{0,1\}^\ell$. At first glance, this seems problematic as the nb-PRGs that we consider are in the Nisan-Wigderson setting, and do not fool the adversaries that are trying to break the OWF. The key observation is that for f' to be a OWF, we do not need G to fool an adversary C (that is trying to break f) but rather an honest party C (that is trying to compute f).

More specifically, it is sufficient to set G to fool the circuit $C = f$ of fixed polynomial size (namely, the "easy direction" of f). This is because, then we have that the distribution $P' = f'(U_r) = f(G(U_r))$ is similar to $P = f(U_m)$, in the sense that it preserves small probabilities. In particular, if we started from the assumption that an adversary D can produce an input in $f^{-1}(P)$ with at most negligible probability, then we can conclude that the same holds when D receives $P' = f'(U_r)$ on input.

Note that this argument is very general and applies to any class of adversaries D (as long as the initial function f was secure against this class). Furthermore, note that for this application, it is crucial to "preserve small probabilities" and one requires a multiplicative nb-PRG, rather than a (standard) nb-PRG to obtain the negligible probability of inversion that is expected for OWFs.

Finally, note that the multiplicative nb-PRG of Theorem 1.12 is precisely what is required in such an application (meaning that this can be done under the hardness assumption in Theorem 1.12).

Saving randomness in information theoretic secure MPC for semi-honest parties [DI06]. In the application discussed above, the nb-PRG was set up to fool honest parties C, rather than adversaries. In protocols for secure multiparty computation (MPC) it is often the case that one designs a protocol against adversaries that are "semi-honest". Namely, they follow the computation prescribed in the protocol, but later on, may perform additional (possibly unbounded) computation trying to learn the secrets of the honest parties.

Loosely speaking, such an adversary corresponds with the two step process of C and D described above in the following sense: During the protocol, the adversary runs a fixed polynomial time computation (determined by the protocol) as a function of its secret, randomness, and communication. This means that the entire system (of honest parties and semi-honest adversaries) can be viewed as a circuit C of fixed polynomial size (that is hardwired with the secrets) takes as input the random strings of each party, and produces the prescribed communication and output to each party. Later on, some coalition of bad parties may choose to examine their communications and outputs (possibly using some unbounded computation D in order to steal the secrets of the honest parties).

Dubrov and Ishai [DI06] formally specify such a scenario, and (using the intuition described above) show that (standard) nb-PRGs can be used to reduce the randomness required by all parties to the total length of communication (which we will denote by ℓ). Loosely speaking, this is achieved by making parties use an nb-PRG to stretch an $r = O(\ell)$ long random string, into an m bit string, which each party uses as its random coin tosses. The correctness and security of this modified protocol follows by using the intuition described above. We refer the reader to [DI06] for precise details.

As this is a cryptographic protocol, it is natural to try and achieve "negligible" probability of adversaries to cheat. This is where multiplicative nb-PRGs come in. Indeed, using multiplicative nb-PRGs one can show that if some event (say that the bad parties steal the credit card number of some honest party) occurs with "negligible" probability in the original protocol, then it also holds with "negligible" probability in the modified protocol.

We remark however, that for this application, nb-PRGs w.r.t. $\overset{m}{\sim}_{(\epsilon,\delta)}$ do not seem to suffice, and one requires nb-PRGs w.r.t. (the double sided multiplicative relation) $\overset{md}{\sim}_{(\epsilon,\delta)}$, defined by:

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

Fortunately, it is easy to see that both Theorem 1.7 and Theorem 1.12 can give PRGs w.r.t. this relation, under the assumption that E is hard for exponential size Σ_2-circuits.[17] We defer the details to a later version.

7 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 [Sha25, BSS25] 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 1.5 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 assumptions? We remark that it is possible to bypass this limitations if one is willing to assume the very strong assumption that E is hard for exponential size PSPACE-circuits (as also mentioned in [AIKS16]). 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 [AASY15], 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

[AASY15] 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.

[AIKS16] 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.

[AK02] V. Arvind and J. Köbler. New lowness results for ZPP^{NP} and other complexity classes. *J. Comput. Syst. Sci.*, 65(2):257–277, 2002.

[AS14] 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.

[BDL22] 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.

[17] Loosely speaking, this is because in the proof of Theorem 1.7 we use the "set lower bound" protocol of Goldwasser and Sipser [GS86], which we describe in Section 3.8. An analogous "set upper bound" protocol is not known for AM protocols, or nondeterministic circuits. However, it can be done in $\mathsf{BPP}^{\mathsf{NP}}$ (by classical results on approximate counting of NP-witnesses [Sto83, Sip83, JVV86]) and using this protocol in the proof, allows us to get "both sides" of the relation $\overset{md}{\sim}_{(\epsilon,\delta)}$, at the cost of assuming a stronger hardness assumption (to account for the additional NP oracle).

[BGDM23] 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.

[BOV07] B. Barak, S. J. Ong, and S. P. Vadhan. Derandomization in cryptography. *SIAM J. Comput.*, 37(2):380–400, 2007.

[BR94] M. Bellare and J. Rompel. Randomness-efficient oblivious sampling. In *35th Annual Symposium on Foundations of Computer Science*, pages 276–287, 1994.

[BSS24] 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.

[BSS25] M. Ball, R. Shaltiel, and J. Silbak. Extractor for samplable distributions with low min-entropy. *To appear in STOC*, 2025.

[BV17] 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.

[CLO+23] 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.

[CT22] 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.

[DI06] 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.

[DMOZ22] D. Doron, D. Moshkovitz, J. Oh, and D. Zuckerman. Nearly optimal pseudorandomness from hardness. *J. ACM*, 69(6):43:1–43:55, 2022.

[Dru13] 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.

[GM84] 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.

[GS86] 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.

[GST03] 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.

[GSV18] 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.

[GW02] O. Goldreich and A. Wigderson. Derandomization that is rarely wrong from short advice that is typically good. In *APPROX-RANDOM*, pages 209–223, 2002.

[HNY17] 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.

[ILL89] R. Impagliazzo, L. A. Levin, and M. Luby. Pseudo-random generation from one-way functions (extended abstracts). In *Proceedings of the 21st Annual ACM Symposium on Theory of Computing*, pages 12–24, 1989.

[ISW99] R. Impagliazzo, R. Shaltiel, and A. Wigderson. Near-optimal conversion of hardness into pseudo-randomness. In *FOCS*, pages 181–190, 1999.

[ISW06] R. Impagliazzo, R. Shaltiel, and A. Wigderson. Reducing the seed length in the nisan-wigderson generator. *Combinatorica*, 26(6):647–681, 2006.

[IW97] R. Impagliazzo and A. Wigderson. $P = BPP$ if E requires exponential circuits: Derandomizing the XOR lemma. In *STOC*, pages 220–229, 1997.

[JVV86] M. Jerrum, L. G. Valiant, and V. V. Vazirani. Random generation of combinatorial structures from a uniform distribution. *Theor. Comput. Sci.*, 43:169–188, 1986.

[KvM02] 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.

[MV05] P. Bro Miltersen and N. V. Vinodchandran. Derandomizing arthur-merlin games using hitting sets. *Computational Complexity*, 14(3):256–279, 2005.

[NW94] N. Nisan and A. Wigderson. Hardness vs. randomness. *JCSS: Journal of Computer and System Sciences*, 49, 1994.

[Sha25] 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.

[Sip83] M. Sipser. A complexity theoretic approach to randomness. In *STOC*, pages 330–335, 1983.

[SS24] 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.

[Sto83] L. J. Stockmeyer. The complexity of approximate counting. In *STOC*, pages 118–126, 1983.

[STV01] M. Sudan, L. Trevisan, and S. P. Vadhan. Pseudorandom generators without the xor lemma. *J. Comput. Syst. Sci.*, 62(2):236–266, 2001.

[SU05] R. Shaltiel and C. Umans. Simple extractors for all min-entropies and a new pseudorandom generator. *J. ACM*, 52(2):172–216, 2005.

[SU06] R. Shaltiel and C. Umans. Pseudorandomness for approximate counting and sampling. *Computational Complexity*, 15(4):298–341, 2006.

[SU09] R. Shaltiel and C. Umans. Low-end uniform hardness versus randomness tradeoffs for am. *SIAM J. Comput.*, 39(3):1006–1037, 2009.

[Sud97] M. Sudan. Decoding of Reed Solomon codes beyond the error-correction bound. *Journal of Complexity*, 13, 1997.

[TV00] L. Trevisan and S. P. Vadhan. Extracting randomness from samplable distributions. In *41st Annual Symposium on Foundations of Computer Science*, pages 32–42, 2000.

[Uma03] C. Umans. Pseudo-random generators for all hardnesses. *Journal of Computer and System Sciences*, 67:419–440, 2003.

ECCC ISSN 1433-8092

https://eccc.weizmann.ac.il