Electronic Colloquium on Computational Complexity, Report No. 107 (2025)

Extractors for Samplable Distributions from the Two-Source Extractor Recipe

Justin Oh[*] Ronen Shaltiel[†]

Abstract

Trevisan and Vadhan [TV00] first constructed seedless extractors for distributions samplable by poly-size circuits under the very strong complexity theoretic hardness assumption that $\mathsf{E} = \mathsf{DTIME}(2^{O(n)})$ is hard for exponential size circuits with oracle access to Σ_6^{P}. Their construction works when the distribution has large min-entropy $k = (1-\gamma) \cdot n$, for small constant $\gamma > 0$.

Recent works build on the approach of [TV00]. [BDSGM23] obtain the same result under the weaker assumption that there is a problem in E that is hard for exponential size nondeterministic circuits. [BSS25] and [Sha25] improve the min-entropy threshold to $k = n^{1-\gamma}$ and $k = n^{\Omega(1)}$ respectively, reinstating oracle access to Σ_i^{P} for some i in the assumption.

We introduce a new approach, inspired by constructions of two-source extractors [CZ16, BDT19], using a new (and incomparable) hardness assumption that only involves *deterministic* circuits. Our approach reduces the task of constructing extractors for samplable distributions to constructing explicit non-malleable extractors with short seed-length.

Our new assumption has the same flavor as the classic assumption of [IW97] that E is hard for exponential size circuits, and similar to one recently considered in the context of fast derandomization [CT21]. Specifically, we assume that there is a constant $0 < \alpha < 1$ such that for every constant $C_{\mathsf{hard}} \geq 1$, there exists a constant C_{easy} and a problem in $\mathsf{DTIME}(2^{C_{\mathsf{easy}} \cdot n})$ that is not in $\mathsf{DTIME}(2^{C_{\mathsf{hard}} \cdot n})/2^{\alpha \cdot n}$. The key feature here is that we allow the "adversary" to run in time larger than 2^n while still only using less than 2^n bits of nonuniformity. Under this assumption, we use currently known constructions of non-malleable extractors to get the following results:

- An extractor for samplable distributions with min-entropy k slightly larger than $n/2$. This is the first construction of any such extractor under an assumption that does not give the adversary nondeterminism.
- An extractor for samplable distributions with min-entropy $k = O(\log n \cdot \log\log n)$ also follows if, in addition to the new assumption, we also have the aforementioned assumption of [BDSGM23], namely, that E is hard for exponential size nondeterministic circuits. This is the first construction in the regime of $k \leq \mathrm{poly}\log n$ under *any* assumption.

We also show that future improvements of the seed length of the best current non-malleable extractors [Li17] would imply the second result without the additional assumption.

Our key observation is that when a given source is samplable, the set of "bad" seeds to a non-malleable extractor is efficiently recognizable. We utilize this observation to show that in the constructions of two-source extractors in [CZ16, BDT19], we can hope to replace the "second source" with (the truth table of) a sufficiently hard function. Thus our work reveals an unexpected connection between two-source extractors and extractors for samplable distributions, similar to Trevisan's connection between extractors and PRGs "in the other direction."

[*]University of Haifa. `sung-ho.oh@fulbrightmail.org`. Supported by a Fulbright Postdoctoral Fellowship and ISF grant 1006/23.

[†]University of Haifa. `ronen@cs.haifa.ac.il`. Supported by ISF grant 1006/23 and also co-funded by the European Union (ERC, NFITSC, 101097959).

ISSN 1433-8092

Contents

1 Introduction

A natural model of the sources of randomness that may occur in nature, or that are otherwise available to a computer, are distributions samplable by some efficient model of computation, such as polynomial size circuits. Trevisan and Vadhan [TV00] first considered the possibility of *seedless extraction* from such sources. They aimed to design a procedure that can convert such distributions into the uniform distribution, where it can be used in countless applications such as randomized algorithms and cryptography.

Definition 1.1. *A distribution X over $\{0,1\}^n$ is samplable by size s circuits, if there is a circuit $X: \{0,1\}^s \to \{0,1\}^n$ (we will use X to denote both for the distribution and the circuit) such that the distribution X is obtained as $X(U_s)$.*

A (seedless) (k,ε)-extractor for distributions samplable by circuits of size s, is a function:

$$\mathsf{Ext}\colon \{0,1\}^n \to \{0,1\}^m$$

such that for every distribution X over $\{0,1\}^n$ that is samplable by size s circuits, and has $H_\infty(X) \geq k$, we have that $\mathsf{Ext}(X)$ is ε-close to U_m.[1].

[TV00] show that if Ext is an extractor for samplable distributions computable in time s, then Ext cannot be computed by circuit of size slightly smaller than s. Given our current inability to prove circuit lower bounds, this means that explicit constructions of extractors for distributions samplable by circuits of size $s = n^c$ (for any constant $c > 1$) require hardness assumptions. Furthermore, even with hardness assumptions, if we want such extractors to run in $\mathrm{poly}(n)$ time, then this polynomial must be larger than n^c. Trevisan and Vadhan [TV00] show how to construct such extractors under a hardness assumption for a certain generalization of nondeterministic circuits.

Hardness assumptions against various types of circuits. Recall that $\mathsf{E} = \mathsf{DTIME}(2^{O(n)})$. We say that "E is hard for exponential size circuits of type X" if there exist constants $0 < \alpha < C_{\mathsf{easy}}$, and a language L in $\mathsf{DTIME}(2^{C_{\mathsf{easy}} \cdot n})$, such that for every sufficiently large n, the characteristic function of L on inputs of length n cannot be computed by circuits of size $2^{\alpha n}$ of type X. See also formal statement in Definition 3.4.

The assumption that E is hard for exponential size (deterministic) circuits was used by the celebrated paper of Impagliazzo and Wigderson [IW97] to imply that BPP = P. The stronger assumption that E is hard for exponential size nondeterministic circuits (see Definition 3.2 for a formal definition of nondeterministic circuits) was introduced in the context of hardness vs. randomness tradeoffs for AM [KvM02, MV05, SU05, SU06].

In their seminal paper on extractors for samplable distributions, Trevisan and Vadhan [TV00] used this form of assumption for an even stronger "nondeterministic circuit class". More specifically, a Σ_i-circuit, is a circuit that in addition to the standard gates, is also allowed to use a special gate (with large fan-in) that solves the canonical complete language for the class Σ_i^{P} (namely, the i'th level of the polynomial time hierarchy). See Definition 3.2 for a formal definition of Σ_i-circuits.

Previous work on extractors for samplable distributions. The extractor of Trevisan and Vadhan [TV00] relies on the strong assumption that E is hard for exponential size Σ_6-circuits. Under this

[1]See Section 3.1 for various definitions such min-entropy, and statistical distance

hardness assumption, [TV00] showed that there exists a constant $\gamma > 0$ such that for every constant $c > 1$, there is a (k, ε)-extractor for distributions samplable by size n^c circuits, where $k = (1 - \gamma) \cdot n$, $\varepsilon = n^{-c}$ and the extractor runs in time $\mathrm{poly}(n^c)$. The seminal result of Trevisan and Vadhan leaves two things to be desired.

- The min-entropy of X is required to be $k = (1 - \gamma) \cdot n$ for some small $\gamma > 0$. One would hope that k can be as small as what [TV00] prove is possible nonexplicitly. For example, for one output bit $m = 1$, constant $\varepsilon > 0$, and X samplable by circuits of size n^c they show that k can be $O_c(\log n)$.
- One could hope that the hardness assumption used for the construction is minimal, or at least not mention nondeterministic or Σ_6-circuits.

Subsequent work on extractors for samplable distributions aims to address these two issues. Ball, Dachman-Soled, Goldin, and Mutreja [BDSGM23] construct extractors with the same parameters as in Trevisan and Vadhan [TV00] under the weaker assumption that E is hard for exponential size nondeterministic circuits. Ball, Shaltiel and Silbak [BSS25] improved the min-entropy parameter to $k = n^{1-\gamma}$ for some constant $\gamma > 0$. This is achieved under the assumption that E is hard for exponential size Σ_5-circuits. Shaltiel [Sha25] further improved the min-entropy parameter to $k = n^{1/i}$ under the assumption that E is hard for exponential size Σ_{i+3}-circuits (giving a tradeoff between min-entropy and hardness assumption). See Table 1 for a summary of previous work.

We note that *all previous works build off of the core ideas of [TV00] and build on hardness assumptions against nondeterministic circuits*. It was pointed out in [Sha24] that all existing proofs yield extractors for a class of distributions that is richer than the class of samplable distributions, and that such extractors *imply* lower bounds against nondeterministic circuits (see [Sha24] for a precise formulation). Consequently, it seems that the current approach to construct extractors for samplable distributions cannot avoid hardness assumptions against nondeterministic circuits.

1.1 Our Results

We introduce a new approach for constructing extractors for samplable distributions (that is not based on the seminal work of Trevisan and Vadhan [TV00]). Instead, our approach is inspired by, and builds on, the recent breakthrough constructions of two-source extractors by Chattopadhyay and Zuckerman [CZ16] and the later improvement by Ben-Aroya, Doron and Ta-Shma [BDT19].

Using this new approach one can aim to achieve extractors for samplable distributions for very low min-entropy parameter k. We obtain extractors for $k = O(\log n \cdot \log\log n)$ which comes very close to the min-entropy threshold $k = O_c(\log n)$ obtained by a nonexplicit argument [TV00]. Our approach relies on a hardness assumption that is incomparable to those used in previous work on extractors for samplable distributions, and which we now introduce.

1.1.1 A New Hardness Assumption for Extractors for Samplable Distributions

We introduce a new hardness assumption of a flavor that was not considered in previous work on extractors for samplable distributions. The assumption is incomparable previous the previous ones. Chen and Tell [CT21] considered stronger version of the assumption in the context of fast derandomization. The assumption considers a circuit model where running time is separated from the amount of nonuniformity.

Separating time and advice in the definition of circuits. We consider a circuit model which separates running time from advice. More specifically, we will consider nonuniform procedures that run in time t, and use $a \leq t$ bits of advice. (A formal definition of this circuit model is given in Definition 3.3). Standard (deterministic) circuits of a certain size s correspond to the case where $t = a = s$, and we will be interested in a setting in which $t > 2^n$ and $a < 2^n$. (Obviously allowing $a > 2^n$ makes such a circuit all-powerful, and we do not allow this). More specifically, we will consider procedures which on input length n, are allowed to run in time $2^{C_{\mathsf{hard}} \cdot n}$ (for some constant $C_{\mathsf{hard}} > 1$) and use $a = 2^{\alpha \cdot n}$ bits of advice, for a constant $0 < \alpha < 1$. We will scale the assumption of [IW97] against such procedures, as follows:

New hardness assumption against circuits with $2^{O(n)}$ time and $2^{\Omega(n)}$ advice. We say that "E is hard for large exponential time with exponential advice" if there exists a constant $\alpha > 0$, such that for every constant $C_{\mathsf{hard}} > 1$, there exists a constant $C_{\mathsf{easy}} > C_{\mathsf{hard}}$, and a language L in $\mathsf{DTIME}(2^{C_{\mathsf{easy}} \cdot n})$, such that for every sufficiently large n, the characteristic function of L on inputs of length n cannot be computed by nonuniform procedures that run in time $2^{C_{\mathsf{hard}} \cdot n}$ and use $2^{\alpha \cdot n}$ bits of advice. See also formal statement in Definition 3.5.

To the best of our knowledge, hardness assumptions of this form (separating time from advice) were first introduced by Chen and Tell [CT21] (see also [CT23]) in the context of fast derandomization. The assumption that we introduce is quantitatively weaker than the assumptions introduced previously by [CT21].[2]

Comparing Hardness Assumptions. The assumption that E is hard for large exponential time with exponential advice is incomparable to the previous assumptions used in the context of extractors for samplable distributions. That is, it is not known whether "E is hard for large exponential time with exponential advice" implies "E is hard for exponential size non-deterministic circuits (or Σ_i-circuits)" or vice versa. The new assumption is stronger as it allows the "adversary" a polynomial increase in time $2^{C_{\mathsf{hard}} \cdot n} = \mathrm{poly}(2^{\alpha \cdot n})$, and weaker in the sense that it does not give the "adversary" nondeterminism. A discussion on potential instantiations of this assumption appears in Section 9.1.

1.1.2 A Reduction from Extractors for Samplable Distributions to Non-malleable Extractors

Following the two-source extractor constructions of [CZ16, BDT19] our construction relies on (seeded) non-malleable extractors. These extractors (defined by Dodis and Wichs [DW09]) are a variant of (standard) seeded extractors with an additional property. A precise formal definition appears in Definition 3.16, however, the precise formal definition is not necessary to describe our results for readers familiar with (standard) seeded extractors.

We say that an explicit construction of non-malleable extractors is "*nice*" if the seed-length is $d = O(\log \frac{n}{\varepsilon_{\mathsf{nm}}})$ (at least when $\varepsilon_{\mathsf{nm}}$ is not too small). A precise formal definition of a "nice construction of non-malleable extractors" appears in Definition 3.17. Our main contribution is the following theorem (stated more formally in Theorem 7.1).

[2]The hardness assumption considered in [CT21] considers a case where α is very close to one, whereas we allow α to be close to zero. This makes our assumption weaker. The hardness assumption in [CT21] also requires that C_{easy} is only slightly larger than C_{hard}, whereas we allow arbitrary dependence on of C_{easy} on C_{hard}. Once again, this makes our assumption weaker. However, the two assumptions are of the same flavor apart from these quantitative differences.

Theorem 1.2 (Informal). *If*

- E *is hard for large exponential time with exponential advice, and*
- *there is a nice construction of non-malleable extractors for min-entropy k,*

then for every constant $c \geq 1$, every constant $\varepsilon > 0$, and every sufficiently large n, there is a (k, ε)-extractor $\mathsf{Ext} : \{0,1\}^n \to \{0,1\}$ *for distributions samplable by circuits of size n^c. Furthermore,* Ext *runs in time* $\mathrm{poly}(n^c)$.

Plugging in a nice explicit construction of non-malleable extractors by Cohen, Raz and Segev [CRS14] (which works for min-entropy k that is slightly larger than $\frac{n}{2}$) we obtain the following result.

Theorem 1.3 (Extractors for samplable distributions with min-entropy k slightly larger than $n/2$). *If* E *is hard for large exponential time with exponential advice, then for every constants $c \geq 1$, and $\varepsilon, \gamma > 0$ and for every sufficiently large n, there is a $(k = (\frac{1}{2} + \gamma) \cdot n, \varepsilon)$-extractor* $\mathsf{Ext} : \{0,1\}^n \to \{0,1\}$ *for distributions samplable by circuits of size n^c. Furthermore,* Ext *runs in time* $\mathrm{poly}(n^c)$.

Recent constructions of non-malleable extractors ([CGL16, Coh16b, Coh16a, CL16] culminating in [Li17]) achieve $d = O(\log n)$ for very low min-entropy $k = \mathrm{poly}\log n$. Indeed, these non-malleable extractors are a key ingredient in the aforementioned constructions of two-source extractors (that inherit this min-entropy [CZ16, BDT19]).

Unfortunately, at this point in time, these explicit constructions are not "sufficiently nice" for our purposes, as the dependence on $\varepsilon_{\mathsf{nm}}$ is "slightly off". More specifically, instead of seed length $d = O(\log \frac{n}{\varepsilon_{\mathsf{nm}}}) = O(\log n + \log \frac{1}{\varepsilon_{\mathsf{nm}}})$, these construction require seed length at least $d \geq \log n + \log \frac{1}{\varepsilon_{\mathsf{nm}}} \cdot \log\log \frac{1}{\varepsilon_{\mathsf{nm}}}$

By Theorem 1.2, if these constructions are improved to give "nice non-malleable extractors" (which is a well known open problem) then our approach immediately produces extractors for samplable distributions for the very low min-entropy of $k = O(\log n \cdot \log\log n)$.

Our next result bypasses the lack of nice constructions of non-malleable extractors for low min-entropy, by showing that we can obtain extractors for samplable distributions for min-entropy $k = O(\log n \cdot \log\log n)$ with current non-malleable extractors. However, to achieve this, in addition to the (new) assumption that E is hard for large exponential time with exponential advice, we also assume the weakest assumption that was previously used in the context of extractors for samplable distributions, namely that E is hard for exponential size nondeterministic circuits. This is stated in the theorem below.

Theorem 1.4 (Extractors for samplable distributions with min-entropy $k = O(\log n \cdot \log\log n)$). *If*

- E *is hard for large exponential time with exponential advice, and*
- E *is hard for exponential size nondeterministic circuits*

then for every constant $c \geq 1$, every constant $\varepsilon > 0$, and every sufficiently large n, there is a $(k = O(\log n \cdot \log\log n), \varepsilon)$-extractor $\mathsf{Ext} : \{0,1\}^n \to \{0,1\}$ *for distributions samplable by circuits of size n^c. Furthermore,* Ext *runs in time* $\mathrm{poly}(n^c)$.

As we explain in Section 8 below, this result follows as we can use the additional hardness assumption to convert current non-malleable extractors into *nice* non-malleable extractors (in the special case that the source is efficiently samplable, which is sufficient for our purposes). Nevertheless, we stress again that we should expect to get nice non-malleable extractors with low min-entropy without additional hardness assumptions, and such a result would get rid of the second assumption.

1.2 Comparison to Previous Work on Extractors For Samplable Distributions

Table 1: Comparison of constructions of extractors for distributions samplable by size n^c circuits. In all results $\gamma > 0$ is some small constant, and we ignore technicalities like "infinitely often".

Work	Min-entropy k	Assumption	Comments
[TV00]	$(1-\gamma)n$	E is hard for exponential size Σ_6-circuits: $\exists 0 < \alpha < 1 < C_{\text{easy}}$ s.t. $\mathsf{DTIME}(2^{C_{\text{easy}}\cdot n}) \not\subseteq \mathsf{SIZE}^{\Sigma_6^{\mathsf{P}}}(2^{\alpha\cdot n})$	$m \approx k$, $\varepsilon = n^{-c}$
[BDSGM23]	$(1-\gamma)n$	E is hard for exponential size nondeterministic circuits: $\exists 0 < \alpha < 1 < C_{\text{easy}}$ s.t. $\mathsf{DTIME}(2^{C_{\text{easy}}\cdot n}) \not\subseteq \mathsf{NSIZE}(2^{\alpha\cdot n})$	$m \approx k$, $\varepsilon = n^{-c}$
[BSS25]	$n^{1-\gamma}$	E is hard for exponential size Σ_5-circuits: $\exists 0 < \alpha < 1 < C_{\text{easy}}$ s.t. $\mathsf{DTIME}(2^{C_{\text{easy}}\cdot n}) \not\subseteq \mathsf{SIZE}^{\Sigma_5^{\mathsf{P}}}(2^{\alpha\cdot n})$	$m \approx k$, $\varepsilon = n^{-c}$
[Sha25]	$n^{1/i}$	E is hard for exponential size Σ_{i+3}-circuits: $\exists 0 < \alpha < 1 < C_{\text{easy}}$ s.t. $\mathsf{DTIME}(2^{C_{\text{easy}}\cdot n}) \not\subseteq \mathsf{SIZE}^{\Sigma_{i+3}^{\mathsf{P}}}(2^{\alpha\cdot n})$	$m \approx k$, $\varepsilon = n^{-c}$
This work Theorem 1.2	Any k	(i) E is hard for large exponential time with exponential advice: $\exists 0 < \alpha < 1,$ s.t. $\forall C_{\text{hard}} \geq 1,\ \exists C_{\text{easy}} > C_{\text{hard}}$ s.t. $\mathsf{DTIME}(2^{C_{\text{easy}}\cdot n}) \not\subseteq \mathsf{SIZE}(2^{C_{\text{hard}}\cdot n})/2^{\alpha\cdot n}$ (ii) There is a "nice non-malleable extractor" for min-entropy k.	$m = 1$, any constant $\varepsilon > 0$
This work Theorem 1.3	$(\frac{1}{2}+\gamma)\cdot n$	E is hard for large exponential time with exponential advice: $\exists 0 < \alpha < 1,$ s.t. $\forall C_{\text{hard}} \geq 1,\ \exists C_{\text{easy}} > C_{\text{hard}}$ s.t. $\mathsf{DTIME}(2^{C_{\text{easy}}\cdot n}) \not\subseteq \mathsf{SIZE}(2^{C_{\text{hard}}\cdot n})/2^{\alpha\cdot n}$	$m = 1$, any constant $\varepsilon > 0$
This work Theorem 1.4	$\log n \cdot \log\log n$	(i) E is hard for large exponential time with exponential advice: $\exists 0 < \alpha < 1,$ s.t. $\forall C_{\text{hard}} \geq 1,\ \exists C_{\text{easy}} > C_{\text{hard}}$ s.t. $\mathsf{DTIME}(2^{C_{\text{easy}}\cdot n}) \not\subseteq \mathsf{SIZE}(2^{C_{\text{hard}}\cdot n})/2^{\alpha\cdot n}$ (ii) E is hard for exponential size nondeterministic circuits: $\exists 0 < \alpha < 1 < C_{\text{easy}}$ s.t. $\mathsf{DTIME}(2^{C_{\text{easy}}\cdot n}) \not\subseteq \mathsf{NSIZE}(2^{\alpha\cdot n})$	$m = 1$, any constant $\varepsilon > 0$

A comparison of our results and the previous results appears in Table 1.

Advantages. In Section Section 1.1.1, we have already elaborated on the difference between our new hardness assumption (that E is hard for large exponential time with exponential advice) and compared it to the assumptions previously used in extractors for samplable distributions.

We stress again that as demonstrated in Theorem 1.3, our approach *already* gives extractors for samplable distributions for min-entropy $k = (\frac{1}{2} + \gamma) \cdot n$ (that is smaller than that achieved by Trevisan and Vadhan [TV00]) under a hardness assumption that *does not involve nondeterministic circuits*.

Furthermore, by Theorem 1.2 (and as explained earlier) if current explicit constructions of non-malleable extractors can be improved, then extractors for samplable distributions with very low min-entropy of $k = O(\log n \cdot \log\log n)$ (which is close to the $O_c(\log n)$ bound achieved by the probabilistic method [TV00]) immediately follow.

Moreover, as shown in Theorem 1.4 even with current non-malleable extractors (which are not sufficiently "nice" for our purposes) we can obtain extractors for samplable distributions with very low min-entropy of $k = O(\log n \cdot \log\log n)$, if in addition to the new hardness assumption, we also assume the weakest assumption assumed previously in this context. We stress that extractors for samplable distributions with such low min-entropy were not previously known under *any* assumption.

Drawbacks. Two drawbacks of our new results is that at the moment:

- We only obtain extractors for samplable distributions that output one bit, whereas previous work obtained extractors that output m bits, for m is that is very close to k.
- We only obtain extractors for samplable distributions that have constant error $\varepsilon > 0$, whereas previous work obtain extractors with error $\varepsilon = n^{-c}$, and sometimes even "multiplicative error" [Sha24, Sha25].

While these are significant drawbacks, we believe that these drawbacks are not inherent to the new approach, and in Section 2.5 we elaborate on potential directions to overcome these issues.

2 Technical Overview

In this section we provide a high level explanation of the proofs of Theorem 1.2, Theorem 1.3, and Theorem 1.4. In Section 2.1 we review the ideas behind the construction of two-source extractors in [CZ16], and we also explain the computational analogue of their technique as relevant to our problem. In Section 2.2, we discuss some additional technical details using the (computationally analogous) ideas of [BDT19] that are required to make our parameters work from the tools available to us. The two sections mentioned above are sufficient for Theorem 1.2 and Theorem 1.3. Finally, in Section 2.4, we show how to utilize recent constructions of non-malleable extractors that do not quite have the right seed length in order to achieve Theorem 1.4.

The later technical sections 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 The Solution at a High Level

We first give a high level overview of the construction of two-source extractors from [CZ16].

The Reduction from Two-Source Extractors to Non-Malleable Extractors. [CZ16] provides a construction of two source extractors using non-malleable extractors (see Definition 3.16 for a precise definition). We'll first discuss the high level points of this result, which entails first discussing the notion of a non-malleable extractor, and what a "bad" seed means in this context. Following the work of [CZ16], a $(t, k, \varepsilon_{\mathsf{nm}})$-non-malleable extractor

$$\mathsf{nmExt}\colon \{0,1\}^n \times \{0,1\}^d \to \{0,1\},$$

has the property that for any X with min-entropy k, for most seeds y, $\mathsf{nmExt}(X, y)$ close to uniform, even conditioned on the outcomes $\mathsf{nmExt}(X, y_1), \ldots, \mathsf{nmExt}(X, y_t)$ for any other choice of distinct seeds $y_1, \ldots, y_t$. A bit more formally, for $h_y(x) = \mathsf{nmExt}(x, y)$, and any $y_1, \ldots, y_t$ all unique and distinct from y:

$$\Delta\left((h_y(X), h_{y_1}(X), \ldots, h_{y_t}(X)); (U_1, h_{y_1}(X), \ldots, h_{y_t}(X))\right) \le \sqrt{\varepsilon_{\mathsf{nm}}}. \tag{1}$$

We say that a seed with this property is "good." The set of good and bad seeds depend on X, and for a non-malleable extractor, there are at most $\sqrt{\varepsilon_{\mathsf{nm}}} \cdot 2^d$ bad seeds. A precise formal statement appears in Theorem 3.21. Roughly speaking, with a bit more work, this implies that the distribution $Z = \{\mathsf{nmExt}(X, y)\}_{y \in \{0,1\}^d}$ on 2^d bits has the property that the restriction of Z onto the good coordinates (corresponding to good seeds) is $(\gamma = \sqrt{\varepsilon_{\mathsf{nm}}})$-almost t-wise independent; that is, further restricting to any subset of t good coordinates yields a distribution that is γ-close to U_t. There are at most $q = \sqrt{\varepsilon_{\mathsf{nm}}}$ bad coordinates. Such distributions are called (q, t, γ)-non-oblivious bit fixing sources (see Definition 3.13 for a formal definition).

Extractors for Non-Oblivious Bit Fixing Sources. The above observation is helpful, because it is possible to construct seedless extractors for such sources. This follows from the fact that there are such extractors when $\gamma = 0$; that is, when the restriction to good coordinates is *truly* t-wise independent. Using the fact that a γ-almost t-wise independent distribution on R bits is $\gamma \cdot R^t$-close to a (truly) t-wise independent distribution, and applying such an extractor on Z yields the final extractor, with an added error of $\gamma \cdot R^t$. Here lies the main caveat, as pointed out in [CZ16]. We said above, that $\gamma = \sqrt{\varepsilon_{\mathsf{nm}}}$ and $R = 2^d$. However, d and $\varepsilon_{\mathsf{nm}}$ are *related*. Indeed, by the known lower bound of [RT00], $d \ge \log \frac{n}{\varepsilon_{\mathsf{nm}}^2} - O(1)$. One can check that this means that $\gamma \cdot R^t$ is always greater than 1.

The key insight of [CZ16] is to instead pick a subset S of R coordinates containing roughly the same fraction of bad seeds. The restriction of coordinates to S is still a non-oblivious bit fixing source with a similar q. Furthermore, using a second independent source X_2, and using a seeded extractor, they can construct the subset so that its size R is unrelated to d, and therefore $\varepsilon_{\mathsf{nm}}$. Specifically given a sample $x_2 \sim X_2$, the subset S they construct is the image of $\mathsf{Ext}(x_2, \cdot)$ for an appropriate $\mathsf{Ext}\colon \{0,1\}^n \times \{0,1\}^r \to \{0,1\}^d$. Since γ and $R = 2^r$ are now "decoupled," they are free to take the error of the non-malleable extractor as sufficiently small as they please, in order to make $\gamma \cdot R^t \ll 1$ and they achieve their two source extractor result.

Constructing S in the Case of Samplable Distributions. Since our goal is to construct a deterministic extractor given the fact that X is samplable by a small circuit, we do not have access to a second independent source X_2. The key observation we make is that in this scenario, the set of bad seeds to a non-malleable extractor is *recognizable* in polynomial time.

Observation 1. *Suppose there is a PRG against polynomial size circuits (and computable in polynomial time). (Such a PRG can be obtained using the seminal work of Impagliazzo and Wigderson [IW97] under a hardness assumption, see Theorem 3.7 for a precise statement). If X is samplable by a circuit of size n^c, and $\mathsf{nmExt}\colon \{0,1\}^n \to \{0,1\}^d \to \{0,1\}$ is a non-malleable extractor for constant t, with $d = \log(n/\varepsilon_{\mathsf{nm}})$, then for any $\varepsilon_{\mathsf{nm}} = 1/\mathrm{poly}(n)$ there is an algorithm IsBad_X that recognizes the set of bad seeds to nmExt (with respect to X) in time $\mathrm{poly}(n)$.*

Indeed, an algorithm $\mathsf{IsBad}_X(y)$ as follows can decide for a given X and seed y, whether y is good for X:

- Enumerate all t-tuples of distinct seeds $y_1, \ldots, y_t$. There are $2^{dt} = \mathrm{poly}(n)$ such tuples.
- For each of the 2^{2^t} subsets $S' \subset \{0,1\}^t$, use the PRG to estimate the probability that $(h_y(X), h_{y_1}(X), \ldots, h_{y_t}(X))$ falls in S, and $(U_1, h_{y_1}(X), \ldots, h_{y_t}(X))$ falls in S'.
- If these values differ significantly for any $y_1, \ldots, y_t$ and any S, then Equation (1) is violated and y is a bad seed.

The key properties that allow IsBad_X to run in polynomial time is threefold: the first is that t is constant, the second is that $d = \log(n/\varepsilon_{\mathsf{nm}})$ (and $\varepsilon_{\mathsf{nm}} = 1/\mathrm{poly}(n)$), and the third is that $h_y(X)$ is a polynomial size circuit, and so we can execute the second step using the PRG from [IW97] to estimate the probabilities to within $\varepsilon_{\mathsf{nm}}$. See Section 5 for details.

Since IsBad_X runs in polynomial time, we can once again use a PRG (fooling IsBad_X) to construct a subset of seeds S of size R. More concretely, for an appropriate PRG $G\colon \{0,1\}^r \to \{0,1\}^d$, we take $S = \mathrm{Im}(G) \subset \{0,1\}^d$. The fraction of seeds within S that are bad is also $\approx \sqrt{\varepsilon_{\mathsf{nm}}}$.

A Crucial Step: Decoupling R from $\varepsilon_{\mathsf{nm}}$. A crucial property of the subset S, as discussed in the overview of [CZ16] above, is that its size R should not be related to the error of the non-malleable extractor $\varepsilon_{\mathsf{nm}}$ (or in other words not related to the seed length d). The solution we present here is one of the core conceptual contributions of this work.

Unfortunately, if S is the image of a PRG constructed as in [IW97], then R and $\varepsilon_{\mathsf{nm}}$ will be inherently related. Indeed, IsBad_X runs time in at least $2^{t \cdot d}$, and is therefore implementable by a circuit of roughly the same size. Using the standard constructions of PRGs from [IW97], fooling circuits of size s requires a seed length of $r \geq \log s \geq td \geq d$. Therefore, $R \geq 2^d$.

To get around this problem, we observe that we do not seek to fool all circuits of size equal to $\mathsf{SIZE}(\mathsf{IsBad}_X)$. In fact, the total number of tests we wish to fool is the total number of sources X that are samplable by a circuit of size n^c. A probabilistic argument shows that a PRG with seed length $r \approx c \cdot \log n$ suffices for this. Notice here, that now $R = 2^r$ does not depend on d, it only depends on the size of the sampling circuits we consider. What remains is to find the right hardness assumption that allows us to construct a PRG with such seed length that fools all IsBad_X.

We observe that IsBad_X is an algorithm that runs in time $\mathrm{poly}(n) \gg n^c$ but with only n^c bits of advice. That is, the only nonuniformity in these algorithms is the description of the circuit sampling X. Following an idea of [CT21], given a hardness assumption that separates time and advice, namely that E is hard for large exponential time with exponential advice (see Definition 3.5 for a precise statement), one can adapt the hardness vs randomness framework to construct the desired PRG. In particular, we can convert a truth table of length $\mathrm{poly}(n^c)$ of a function that is hard for algorithms using $\approx n^c$ bits of advice and time a much larger $\mathrm{poly}(n)$ into the desired PRG. Since the seed length of a PRG obtained by such a transformation is logarithmic in the length of the truth table, we are done.[3] The precise details are give in Section 4.

Observation 2. *Assuming* E *is hard for large exponential time and exponential advice, one can construct a PRG against* $\mathsf{IsBad}_\mathsf{X}$ *with seed length depending only on the size of the sampling circuit* n^c*, even if* $\mathsf{IsBad}_\mathsf{X}$ *runs in a much larger polynomial time (that depends on the seed length of the non-malleable extractor).*

As a final note, notice that IsBad_X requires approximating the difference between the two distributions in Equation (1) to within $\sqrt{\varepsilon_{\mathsf{nm}}}$, which in general takes $1/\varepsilon_{\mathsf{nm}}$ samples, and thus a similar amount of time. Therefore, it is crucial that we can generate these samples in uniform polynomial time, rather than nonuniformly, so that the amount of advice used by IsBad_X remains n^c, and so R does not depend on d. This is possible since our assumption enables us to compute the first PRG (the one taken from [IW97]) in uniform polynomial time.

A Final Bird's Eye View. Overall, the current suggestion for our construction of the extractor is as follows. We can take a non-malleable extractor $\mathsf{nmExt}\colon \{0,1\}^n \to \{0,1\}^d \to \{0,1\}$ with $d = O(\log n/\varepsilon_{\mathsf{nm}})$. As long as $\varepsilon_{\mathsf{nm}} = 1/\mathrm{poly}(n)$, the set of bad seeds to nmExt for any X samplable by a circuit of size n^c is recognizable in polynomial time. Then, we can construct our subset S of seeds to nmExt using a PRG $G : \{0,1\}^r \to \{0,1\}^d$ with seed length $r \approx c \log n$, by setting $S = \{G(y) : y \in \{0,1\}^r\}$. Thus $|S| = R \approx n^c$. On the other hand, we can take the error of the non-malleable extractor to be $\varepsilon_{\mathsf{nm}}$ sufficiently smaller than $R^{-t} \approx n^{-ct}$. Since t is constant, this means $\varepsilon_{\mathsf{nm}}$ only needs to polynomially small. Then overall, the error of the final extractor construction will be $\gamma R^t = \sqrt{\varepsilon_{\mathsf{nm}}} R^t \ll 1$.

2.2 Trouble with Parameters

So far, we've shown that whenever t is constant, we can construct a PRG that samples the set of bad seeds to nmExt. This in turn produces a non-oblivious bit fixing source for constant t. The extractor for non-oblivious bit fixing sources from [CZ16] requires $t = \mathrm{poly}\log n$. Since it is crucial for us that t remain constant, we follow the ideas of [BDT19] which enables the use of the function majority as an extractor for such non-oblivious bit fixing sources.

Extractors for Non-Oblivious Bit Fixing Sources with Constant t. So far, what we have shown is that we can construct a subset of coordinates $S \subset \{0,1\}^d$ such that the restriction of Z to S is a $(q, t, \sqrt{\varepsilon_{\mathsf{nm}}})$-non-oblivious bit fixing source for a constant t. As is the case in [CZ16, BDT19], there are two related questions now. The first question is what is q, and the second question is what function extracts from such a source with such a q and t. To the best of our knowledge, the only known function that can extract from a non-oblivious bit fixing source when t is constant is the majority function [Vio14]. However, to do so, majority on R bits requires $q \le \sqrt{R}$. That is, the total number of bad coordinates must be at most square root the total number of coordinates.

The fact that q must be so small (relative to the total number of coordinates R) raises challenges, and we follow the solution of [BDT19], which also uses majority. First, let us make a distinction between the error of the non-malleable extractor, $\varepsilon_{\mathsf{nm}}$ (which governs d), and the error of the PRG $\varepsilon_{\mathsf{PRG}}$. We did not explicitly discuss the error of the PRG in the previous discussion, as it wasn't a

[3] We note that our analysis of the hardness vs randomness framework when separating time and advice is easier than that of [CT21]. This is because we are not concerned with optimizing the seed length beyond the fact that it is $O(c \log n)$.

necessary detail until now. However, the seed length of the PRG is $r = O(\log \frac{n}{\varepsilon_{\mathsf{PRG}}})$ while the seed length of nmExt is $d = O(\log \frac{n}{\varepsilon_{\mathsf{nm}}})$. What we learned from the previous discussion, is that r and d can be unrelated, and in particular, for any choice of $\varepsilon_{\mathsf{PRG}}$ we are free to choose $\varepsilon_{\mathsf{nm}}$ as sufficiently smaller than $\varepsilon_{\mathsf{PRG}}$ as we please. Doing so does not affect the number of sample points R, although it does affect where they live, i.e. their length as strings in $\{0,1\}^d$.

We now discuss what settings of $\varepsilon_{\mathsf{nm}}$ and $\varepsilon_{\mathsf{PRG}}$ make sense. First we observe that since the fraction of bad seeds overall is $\sqrt{\varepsilon_{\mathsf{nm}}}$, the fraction of bad seeds in S is no more than $\sqrt{\varepsilon_{\mathsf{nm}}} + \varepsilon_{\mathsf{PRG}} \approx \varepsilon_{\mathsf{PRG}}$ for sufficiently small choice of $\varepsilon_{\mathsf{nm}}$. Therefore, the total number of bad seeds in S is at most $\varepsilon_{\mathsf{PRG}} \cdot R$. The question now, is whether $\varepsilon_{\mathsf{PRG}} \cdot R \leq \sqrt{R}$? Unfortunately, the answer is no. For a PRG with error $\varepsilon_{\mathsf{PRG}}$, the seed length should be $r \geq \log n + 2 \cdot \log 1/\varepsilon_{\mathsf{PRG}}$. A calculation confirms our fears that no choice of $\varepsilon_{\mathsf{PRG}}$ gives us what we want. We do note however, that if $r = O(\log n) + (1+\beta)\log 1/\varepsilon_{\mathsf{PRG}}$, for some $\beta < 1/2$, then there is a sufficently small choice of $\varepsilon_{\mathsf{PRG}} = 1/\mathrm{poly}(n)$ such that $\varepsilon_{\mathsf{PRG}} \cdot R < \sqrt{R}$ and this is what we aim to achieve.

2.3 The Solution of [BDT19], and Its Computational Analogue

Since [CZ16] obtains the subset of coordinates S by using a second source, X_2, and applying a seeded extractor $\mathsf{Ext}\colon \{0,1\}^n \times \{0,1\}^r \to \{0,1\}^d$ to it, the same problem discussed above arises in their technique. Namely, it is also true for seeded extractors that $r \geq \log n + 2 \cdot \log 1/\varepsilon$, and so $q > \sqrt{R}$. However, this is not really an issue for them, as they use a different function than majority, that can handle such a large q (albeit requiring superconstant t). Since keeping t constant is crucial for our argument, we follow the approach of [BDT19] in order to use majority (as they do).

Achieving the Right Dependence on $\varepsilon_{\mathsf{PRG}}$: Moving from Pseudorandomness to "Pseudoentropy." As [BDT19] analogously point out for the information theoretic case, we observe that the argument sketched in the previous section for why a PRG works, only uses the fact that the PRG fools a test with extremely small density $\sqrt{\varepsilon_{\mathsf{nm}}}$. Therefore, we don't need to fool such a test with a distribution computationally indistinguishable from the uniform distribution, it suffices to use one computationally indistinguishable from sufficiently high min-entropy. We elaborate on this idea now.

First, observe, that if the seed length of the non-malleable extractor is $c_{\mathsf{nm}} \cdot \log(n/\varepsilon_{\mathsf{nm}})$ for some constant c_{nm}, then the the total number of bad seeds is roughly $2^{(1-1/c_{\mathsf{nm}})d} = D^{1-\frac{1}{c_{\mathsf{nm}}}}$ (for sufficiently small choice of $\varepsilon_{\mathsf{nm}}$). Concretely, suppose the total number of bad seeds is $D^{.98}$. We claim that this implies that a generator producing a string with $.99d$ bits of *pseudoentropy* suffices for our purposes. Although we won't formally define such a notion here, the high level idea is that now, the set S has the property that the fraction of seeds in S that fall within any efficiently recognizable set (such as IsBad_X) cannot be more than the probability any $.99d$-source does so (with an additional error $\varepsilon_{\mathsf{PEG}}$). In particular, this means that for the set of bad seeds, the fraction falling within S is:

$$\frac{D^{.98}}{D^{.99}} + \varepsilon_{\mathsf{PEG}} = \frac{1}{D^{.01}} + \varepsilon_{\mathsf{PEG}}$$

Once again, by taking $\varepsilon_{\mathsf{nm}}$ sufficiently small, which in turn makes D sufficiently large, this fraction is $\approx \varepsilon_{\mathsf{PEG}}$. See Definition 6.1 for the precise notion of pseudoentropy we consider.

The upshot here is that, just as that it is possible to construct condensers with seed length $1 \cdot \log 1/\varepsilon_{\mathsf{PEG}}$ (as [DPW14, BDT19] point out), it is plausible to construct pseudoentropy generators with the same dependence on $\varepsilon_{\mathsf{PEG}}$.[4]

A Table of Seeds, and the XOR Trick Unfortunately, since we are unaware of any suitable constructions of pseudoentropy generators with $d \approx 1 \cdot \log 1/\varepsilon_{\mathsf{PEG}}$, we must again follow the approach from [BDT19] to circumvent this barrier. Following their observation, we show that an even weaker object than either a pseudorandom generator or pseudoentropy generator suffices for our application. Instead, the object we construct, roughly described as a "somewhere" pseudoentropy generator, is a function that takes as input two "seeds" (rather than one as in a pseudoentropy generator) and outputs a seed to the non-malleable extractor: $\mathsf{SEPEG}\colon \{0,1\}^{r'} \times [A] \to \{0,1\}^d$.

The property of the somewhere pseudoentropy generator will be as follows (see Section 6 for a formal definition and details). We interpret the outputs of SEPEG as an $[R'] \times [A]$ table of seeds to nmExt (for constant A), with the property that all but an $\varepsilon_{\mathsf{PEG}}$ fraction of rows contains at least one good seed.

If nmExt is a $(t' = t \cdot A, k, \varepsilon_{\mathsf{nm}})$-non-malleable extractor, then for every such "good" row, the good seed y^* in the row has the property that $h_{y^*}(X)$ is a nearly uniform bit that is nearly independent of $h_y(X)$ for any other y in the row, and also any y in any other $t-1$ rows. Following a trick original from [Coh16a], if we take the XOR of $h_y(X)$ for all entries in each row, the result is a $(q = \varepsilon_{\mathsf{PEG}} \cdot R', t, \sqrt{\varepsilon_{\mathsf{nm}}})$ non-oblivious bit fixing source of length R'. As discussed before, this is exactly what we want, provided that $r' = 1.1 \cdot \log \frac{1}{\varepsilon_{\mathsf{PEG}}}$, or $R' = \left(\frac{1}{\varepsilon_{\mathsf{PEG}}}\right)^{1.1}$.

The Disperser Trick of [BDT19] for Error Reduction We finally discuss how to construct SEPEG and generate the table of seeds discussed above. The first ingredient is a PRG $G\colon \{0,1\}^r \to \{0,1\}^d$ against $\mathsf{IsBad}_X\colon \{0,1\}^d \to \{0,1\}$ with the wrong dependence on $\varepsilon_{\mathsf{PRG}}$. As discussed before, under the assumption that E is hard for large exponential time with exponential advice, it's possible to construct such PRGs with seed length $r = c_{\mathsf{PRG}}(\log n + \log 1/\varepsilon_{\mathsf{PRG}})$. In other words, for parameter $\varepsilon_{\mathsf{PEG}}$ if we insist on only paying, say, $r = .1 \cdot \log 1/\varepsilon_{\mathsf{PEG}}$ seed, we obtain a PRG with error $\varepsilon_{\mathsf{PRG}} = \varepsilon_{\mathsf{PEG}}^{1/(10 \cdot c_{\mathsf{PRG}})}$.

The second ingredient is a disperser with the right parameters. Ultimately, using the disperser of [Zuc07], with the same instantiation of parameters in [BDT19], one can construct a bipartite graph with $R' = \left(\frac{1}{\varepsilon_{\mathsf{PEG}}}\right)^{1.1}$ left nodes, left degree a constant A, and $R = \left(\frac{1}{\varepsilon_{\mathsf{PEG}}}\right)^{.1}$ right nodes. The property of this graph is that any subset of nodes on the left with fraction at least $\varepsilon_{\mathsf{PEG}}$ has a neighborhood on the right of fraction at least $\varepsilon_{\mathsf{PRG}} = \varepsilon_{\mathsf{PEG}}^{1/(10 \cdot c_{\mathsf{PRG}})}$.

Overall, the construction of SEPEG is then as follows, use r' bits of seed to choose a random left node y' of the disperser. Then, let the seeds to nmExt in the y'-th row be the evaluation of the PRG G on all neighbors of y' in the disperser. For correctness, suppose more than $\varepsilon_{\mathsf{PEG}}$ fraction of rows in this table have all entries as bad seeds. This implies the neighborhood of these rows is a $\varepsilon_{\mathsf{PRG}} = \varepsilon_{\mathsf{PEG}}^{1/(10 \cdot c_{\mathsf{PRG}})}$ fraction of seeds $y \in \{0,1\}^r$ to G such that $G(y)$ is a bad seed for nmExt. This is a contradiction, since, as discussed before, the fact that G is a $\varepsilon_{\mathsf{PRG}}$-PRG to IsBad_X implies that there cannot be more than $\sqrt{\varepsilon_{\mathsf{nm}}} + \varepsilon_{\mathsf{PRG}} \approx \varepsilon_{\mathsf{PRG}}$ fraction of bad seeds y.

[4]A probabilistic argument, similar to that in [DPW14] can show that pseudoentropy generators (for some appropriate notion) with seed length $\log \log |\mathcal{F}| + \log 1/\varepsilon_{\mathsf{PEG}}$ exists for any family of tests $\mathcal{F}$. Additionally, [AIKS16] construct the similar notion of a hitting set generator with seed length $\log 1/\varepsilon$.

The final construction and analysis is only slightly more involved, as we also require every entry of the table to be a *unique* seed to nmExt. See Theorem 6.3 for details.

2.4 The Final Piece: Sufficiently Good Explicit Non-Malleable Extractors

The discussion above is sufficient to obtain Theorem 1.2, and also Theorem 1.3 using currently known explicit constructions of non-malleable extractors with seed length $O(\log \frac{n}{\varepsilon_{\mathsf{nm}}})$. In order to improve the min-entropy parameter k, we must use newer constructions of non-malleable extractors, that do not quite have the right dependence on $\varepsilon_{\mathsf{nm}}$ in the seed length. The best known current constructions instead have seed length $O(\log n + \log \frac{1}{\varepsilon_{\mathsf{nm}}} \cdot \log\log \frac{1}{\varepsilon_{\mathsf{nm}}})$ [Li17]. For $\varepsilon_{\mathsf{nm}} = 1/\mathrm{poly}(n)$, this makes the runtime of IsBad_X superpolynomial. To fix this issue, we construct a new object, that we call a "seeded non-malleable extractor for samplable distributions," and use this in place of the non-malleable extractor. Such an object has the same properties as a non-malleable extractor, except that the properties are only true for sources X that are samplable by circuits of size n^c. In particular, the set of bad seeds to the non-malleable extractor (which depends on source X), is only guaranteed to be small when X is samplable. The point here, is that a non-malleable extractor for samplable distributions is both a weaker object than non-malleable extractors (it does not have to work for all k-sources), and extractors for samplable distributions (as it's allowed to use seed). Therefore, it should be simpler to construct such an object.

Our final observation is that the ideas to construct such an object have already been discussed above. We first take an explicit construction nmExt of a seeded non-malleable extractor, with wrong dependence on $\varepsilon_{\mathsf{nm}}$ in the seed length, and reduce the seed length by applying an appropriate PRG G. That is, the final construction will be $\mathsf{nmExt}(x, G(r))$. Since r can be $O(\log \frac{n}{\varepsilon_{\mathsf{nm}}})$, we are done. As long as G fools IsBad_X, then this construction has the property that the set of bad seeds $\in \{0,1\}^r$ is small whenever X is a samplable distribution.

We observe that even if the original number of seeds to nmExt is superpolynomial, which would lead to the procedure IsBad_X running in super-polynomial time, we can *speed up* IsBad_X using *nondeterminism* and show that it can be implemented by a polynomial sized *nondeterministic* circuit.

Indeed, on input a seed y, the circuit can use nondeterminism to guess the seeds $y_1, \ldots, y_t$ that violate Equation (1) and verify that this is true. Notice we are only aiming for the final construction to have $c_{\mathsf{nm}} \cdot \log \frac{n}{\varepsilon_{\mathsf{nm}}}$ seed length and run in polynomial time $n^{c_{\mathsf{nm}}}$, without any particular care for the dependence of the constant c_{nm} on the size of the circuits in question, n^c. Therefore, we can use results from [SU05] (stated formally in Theorem 3.9) to construct PRGs against nondeterministic circuits, under a hardness assumption against such circuits.

This argument proves Theorem 1.4. Note that the hardness assumption for nondeterministic circuits is used only to reduce the seed length of non-malleable extractors, and is unnecessary (as seen in Theorem 1.2) in case explicit constructions of non-malleable extractors with better dependence of the seed length on $\varepsilon_{\mathsf{nm}}$ can be achieved.

2.5 Perspective: A Computational Analog of Two Source Extractors

Thanks to Trevisan [Tre01], it is by now well known and widely celebrated wisdom in pseudorandomness that PRGs are a computational analogue to seeded extractors. We discuss here, that our work demonstrates a similar idea, used in the "other" direction. That is, we demonstrate that replacing an ingredient in the recipe for two-source extractors from [CZ16] with its computational analogue yields extractors for samplable distributions.

More specifically, at a high level, the celebrated work of [Tre01] observes that given a construction of a black-box PRG $G^f \colon \{0,1\}^d \to \{0,1\}^m$ that is given access to the truth table of a hard function f, replacing f with a sufficiently high entropy source X yields an extractor for that source. That is, the function $\mathsf{Ext}(x, y) = G^x(y)$ is an extractor.

Conversely, given a construction of a "reconstructive extractor" $\mathsf{Ext} \colon \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$, replacing the (true) randomness from the source X with the truth table of an appropriately hard function f yields a PRG $G^f(y) = \mathsf{Ext}(f, y)$.

Our work suggests a similar bridge between two sister lines of inquiry regarding deterministic extraction. On the "information theoretic" side, there is the question of extracting from two independent sources. On the "computational" side, there is the question of extracting from samplable distributions. While Trevisan and Vadhan [TV00] already observed that there is a similar analogy between the two objects, we suggest that the *specific recipe* used in recent two-source extractor constructions [CZ16, BDT19] can be followed to construct extractors for samplable distributions, if one can explicitly construct computational analogues of the information theoretic components used in these recipes.

More specifically, in the recipe of [CZ16] (see [Cha20] for a discussion) one applies a seeded non-malleable extractor to the first source X_1, and a seeded extractor on the second source X_2. We suggest that by replacing the second source X_2 with an appropriate hard function, we can obtain an extractor for samplable distributions, if we can replace the information theoretic seeded-extractor, with a suitable computational PRG.

Our current implementation of this analogy requires a shorter seed length from the non-malleable, compared to what is required in the information theoretic scenario of two-source extractors (see Remark 3.18 for a more technical discussion) and consequently, some current explicit constructions of non-malleable extractors are not "sufficiently nice" for our purposes. Nevertheless, as Theorem 1.2 demonstrates, a future improvement in the parameters of explicit constructions of non-malleable extractors, will immediately translate into improved extractors for samplable distributions.

The technical construction presented in this paper imitates a specific instantiation of the recipe for two-source extractors, given by Ben-Aroya, Doron and Ta-Shma [BDT19]. As we follow this specific instantiation, our extractors for samplable distributions inherit some limitations of this specific instantiation, and we only get constant error $\varepsilon > 0$, and only output a single bit.

However, it seems to us that the general approach could potentially imitate other instantiations of the recipe (say [CZ16] which do not suffer from these limitations).

More concretely, the reason that we currently output a single bit with constant error $\varepsilon > 0$ is that (at least the way we currently do it, and as explained in the overview above) we are pushed to use non-malleable extractors with constant t, which in turn pushes us to use the majority function as a bit-fixing source extractor. However, we would circumvent these limitations if we could find a way to use super-constant t (as is done in [CZ16]) or if we could find a way to replace the majority function with an extractor for non-oblivious bit-fixing sources with better error, and better output length.

3 Preliminaries

3.1 Random Variables

In general, lowercase variables (that represent integers) are base 2 logarithms of their upper case counterparts. For example, $D = 2^d$, and $R = 2^r$. The only explicit exception to this rule is T and t, where T usually denotes the running time of an algorithm, and t is the independence parameter of a non-malleable extractor.

We use U_n to denote the uniform distribution in n bits. We say that a random variable X has min-entropy k, or is a k-source, if $H_\infty(X) \stackrel{\text{def}}{=} -\log \max_x \Pr[X = x] \geq k$. We may also refer to such an X as a k-source.

For distributions A, B over universe Ω, we use $\Delta(A; B)$ to denote the statistical distance. That is, $\Delta(A; B) = \frac{1}{2}\sum_{x\in\Omega} |A(x) - B(x)| = \max_{S\subset\Omega} |A(S) - B(S)|$. For arbitrary random variables A, B, C, we use the notation $\Delta((A; B)|C)$ to denote the quantity $\Delta((A, C); (B, C))$. For a collection of random variables $A_1, \ldots, A_t$ we use $\{A_i\}_{i=1}^t$ to denote the joint distribution $(A_1, \ldots, A_t)$.

For subsets of sets, such as $A \subset U$ we will often abuse notation and conflate the set with an algorithm recognizing it. That is, we use the notation $x \in A$ and $A(x) = 1$ interchangeably. We also use $|A|$ and $|A^{-1}(1)|$ interchangeably.

We'll use the following lemma about boolean random variables:

Lemma 3.1 ([CZ16], Lemma 2.9)**.** *Let $X_1, \ldots, X_t, Y_1, \ldots, Y_k$ be boolean random variables. Further suppose that for any $i \in [t]$,*

$$\Delta((X_i; U_1)|\{X_j\}_{j\neq i}, Y_1, \ldots, Y_k) \leq \varepsilon.$$

Then

$$\Delta((X_1, \ldots, X_t; U_t)|Y_1, \ldots, Y_k) \leq 5t\varepsilon.$$

3.2 Definition of Circuits of Various Types

3.2.1 Nondeterministic Circuits And Σ_i-Circuits

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

Definition 3.2 (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 exist 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.[5]

[5]An alternative approach to define these circuit classes is using the Karp-Lipton notation for Turing machines with

3.2.2 Separating The Time And Advice Of A Nonuniform Procedure

We will use a hardness assumption against a "circuit model" in which we separate the running time and the amount of nonuniformity that a circuit uses. In order to comply with the standard notion of circuits (that work for a fixed input length n and have a fixed size) the definition below considers a circuit model of "nonuniform procedures with time t and a bits of advice" that is defined on a fixed input length n.

Definition 3.3 (Nonuniform procedures separating time and advice)**.** *Fix some encoding of Turing machines, and a universal Turing machine $\mathcal{U}$. We say that a function $f : \{0,1\}^n \to \{0,1\}$ is* computable by a nonuniform procedure with time t and a bits of advice*, if there exists a Turing machine M such that the length of an encoding $< M >$ of M is at most a, and on every input $x \in \{0,1\}^n$, the machine $\mathcal{U}$ simulates $M(x)$ in time at most t, and outputs $f(x)$.*

This notion obviously generalizes the standard notion of circuit a.k.a "machines with nonuniform advice" in the sense that for $a = t$ a nonuniform procedure with time t and a bits of advice is equivalent to a circuit of size t (when ignoring the precise circuit model which only matters up to a fixed polynomial).

This definition allows such procedures to run in time $t > 2^n$ and use $a < 2^n$ bits of advice.

3.3 Hardness Assumptions

3.3.1 The Old Assumptions: E is Hard for Exponential Size Circuits of Type X

The following hardness assumption was popularized by the celebrated work of Impagliazzo and Wigderson [IW97].

Definition 3.4 (E is hard for exponential size circuits)**.** *We say that "E is hard for exponential size circuits of type X" if there exist constants $0 < \alpha < C_{\mathsf{easy}}$, and a language L in $\mathsf{DTIME}(2^{C_{\mathsf{easy}} \cdot n})$, such that for every sufficiently large n, the characteristic function of L on inputs of length n cannot be computed by circuits of size $2^{\alpha n}$ of type X.*

Impagliazzo and Wigderson [IW97] (see also [STV01, SU05, Uma03]) showed that $\mathsf{BPP} = \mathsf{P}$ assuming that E is hard for exponential size (deterministic) circuits. Later work [KvM02, MV05, SU05, SU06] considered the stronger assumption that E is hard for nondeterministic circuits, and used such assumptions to show that $\mathsf{AM} = \mathsf{NP}$.

In their seminal paper on extractors for samplable distributions, Trevisan and Vadhan introduced a version of the assumption for Σ_i-circuits, and their extractor is constructed under the assumption that E is hard for exponential size Σ_6^{P}-circuits.

3.3.2 The New Assumption: E is Hard for Large Exponential Time with Exponential Advice

In this paper we will use a version of the Impagliazzo and Wigderson assumption against nonuniform procedures with time $t = 2^{O(n)}$ (that can be larger than 2^n) that use $a < 2^n$ bits of advice. To the best of our knowledge hardness assumptions of this form were first introduced by Chen and Tell [CT21] in the context of fast derandomization. We will use the following assumption.

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 $\mathsf{DTIME}^{\Sigma_i^{\mathsf{P}}}(s^{\Theta(1)})/s^{\Theta(1)}$.

Definition 3.5 (E is hard for large exponential time with exponential advice)**.** *We say that "E is hard for large exponential time with exponential advice" if there exists a constant $\alpha > 0$ such that for every constant $C_{\mathsf{hard}} > 1$, there exists a constant $C_{\mathsf{easy}} > C_{\mathsf{hard}}$, and a language L in $\mathsf{DTIME}(2^{C_{\mathsf{easy}} \cdot n})$, such that for every sufficiently large n, the characteristic function of L on inputs of length n cannot be computed by nonuniform procedures that run in time $2^{C_{\mathsf{hard}} \cdot n}$ and use $2^{\alpha \cdot n}$ bits of advice.*

Note that the assumption above is similar to the assumption in Definition 3.4 except that we now allow the circuit/procedure to run in time $2^{C_{\mathsf{hard}} \cdot \ell}$ for an arbitrary $C_{\mathsf{hard}} \geq 1$, and we require that for every constant $b \geq 1$ there is a constant $C_{\mathsf{easy}} > C_{\mathsf{hard}}$ such that the problem L can be computed in time $2^{C_{\mathsf{easy}} \cdot n}$.

Consequently, it is obvious the assumption that E is hard for large exponential time with exponential advice implies the assumption that E is hard for exponential size circuits.

A discussion that compares this assumption to assumptions like E is hard for exponential deterministic/nondeterministic circuits, appears in Section 9.1.

3.4 Pseudorandom Generators

We need the following standard definition of pseudorandom distributions and generators.

Definition 3.6 (Pseudorandom generators)**.** *A distribution X on n bits is ε-*pseudorandom *for a class $\mathcal{C}$ of functions, if for every $C : \{0,1\}^n \to \{0,1\} \in \mathcal{C}$,*

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

*A function $G : \{0,1\}^d \to \{0,1\}^n$ is an ε-*PRG *for $\mathcal{C}$ if $G(U_d)$ is ε-pseudorandom for $\mathcal{C}$. G is* seed-extending *if the function $G'(x) = x \circ G(x)$ is an ε-PRG for $\mathcal{C}$.*

3.4.1 PRGs for Deterministic Circuits

The classical result of Impagliazzo and Wigderson [IW97] (see also [STV01, SU05, Uma03]) gives a PRG for poly-size circuits, under the assumption that E is hard for exponential size circuits.

Theorem 3.7 (PRGs from hardness assumptions [IW97])**.** *If E is hard for exponential size circuits then there exists a constant $a > 1$ such that for every sufficiently large s, there is a*

$$G : \{0,1\}^{a \cdot \log s} \to \{0,1\}^s$$

that is a seed-extending $\frac{1}{s}$-PRG for circuits of size s. Furthermore, G is computable in time $\mathrm{poly}(s)$. (Here, the polynomial depends on the constants $\alpha, C_{\mathsf{easy}}$ in the hardness assumption).

An immediate corollary of Theorem 3.7 is the standard application of PRGs to approximate the number of accepting inputs of a given circuit.

Theorem 3.8. *If E is hard for exponential size circuit then for every sufficently large n, $s \geq n$ and every $\varepsilon > 0$, there is an algorithm $A'_{s,\varepsilon}$ that given a circuit A of size s, outputs a number $\widetilde{p}$ such that*

$$|\Pr[A(U_n) = 1] - \widetilde{p}| \leq \frac{1}{\varepsilon}.$$

Furthermore, A' runs in time $\mathrm{poly}(s, \frac{1}{\varepsilon})$ where the polynomial depends on the constants in the hardness assumption.

3.4.2 PRGs for Nondeterministic Circuits

Subsequent work by [KvM02, MV05, SU05, SU06] extended Theorem 3.7 replacing "(deterministic) circuits" by "nondeterministic circuits". The theorem below is identical to Theorem 3.7 except that the hardness assumption is against nondeterministic circuits, and the PRG fools nondeterministic circuits.

Theorem 3.9 (PRGs for nondeterministic circuits from hardness assumptions against nondeterministic circuits [SU05])**.** *If* E *is hard for exponential size nondeterministic circuits then there exists a constant* $a > 1$ *such that for every sufficiently large* s*, there is a*

$$G : \{0,1\}^{a \cdot \log s} \to \{0,1\}^s$$

that is a seed-extending $\frac{1}{s}$*-PRG for nondeterministic circuits of size* s*. Furthermore,* G *is computable in time* $\mathrm{poly}(s)$*. (Here, the polynomial depends on the constants* $\alpha, C_{\mathsf{easy}}$ *in the hardness assumption).*

3.5 Dispersers

We'll use dispersers as an ingredient in one of our constructions. We use a less standard parametrization, stated below.

Definition 3.10. *A function* $\Gamma\colon [N] \times [D] \to [M]$ *is a* (K, K')*-disperser if for every* $S \subset [N]$ *with* $|S| \geq K$*,* $\left|\bigcup_{i \in [D]} \Gamma(S, i)\right| \geq K'$*.*

The more standard definition sets $K' = (1 - \varepsilon) \cdot M$, where ε is the parameter. However, for the setting that we are interested in the "error parameter" ε is close to one, rather than close to zero, and the parametrization in Definition 3.10 is more natural. This setting (where ε approaches one) was considered by Zuckerman [Zuc07]. The following theorem is a restatement of Zuckerman's theorem [Zuc07] using Definition 3.10.

Theorem 3.11 ([Zuc07], Theorem 1.9)**.** *For every constant* $\delta > 0$ *there exists a constant* c_δ *such that for every sufficiently large* n*, and every* $0 < s < 1$*, for* $N = 2^n$*,* $K = N^\delta$*,* $M = N^{\delta/2} = K^{1/2}$ *and* $K' = s \cdot M$*, there is a* (K, K')*-disperser* $\Gamma : [N] \times [D] \to [M]$*, with* $D = c_\delta \cdot \frac{n}{\log \frac{1}{s}}$*. Furthermore,* Γ *is computable in time* $\mathrm{poly}(n)$*.*

We remark that Zuckerman's theorem is more general than the one we state here, and allows M to be closer to K. We do not need this property in this paper.

A key property of Theorem 3.11 is that setting $s = M^{-\gamma}$ for a constant $0 < \gamma < 1$, one obtains that D is a constant that depends on δ, γ.

3.6 Majority as an Extractor for Non-oblivious Bit Fixing Sources

Following [BDT19], we use the function majority as an extractor for non-oblivious bit-fixing source.

Definition 3.12. *A distribution* X *is called* (t, γ)*-wise independent if the restriction of* X *to every* t*-coordinates is* γ*-close to* U_t*.*

Definition 3.13. *A source* X *over* $\{0,1\}^n$ *is a* (q, t, γ) *non-oblivious bit-fixing source if there is a subset* $Q \subset [n]$ *of size at most* q *such that the joint distribution of the bits in* $[n] \setminus Q$ *is* (t, γ)*-wise independent. The bits in* Q *are allowed to arbitrarily depend on the bits in* $[n] \setminus Q$*.*

Viola [Vio14] showed that the majority function is an extractor for non-oblivious bit-fixing sources. This extractor was used by Ben-Aroya, Doron and Ta-Shma [BDT19].

Theorem 3.14 ([Vio14], See [BDT19], Lemma 2.17.)**.** *For* $\mathrm{Maj}\colon \{0,1\}^R \to \{0,1\}$*, there exists a constant* c_{Maj} *such that for every* $\alpha > 0$ *and a* $\left(q = R^{\frac{1}{2}-\alpha}, t, \gamma\right)$ *non-oblivious bit-fixing source* X *on* R *bits,*

$$\left|\Pr[\mathrm{Maj}(X_1,\ldots,X_R)=1]-\frac{1}{2}\right| \le c_{\mathrm{Maj}} \cdot \left(\frac{\log t}{t} + R^{-\alpha} + \gamma R^t\right).$$

For a fixed constant error ε of the final extractor, we will instantiate t, and α, and choose R (and γ) appropriately, so that the RHS is at most ε.

Corollary 3.15. *For any* $\varepsilon > 0$*, let* $t = \frac{9c_{\mathrm{Maj}}^2}{\varepsilon^2}$*, and let* $\alpha = 1/8$*. Suppose further that* $R \ge \frac{10^4 \cdot c_{\mathrm{Maj}}^8}{\varepsilon^8}$*, and* $\gamma \le \frac{\varepsilon}{3\cdot c_{\mathrm{Maj}} \cdot R^t}$*. Then for a* $\left(q = R^{\frac{1}{4}}, t, \gamma\right)$ *non-oblivious bit-fixing source* X *on* R *bits,*

$$\left|\Pr[\mathrm{Maj}(X_1,\ldots,X_R)=1]-\frac{1}{2}\right| \le \varepsilon.$$

3.7 Non-Malleable Extractors

3.7.1 Definition of Non-malleable extractors

We will use the following standard definition of non-malleable extractors (which is defined using the notation explained in the beginning of Section 3).

Definition 3.16. *A function* $\mathsf{nmExt}\colon \{0,1\}^n \times \{0,1\}^d \to \{0,1\}$ *is a* $(t, k, \varepsilon_{\mathsf{nm}})$*-non-malleable extractor if the following holds: for any* (n,k)*-source* X*, and* t*-tuple of functions* $(f_1,\ldots,f_t)$*, where each* $f_i\colon \{0,1\}^d \to \{0,1\}^d$ *has no fixed points, we have:*

$$\Delta((\mathsf{nmExt}(X,U_d);U_1)|\{\mathsf{nmExt}(X,f_i(U_d))\}_{i=1}^t) \le \varepsilon_{\mathsf{nm}}.$$

3.7.2 Nice Constructions of Non-Malleable Extractors

We plan to use non-malleable extractors in order to construct extractors for samplable distributions. When shooting for small constant error in the extractor for samplable distributions, we will require non-malleable extractors with constant t, and $\varepsilon_{\mathsf{nm}} = n^{-c_{\mathsf{error}}}$ for a sufficiently large constant c_{error} that will be chosen in the proof.

In our reduction, the running time of the final extractor will be $\mathrm{poly}(n, 2^d, 2^{2^t}, \mathsf{time}(\mathsf{nmExt}))$. Therefore, in order to obtain extractors for samplable distributions that run in time polynomial in n, we will require constructions of non-malleable extractors in which this is $\mathrm{poly}(n)$. We refer to such constructions of non-malleable extractors as "nice", and the precise formulation is stated below.

Definition 3.17 (Nice construction of non-malleable extractors)**.** *Let* $k(t,n,\varepsilon_{\mathsf{nm}})$ *be some integer function. We say that "there is a nice construction of non-malleable extractors for min-entropy* k*" if for every constant* t*, there exists a constant* c_{nm} *such that for every constant* $c_{\mathsf{error}} \ge 1$*, and for every sufficiently large* n*, setting* $\varepsilon_{\mathsf{nm}}(n) = n^{-c_{\mathsf{error}}}$*, there is a function* $d(n) \le c_{\mathsf{nm}} \cdot \log \frac{n}{\varepsilon_{\mathsf{nm}}(n)}$ *and a* $(t, k(t,n,\varepsilon_{\mathsf{nm}}), \varepsilon_{\mathsf{nm}}(n))$*-non-malleable extractor* $\mathsf{nmExt} : \{0,1\}^n \times \{0,1\}^{d(n)} \to \{0,1\}$*, such that* nmExt *can be computed in time* $(\frac{n}{\varepsilon_{\mathsf{nm}}(n)})^{c_{\mathsf{nm}}}$.

Remark 3.18 (The role of nice constructions of non-malleable extractors in two-source extractors)**.** *Loosely speaking, a nice construction of non-malleable extractors (as defined in Definition 3.17) should satisfy two properties:*

- *The running should be* $\mathrm{poly}(n)$*, even if* $\varepsilon_{\mathsf{nm}}(n) = n^{-c_{\mathsf{error}}}$ *for a large constant* c_{error}*.*
- *The seed length* d *should be* $O(\log \frac{n}{\varepsilon_{\mathsf{nm}}(n)})$ *for* $\varepsilon_{\mathsf{nm}}(n) = n^{-c_{\mathsf{error}}}$*.*

We remark that while the previous reductions that reduce two-source extractors to non-malleable extractors [CZ16, BDT19] do require *that the non-malleable extractor is nice in the first sense (and some of them require this for* $\varepsilon_{\mathsf{nm}}(n) = n^{-\omega(1)}$*), they* do not require *that the non-malleable extractor is nice in the second sense.*

This means that some previously known constructions of non-malleable extractors that are suitable for the application of constructing two-source extractors, will not be suitable for our application of constructing extractors for samplable distributions.

3.7.3 Some Known Constructions of Non-Malleable Extractors.

Some of the known constructions on non-malleable extractors in the literature are nice. One such example is by Cohen, Raz and Segev [CRS14] that gave a nice construction of non-malleable extractors for k that is slightly larger than $n/2$.

Theorem 3.19 (See [CRS14], Theorem 1.7)**.** *For every constant* $\delta > 0$*, there is a nice construction of non-malleable extractors for min-entropy* $k(t, n, \varepsilon_{\mathsf{nm}}) = \left(\frac{1}{2} + \delta\right) n$*.*

For smaller values of min-entropy k, the best known construction is by Li [Li17]. This construction is not quite "nice" according to our definition. This is because the dependence of the seed length d on $n, \varepsilon_{\mathsf{nm}}$ is not $O(\log \frac{n}{\varepsilon_{\mathsf{nm}}(n)})$, but rather $d = O(\log n + \log \frac{1}{\varepsilon_{\mathsf{nm}}} \cdot \log\log \frac{1}{\varepsilon_{\mathsf{nm}}})$, which is slightly worse than what we need.

Theorem 3.20 ([Li17])**.** *For any constant* t *the following holds, there exists a constant* $c_{\mathsf{Li}} = c_{\mathsf{Li}}(t)$ *such that the following holds. For every sufficiently large* n *and for any* $\varepsilon_{\mathsf{nm}} > 0$*, there exists a* $(t, k = d, \varepsilon_{\mathsf{nm}})$*-non-malleable extractor*

$$\mathsf{nmExt}\colon \{0,1\}^n \times \{0,1\}^{d \le c_{\mathsf{Li}}\left(\log n + \log \frac{1}{\varepsilon_{\mathsf{nm}}} \cdot \log\log \frac{1}{\varepsilon_{\mathsf{nm}}}\right)} \to \{0,1\}$$

computable in time $\left(\frac{n}{\varepsilon_{\mathsf{nm}}}\right)^{c_{\mathsf{Li}}}$*.*

3.7.4 A Useful Property of Non-Malleable Extractors

Following Chattopadhyay and Zuckerman [CZ16] we will make use of the following property of non-malleable extractors (that also plays an important role in constructions of two-source extractors).

Loosely speaking, Theorem 3.21 below says that there is a small set of "bad seeds", such that for every seed $y \in \{0,1\}^d$ that is not bad, and every t additional distinct seeds, the output of the extractor using y is uniform, "even when conditioned" on the t outputs obtained using the additional seeds. Furthermore, the joint distribution of the output of any t good seeds is close to uniform.

Theorem 3.21 (See [CZ16], Lemma 3.4). *Let* $\mathsf{nmExt}\colon \{0,1\}^n \times \{0,1\}^d \to \{0,1\}$ *be a* $(t,k,\varepsilon_{\mathsf{nm}})$*-non-malleable extractor. For each* $y \in \{0,1\}^d$*, define the function* $h_y\colon \{0,1\}^n \to \{0,1\}$ *as* $h_y(x) = \mathsf{nmExt}(x,y)$*. For any* (n,k)*-source* X*, and any* $\varepsilon'_{\mathsf{nm}} \geq \varepsilon_{\mathsf{nm}}$*, define:*

$$\mathsf{BAD}_{X,\varepsilon'_{\mathsf{nm}}} = \{y \in \{0,1\}^d \mid \exists \text{ distinct } y_1,\ldots,y_t \in \{0,1\}^d \backslash \{y\}, \Delta\left((h_y(X);U_1)|\{h_{y_i}(X)\}_{i=1}^t\right) > \sqrt{\varepsilon'_{\mathsf{nm}}}\}.$$

Then $|\mathsf{BAD}_{X,\varepsilon'_{\mathsf{nm}}}| \leq \sqrt{\varepsilon'_{\mathsf{nm}}}D$*. Moreover, for any* $y_1,\ldots,y_t \notin \mathsf{BAD}_{X,\varepsilon'_{\mathsf{nm}}}$*:*

$$\Delta(\{h_{y_i}\}_{i=1}^t; U_t) \leq 5t\sqrt{\varepsilon'_{\mathsf{nm}}}.$$

We note that Theorem 3.21 is only stated in [CZ16] for $\varepsilon'_{\mathsf{nm}} = \varepsilon_{\mathsf{nm}}$, however, the statement above is also true as any $(t,k,\varepsilon_{\mathsf{nm}})$-non-malleable extractor is also a $(t,k,\varepsilon'_{\mathsf{nm}})$-non-malleable extractor.

4 PRGs For Procedures With Large Time and Small Advice

We will require a PRG against adversaries that are nonuniform procedures which use s bits of advice, and run in time s^c for a potentially large constant c . It will be crucial for our application that the seed length of the PRG *does not* depend on c. Fortunately, the same argument used by [IW97] in the proof of Theorem 3.7, yields such PRGs, if we replace the hardness assumption in Theorem 3.7 with the new hardness assumption that E is hard for large exponential time with exponential advice (introduced in Definition 3.5).

Theorem 4.1 (stated below) is very similar to the classical Theorem 3.7 with the key difference being that we consider nonuniform procedures that use s bits of advice, but are allowed to run in time s^c, for a large constant c. The key feature of the theorem is that the constant a that governs the seed length *does not* depend on the constant c that governs the running time. Indeed, the theorem allows one to choose c to be arbitrarily large as a function of a.

Theorem 4.1 (PRGs for large time and small advice). *If* E *is hard for large exponential time with exponential advice, then there exists a constant* $a > 1$ *such that for every constant* $c > 1$ *and every sufficiently large* s*, there is a*

$$G : \{0,1\}^{a\cdot\log s} \to \{0,1\}^s$$

that is a seed-extending $\frac{1}{s}$*-PRG for nonuniform procedures that run in time* s^c *and use* s *bits of advice. Furthermore,* G *is computable in time* $\mathrm{poly}(s)$*. (Here, the polynomial depends on the constant* α *in the hardness assumption, the constant* c*, and is also affected by the dependence between* C_{hard} *and* C_{easy} *in the hardness assumption).*

We also remark that alternative analyses and modifications of the hardness vs randomness framework have been carried out before [KvM02, CT21, LP23] in order to accommodate different hardness assumptions and carefully account for the usage of various resources (time, advice, etc.). However, to the best of our knowledge the theorem above has not been stated in its exact form before. Theorem 4.1 follows from the argument of [IW97] in exactly the same way, with minor modifications. This argument is explained in the proof below.

Proof: We review the classical proof of [IW97] for Theorem 3.7. At a high level, the proof shows that for every constant $\alpha > 0$, there are constants $0 < \delta < \frac{\alpha}{2}$, $d > 1$, and a construction that transforms a function $f : \{0,1\}^\ell \to \{0,1\}$ into a function $G_f : \{0,1\}^{d\cdot\ell} \to \{0,1\}^{2^{\delta\cdot\ell}}$. It is not necessary to go into details of this transformation (which involves "hardness amplification from worst case to average case" and using the "Nisan-Wigderson generator"). Instead we note that this transformation has the following properties:

- For every function $f : \{0,1\}^{\ell} \to \{0,1\}$, the function G_f can be computed in time $2^{O(\ell)}$ with oracle access to f.
- For any functions $f : \{0,1\}^{\ell} \to \{0,1\}$, and $D : \{0,1\}^{2^{\delta \cdot \ell}} \to \{0,1\}$, if G_f is not a $\frac{1}{2^{\delta \cdot \ell}}$-PRG for D, then there exists an oracle circuit $C^{(\cdot)}$ of size $2^{\frac{\alpha}{2} \cdot \ell}$ such that C^D computes f.

The standard case. In the regular case of Theorem 3.7 the proof works as follows: under the assumption that E is hard for exponential size circuits, given the parameter s, the proof chooses $\ell = \frac{1}{\delta} \cdot \log s$ so that $2^{\delta \ell} = s$. It takes f to be the characteristic function of the hard problem L on inputs of length ℓ. It obtains a function G_f from $d\ell = \frac{d}{\delta} \log s$ to s bits. This means that the seed length of G_f is $a \log s$ for $a = \frac{d}{\delta}$ which is a constant that depends on α.

By the first item, G_f can be computed in time $2^{O(\ell)} \cdot 2^{C_{\mathsf{easy}} \cdot \ell} = \mathrm{poly}(s)$ (where the polynomial depends on $\alpha, C_{\mathsf{easy}}$). If G_f is not a $\frac{1}{s}$-PRG for some circuit $D : \{0,1\}^s \to \{0,1\}$ of size $s = 2^{\delta \ell}$, then by the second item, f can be computed by the circuit C^D which can be simulated by a circuit of size $2^{\frac{\alpha}{2} \cdot \ell} \cdot \mathsf{size}(D) \leq 2^{\frac{\alpha}{2} \cdot \ell} \cdot 2^{\frac{\alpha}{2} \cdot \ell} = 2^{\alpha \ell}$ which is a contradiction.

The case of separating time and advice. In order to prove Theorem 4.1 we extend the standard argument as follows: under the assumption that E is hard for large exponential time with exponential advice, given the parameters s and c, the proof chooses $\ell = \frac{1}{\delta} \cdot \log s$ so that $2^{\delta \ell} = s$ (just like before). The key difference is that in Theorem 4.1 we are allowing c to depend on α, and so $s^c = 2^{c \cdot \delta \cdot \ell}$ might be larger than 2^{ℓ}. We choose $C_{\mathsf{easy}} = 2 \cdot c \cdot \delta \cdot \ell$, and the hardness assumption provides a constant $C_{\mathsf{easy}} > C_{\mathsf{hard}}$ and a hard problem L. We take f to be the characteristic function of the hard problem on inputs of length ℓ. we obtain a function G_f from $d\ell = \frac{d}{\delta} \log s$ to s bits. This means that the seed length of G_f is $a \log s$ for $a = \frac{d}{\delta}$ which is a constant that depends on α (but not on c).

By the first item, G_f can be computed in time $2^{O(\ell)} \cdot 2^{C_{\mathsf{easy}} \cdot \ell} = \mathrm{poly}(s)$ (where the polynomial depends on $\alpha, C_{\mathsf{easy}}$ and recall that C_{easy} depends on b that depends on c). If G_f is not a $\frac{1}{s}$-PRG for some nonuniform procedure $D : \{0,1\}^s \to \{0,1\}$ with time $s^c = 2^{c \cdot \delta \cdot \ell}$ and $s = 2^{\delta \ell}$ bits of advice, then by the second item, f can be computed by the circuit C^D which can be simulated by a nonuniform procedure with time $2^{\frac{\alpha}{2} \cdot \ell} \cdot \mathsf{time}(D) \leq 2^{\frac{\alpha}{2} \cdot \ell} \cdot 2^{c \cdot \delta \ell} \leq 2^{2 \cdot c \cdot \delta \ell} = 2^{C_{\mathsf{easy}} \cdot \ell}$ and number of advice bits that is used by C^D is $2^{\frac{\alpha}{2} \cdot \ell} + \mathsf{advice}(D) \leq 2^{\frac{\alpha}{2} \cdot \ell} + 2^{\frac{\alpha}{2} \cdot \ell} \leq 2^{\alpha \ell}$. Once again we obtain a contradiction.

■

Remark 4.2 (Necessity of the assumption in Theorem 4.1)**.** *It is easy to see that the assumption that* E *is hard for large exponential time with exponential advice is* necessary *to construct the PRG guaranteed in Theorem 4.1. More specifically, it is easy to see (and was first pointed out in [ISW99]) that given a* 0.49*-PRG* $G : \{0,1\}^d \to \{0,1\}^n$ *against a class* $\mathcal{C}$*, if* $n \geq d+1$*, then the function* $f : \{0,1\}^{d+1} \to \{0,1\}$*, that given* $x \in \{0,1\}^{d+1}$ *checks whether there exists* $y \in \{0,1\}^d$ *such that* x *is a prefix of* $G(y)$*, cannot be computed by* $\mathcal{C}$*. By definition,* f *can be computed in time* $\mathrm{poly}(2^d, n)$*, and therefore, the conclusion of Theorem 4.1 implies that* E *is hard for large exponential time with exponential advice.*

In the remainder of the paper, it will be convenient to use the following immediate corollary of Theorem 4.1.

Theorem 4.3. *If* E *is hard for large exponential time with exponential advice, then for every constant* $c > 1$, *there exists a constant* c_{PRG}, *such that for every constant* $c_{\mathsf{runtime}} > 1$, *every function* $\varepsilon_{\mathsf{PRG}} = \varepsilon_{\mathsf{PRG}}(n)$, *and every sufficiently large* n, *there is a*

$$G\colon \{0,1\}^{c_{\mathsf{PRG}}\cdot\log(n/\varepsilon_{\mathsf{PRG}})} \to \{0,1\}^{n^c}$$

that is a seed-extending $\varepsilon_{\mathsf{PRG}}$*-PRG for nonuniform procedures that run in time* $n^{c_{\mathsf{runtime}}}$ *and use* n^c *bits of advice. Furthermore, G is computable in time* $\mathrm{poly}(n^c, n^{c_{\mathsf{runtime}}}, \frac{1}{\varepsilon_{\mathsf{PRG}}})$. *(Here, the polynomial depends on the constant* α *in the hardness assumption, and is also affected by the dependence between* C_{hard} *and* C_{easy} *in the hardness assumption).*

A key feature of this theorem is that c_{PRG} does not depend on c_{runtime}.

Proof: Theorem 4.3 follows directly from Theorem 4.1 by taking $s = \max(n^c, \frac{1}{\varepsilon_{\mathsf{PRG}}(n)})$. ■

5 An Algorithm For Recognizing Bad Seeds

A key component in recent constructions of two-source extractors [CZ16, BDT19] is applying a $(t, k, \varepsilon_{\mathsf{nm}})$-non-malleable extractor $\mathsf{nmExt} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}$ to transform a distribution X over $\{0,1\}^n$ with $H_\infty(X) \geq k$, into $D = 2^d$ random variables $(\mathsf{nmExt}(X,y))_{y\in\{0,1\}^d}$. As stated formally in Theorem 3.21, [CZ16] showed that there is a small set $\mathsf{BAD}_{X,\varepsilon_{\mathsf{nm}}} \subseteq \{0,1\}^d$ of "bad seeds", such that the distribution $Z = (\mathsf{nmExt}(X,y))_{y\in\{0,1\}^d\setminus\mathsf{BAD}_{X,\varepsilon_{\mathsf{nm}}}}$ has the property that every subset of t bits from Z is (close to) uniform.[6] In this section we show that if:

- X is samplable by size n^c circuits,
- E is hard for large exponential time with exponential advice, and
- The non-malleable extractor nmExt comes from a nice construction (See Definition 3.17),

then for every constant $c_{\mathsf{error}} \geq 1$, setting $\varepsilon_{\mathsf{nm}} = n^{-c_{\mathsf{error}}}$, deciding whether a seed y is bad can be done by a nonuniform procedure with running time $\mathrm{poly}(n, \frac{1}{\varepsilon_{\mathsf{nm}}})$ and advice n^c.

Jumping ahead, we mention that we will later use the theorem choosing a constant c_{error} that is much larger than c, so that this procedure will run in time $n^{c_{\mathsf{runtime}}}$ where c_{runtime} is a constant that is much larger than c. Recall that the PRG of Theorem 4.3 is set up for this scenario, and has seed length that depends on c but not on c_{runtime}.

Theorem 5.1. *Assume that:*

- E *is hard for exponential size circuits.*
- *There is a nice construction* nmExt *of non-malleable extractors for min-entropy* $k = k(t, n, \varepsilon_{\mathsf{nm}})$.

[6]The property mentioned above in the text is oversimplified, and following [BDT19] we need the stronger property (stated formally in Theorem 3.21) which loosely states that every good bit Z_y is (close to uniform) not just conditioned on t good bits, but also when some of the t bits are bad.

Figure 1: A procedure that recognizes bad seeds

```
Procedure $\mathsf{IsBad}_{X,n,t,k,\varepsilon_{\mathsf{nm}}}$:

Ingredients: The procedure utilizes as a subroutine a $(t, k, \varepsilon_{\mathsf{nm}}/16)$-non-malleable extractor
    $\mathsf{nmExt}\colon \{0,1\}^n \times \{0,1\}^d \to \{0,1\}$ (that comes from a nice construction). Define $h_y(x) =$
    $\mathsf{nmExt}(x, y)$.

Input: $y \in \{0,1\}^d$, a seed to nmExt.

Advice: A description of a circuit $X$ of size $n^c$.

Output: A bit $b \in \{0,1\}$.

Operation:

  1. For all $t$-tuples of distinct $y_1, \dots, y_t \in \{0,1\}^d$, $y_i \neq y$:
     (a) For every subset $S \subset \{0,1\}^t$:
          i. Let $\mathbf{1}_S\colon \{0,1\}^t \to \{0,1\}$ denote the algorithm determining membership in $S$.
         ii. Define $A_1\colon \{0,1\}^{n^c} \to \{0,1\}$ as:
                 $A_1(z) = \mathbf{1}_S(h_y(X(z)), h_{y_1}(X(z)), \dots, h_{y_t}(X(z)))$.
             Also define $A_2\colon \{0,1\}^{n^c} \times \{0,1\} \to \{0,1\}$ as:
                 $A_2(z, a) = \mathbf{1}_S(a, h_{y_1}(X(z)), \dots, h_{y_t}(X(z)))$.
        iii. Note that $A_1, A_2$ are circuits of size $s = t \cdot n^c \cdot (\frac{n}{\varepsilon_{\mathsf{nm}}})^{c_{\mathsf{nm}}} = \mathrm{poly}(n^c, n^{c_{\mathsf{error}}})$, for a polyno-
             mial that depends on $t, c_{\mathsf{nm}}$. We will use the algorithm from Theorem 3.8, compute $\widetilde{p}_1$
             and $\widetilde{p}_2$, that are $\sqrt{\varepsilon_{\mathsf{nm}}}/8$ approximations to $\Pr[A_1(U_{n^c}) = 1]$ and $\Pr[A_2(U_{n^c+1}) = 1]$
             respectively. By Theorem 3.8 this can be done in time $\mathrm{poly}(s, \frac{1}{\varepsilon_{\mathsf{nm}}}) = \mathrm{poly}(n^c, n^{c_{\mathsf{error}}})$
             for a polynomial that depends on $t$, $c_{\mathsf{nm}}$, and the constants in the hardness assump-
             tion.
         iv. If $|\widetilde{p}_1 - \widetilde{p}_2| > \sqrt{\varepsilon_{\mathsf{nm}}}/2$, then break and output $b = 1$.
  2. Output $b = 0$.
```

Let $t, c, c_{\mathsf{error}} \geq 1$ be constants. Let n be sufficiently large, and let X be a distribution over $\{0,1\}^n$ that is samplable by a size n^c circuit, and has $H_\infty(X) \geq k(n)$. Set $\mathsf{nmExt} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}$ *to be a $(k, t, \varepsilon_{\mathsf{nm}}/16)$-non-malleable extractor, where $\varepsilon_{\mathsf{nm}} = n^{-c_{\mathsf{error}}}$, and note that $D = 2^d \leq n^{c_{\mathsf{nm}} \cdot c_{\mathsf{error}}}$, let* $\mathsf{IsBad}_{X,n,t,k,\varepsilon_{\mathsf{nm}}}(y)$ *be the procedure in Figure 1 and let* $\mathsf{BAD}'_X = \{y \mid \mathsf{IsBad}_{X,n,t,k,\varepsilon_{\mathsf{nm}}}(y) = 1\}$.

1. $\mathsf{BAD}_{X,\varepsilon_{\mathsf{nm}}} \subset \mathsf{BAD}'_X$ *and in particular, for any $y \notin \mathsf{BAD}'_X$, and any distinct $y_1, \dots, y_t$:*

$$\Delta\left((h_y(X); U_1) | \{h_{y_i}(X)\}_{i=1}^t\right) \leq \sqrt{\varepsilon_{\mathsf{nm}}}.$$

Furthermore, for any $y_1, \ldots, y_t \notin \mathsf{BAD}'_X$:

$$\Delta(\{h_{y_i}\}_{i=1}^t; U_t) \le 5t\sqrt{\varepsilon_{\mathsf{nm}}}.$$

2. *$\mathsf{BAD}'_X \subset \mathsf{BAD}_{X,\varepsilon_{\mathsf{nm}}/16}$ and in particular, $|\mathsf{BAD}'_X| \le \sqrt{\varepsilon_{\mathsf{nm}}}D$.*

3. *$\mathsf{IsBad}_{X,n,t,k,\varepsilon_{\mathsf{nm}}}$ runs in time $\mathrm{poly}(n^c, n^{c_{\mathsf{error}}} = \frac{1}{\varepsilon_{\mathsf{nm}}})$ (where the polynomial depends on t, c_{nm} and the constants in the hardness assumption) and uses n^c bits of nonuniformity.*

Proof:

1. We first prove $\{0,1\}^d \setminus \mathsf{BAD}'_X \subset \{0,1\}^d \setminus \mathsf{BAD}_{X,\varepsilon_{\mathsf{nm}}}$. If $\mathsf{IsBad}_{X,n,t,k,\varepsilon_{\mathsf{nm}}}(y) = 0$ then for every distinct $y_1, \ldots, y_t$:

$$\Delta\left((h_y(X); U_1)|\{h_{y_i}(X)\}_{i=1}^t\right) \le \frac{\sqrt{\varepsilon_{\mathsf{nm}}}}{2} + \frac{\sqrt{\varepsilon_{\mathsf{nm}}}}{4} \le \sqrt{\varepsilon_{\mathsf{nm}}}.$$

   Therefore $y \in \{0,1\}^d \setminus \mathsf{BAD}_{X,\varepsilon_{\mathsf{nm}}}$. Applying Theorem 3.21 yields the result.

2. We first prove $\mathsf{BAD}'_X \subset \mathsf{BAD}_{X,\varepsilon_{\mathsf{nm}}/16}$. If $\mathsf{IsBad}_{X,n,t,k,\varepsilon_{\mathsf{nm}}}(y) = 1$, then for some distinct $y_1, \ldots, y_t$:

$$\Delta\left((h_y(X); U_1)|\{h_{y_i}(X)\}_{i=1}^t\right) > \frac{\sqrt{\varepsilon_{\mathsf{nm}}}}{2} - \frac{\sqrt{\varepsilon_{\mathsf{nm}}}}{4} > \frac{\sqrt{\varepsilon_{\mathsf{nm}}}}{4}.$$

   Since nmExt is a $(t, k, \varepsilon_{\mathsf{nm}}/16)$-non-malleable extractor applying Theorem 3.21 yields the result.

3. It's clear that the only nonuniformity in the algorithm comes from the description of the circuit X. Overall, there are at most $D^t \cdot 2^{2^t}$ iterations of the main loop. In each loop, we apply Theorem 3.8 with error $O\left(\sqrt{\varepsilon_{\mathsf{nm}}}\right)$ on a circuit of size $s = \mathrm{poly}(n^c, n^{c_{\mathsf{error}}})$. By Theorem 3.8 each such iteration can be done in time $\mathrm{poly}(n^c, n^{c_{\mathsf{error}}})$ (for a polynomial that depends on t, c_{nm} and the constants in the hardness assumption).

   Overall (as $D \le n^{c_{\mathsf{nm}} \cdot c_{\mathsf{error}}}$ and t is constant) the entire procedure is computable in time

$$D^t \cdot 2^{2^t} \cdot \mathrm{poly}(n^c, n^{c_{\mathsf{error}}}) = \mathrm{poly}(n^c, n^{c_{\mathsf{error}}})$$

   (where the polynomial depends on t, c_{nm} and the constants in the hardness assumption) and the entire procedure uses n^c bits of advice.

■

6 A "Somewhere Pseudoentropy Generator" for IsBad

In Theorem 5.1 we show that (assuming a hardness assumption) for every distribution X that is samplable by size n^c circuits, there is a procedure IsBad that recognizes whether a seed $y \in \{0,1\}^d$ is a "bad seed" for the considered nice non-malleable extractor nmExt. We have also observed that IsBad is a nonuniform procedure that runs in time $n^{c_{\mathsf{runtime}}}$ using n^c bits of advice, for a constant c_{runtime} that will be set to be much larger than c.

Continuing with the plan explained in Section 2 we would like to use a hardness assumption, and set up a PRG against procedures like IsBad. We have already seen in Theorem 4.3 that under the hardness assumption that E is hard for large exponential time with exponential advice, we can get PRGs for such procedures with seed length that is logarithmic in the amount of advice bits (and does not depend on running time).

However, as explained in Section 2, for our intended application, we would also like that the dependance of the seed length of the PRG on its error ε is $(1+\beta) \cdot \log \frac{1}{\varepsilon}$ for a constant $\beta < 1$ (say $\beta = \frac{1}{3}$). Unfortunately, this is impossible for a PRG. Nevertheless, it is possible for the following weaker variant of a PRG that we call an "evasive pseudoentropy generator" (evasive PEG) that is sufficient for our purposes. A formal definition is given below.

Definition 6.1. *A function $G\colon \{0,1\}^r \to \{0,1\}^d$ is a $(\mu, \varepsilon_{\mathsf{PEG}})$-evasive pseudoentropy generator (evasive PEG) against a class $\mathcal{C}$ if for every $C\colon \{0,1\}^d \to \{0,1\} \in \mathcal{C}$, with $\Pr[C(U_c)=1] \le \mu$, we have that:*

$$\Pr_{y \sim U_r}[C(G(y))=1] \le \varepsilon_{\mathsf{PEG}}.$$

Note that an ε-PRG is by definition, a $(\mu, \mu+\varepsilon)$-evasive PEG for every $0 \le \mu \le 1$. However, an evasive-PEG only needs to satisfy the latter property for some specific (typically very small) μ. This weaker property allows an evasive-PEG to have seed length where the dependence on $\varepsilon_{\mathsf{PEG}}$ is $(1+\beta) \cdot \log \frac{1}{\varepsilon_{\mathsf{PEG}}}$ for $\beta < 1$.

Unfortunately, we do not know how to construct such evasive PEGs. We will therefore settle for an even weaker object, which we call a *somewhere*-evasive-PEG. The definition (that is given below) is inspired by the "somewhere condenser" used by [BDT19] in their two-source extractor construction (as explained in Section 2). Loosely speaking, a somewhere-evasive-PEG gets "A attempts" to succeed in avoiding the set $\{x : C(x)=1\}$. A formal definition is given below.

Definition 6.2. *A function $G\colon \{0,1\}^r \times [A] \to \{0,1\}^d$ is a $(\mu, \varepsilon_{\mathsf{PEG}})$-somewhere-evasive PEG against a class $\mathcal{C}$ if for every $C\colon \{0,1\}^d \to \{0,1\} \in \mathcal{C}$, such that $\Pr[C(U_c)=1] \le \mu$, we have that:*

$$\Pr_{y \sim U_r}[\forall a \in [A] : C(G(y,a))=1] \le \varepsilon_{\mathsf{PEG}}.$$

An evasive-PEG can be thought of as a special case of a somewhere-evasive PEG where $A=1$. In our construction below we will show that under the hardness assumption, there is a somewhere-evasive-PEG where A is a constant (and this will be sufficient for our application later on).

In the theorem below we state our construction of somewhere-evasive PEGs. We do not aim for general parameters, and instead state a theorem where the parameters are tailored for the intended application of constructing extractors for samplable distributions.

Theorem 6.3. *Assume that E is hard for large exponential time with exponential advice. For every constants $c, c_{\mathsf{runtime}} > 0$ and $0 < \beta, \eta < 1$ there is a constant $A = A(\beta, c) \ge 1$ (that does not depend on η and c_{runtime}) such that for every sufficiently large n, and for every $\varepsilon_{\mathsf{PEG}} > 0$ and every integer d that satisfy:*

- *The desired error $\varepsilon_{\mathsf{PEG}} > 0$ is a sufficiently small inverse polynomial in n:*

$$\frac{1}{\operatorname{poly}(n)} \le \varepsilon_{\mathsf{PEG}} \le \left(\frac{1}{n}\right)^{1/\nu}.$$

where c_{PRG} (that depends on c and the constant $\alpha > 0$ from the hardness assumption) is the constant from Theorem 4.3 and $\nu = \frac{\beta}{4c_{\mathsf{PRG}}}$.

- *The desired output length d is a sufficiently large, compared to the desired error $\varepsilon_{\mathsf{PEG}}$:*

$$\left(\frac{1}{\varepsilon_{\mathsf{PEG}}}\right)^{\frac{12}{\eta}} \leq 2^d = D \leq \mathrm{poly}(n).$$

there is an injective function

$$\mathsf{SEPEG}\colon \{0,1\}^{r'=(1+\beta)\log\frac{1}{\varepsilon_{\mathsf{PEG}}}} \times [A] \to \{0,1\}^d$$

That is a $(\mu = D^{-\eta}, \varepsilon_{\mathsf{PEG}})$-somewhere-evasive PEG for nonuniform procedures $B\colon \{0,1\}^d \to \{0,1\}$ running in time $n^{c_{\mathsf{runtime}}}$ with n^c bits of advice. Furthermore, SEPEG is computable in time $\mathrm{poly}(n^c)$.[7]

In particular, for $\beta = 1/3$ this implies that the total number of seeds y' for which $B(\mathsf{SEPEG}(y', a)) = 1$ for every $a \in A$ is at most $(R')^{1/4}$ for $R' = 2^{r'}$.

The theorem statement above is quite technical, and so, we would like to point out the key points: We are aiming to fool nonuniform procedures that run in $n^{c_{\mathsf{runtime}}}$ using n^c bits of advice. We are not aiming for a large stretch and the output length d is only linearly larger than the seed length r. What is important is that (as long as $\varepsilon_{\mathsf{PEG}}$ is sufficiently small inverse polynomial in n, and $D = 2^d$ is a sufficiently large polynomial in n) we obtain a seed length $r = (1+\beta)\cdot\log\frac{1}{\varepsilon_{\mathsf{PEG}}}$ (which is the whole point) with constant A. We achieve this for a parameter $\mu = D^{-\eta} = 2^{-\eta\cdot d}$ which is quite small, but will be sufficient for our purposes. Furthermore, we insist that SEPEG is an injective function (as this will be required when we use SEPEG to construct extractors for samplable distributions).

Proof: The construction closely follows the structure of Theorem 3.1 in [BDT19].

Construction:

Hardness Assumption: We are assuming that E is hard for large exponential size with exponential advice.

Parameters:

- Constants $c, c_{\mathsf{runtime}} \geq 1$ and $0 < \beta, \eta < 1$.
- A sufficiently large n, which determined an output length $d = d(n)$ and an error parameter $\varepsilon_{\mathsf{PEG}} = \varepsilon_{\mathsf{PEG}}(n)$ that satisfy the requirements in Theorem 6.3.

Goal: For $D = 2^d$, construct a $(D^{-\eta}, \varepsilon_{\mathsf{PEG}})$-somewhere-evasive PEG with seed length $r' = (1+\beta)\cdot\log\frac{1}{\varepsilon_{\mathsf{PEG}}}$ and output length d.

Ingredients:

A PRG: We will apply Theorem 4.3 with the constant c, to obtain a constant $c_{\mathsf{PRG}} = c_{\mathsf{PRG}}(c)$. We choose $\bar{c}_{\mathsf{runtime}}$ so that $n^{\bar{c}_{\mathsf{runtime}}} = \frac{n^{c_{\mathsf{runtime}}}}{\varepsilon^4_{\mathsf{PEG}}}$, and note that we can indeed choose such a constant $\bar{c}_{\mathsf{runtime}}$ as we are requiring that $\varepsilon_{\mathsf{PEG}} \geq \frac{1}{\mathrm{poly}(n)}$. We choose $\bar{\varepsilon}_{\mathsf{PRG}} = \varepsilon^{\nu}_{\mathsf{PEG}}$. Recall that $\nu = \frac{\beta}{4c_{\mathsf{PRG}}}$ and note that $\bar{\varepsilon}_{\mathsf{PRG}} = \varepsilon^{\nu}_{\mathsf{PEG}} \leq \frac{1}{n}$, where the last

[7]One may ask whether we can be more explicit about the exponent of the runtime, perhaps as a function of c_{runtime}. In order to keep the assumption in Definition 3.5 as general as possible, we cannot. This is because we make no assumption on how exactly C_{easy} depends on C_{hard}.

inequality follows because we are requiring that $\varepsilon_{\mathsf{PEG}} \leq \left(\frac{1}{n}\right)^{1/\nu}$. Using Theorem 4.3, we obtain a function

$$\bar{G} : \{0,1\}^{\bar{r}=c_{\mathsf{PRG}}\cdot\log(n/\bar{\varepsilon}_{\mathsf{PRG}})} \to \{0,1\}^{n^c}$$

that is an $\bar{\varepsilon}_{\mathsf{PRG}}$-PRG for nonuniform procedures that run in time $n^{c_{\mathsf{runtime}}}$ and use n^c bits of advice. Furthermore, $\bar{G}$ is computable in time $\mathrm{poly}(n^c, n^{c_{\mathsf{runtime}}}, \frac{1}{\bar{\varepsilon}_{\mathsf{PRG}}})$ which is a polynomial in n.

We have that:

$$\begin{aligned}\bar{r} &= c_{\mathsf{PRG}} \cdot \log(n/\bar{\varepsilon}_{\mathsf{PRG}}) \\ &= c_{\mathsf{PRG}} \cdot \log n + \nu \cdot c_{\mathsf{PRG}} \cdot \log(1/\varepsilon_{\mathsf{PEG}}) \\ &= c_{\mathsf{PRG}} \cdot \log n + \frac{\beta}{4} \cdot \log(1/\varepsilon_{\mathsf{PEG}}) \\ &\leq \frac{\beta}{2} \cdot \log(1/\varepsilon_{\mathsf{PEG}}).\end{aligned}$$

Here, the point is that we set $\bar{\varepsilon}_{\mathsf{PRG}}$ to be larger than our desired $\varepsilon_{\mathsf{PEG}}$. More specifically, we took $\bar{\varepsilon}_{\mathsf{PRG}} = \varepsilon_{\mathsf{PEG}}^{\nu}$ for a small constant $\nu > 0$. This means that the seed length $\bar{r}$ of $\bar{G}$ depends on $\bar{\varepsilon}_{\mathsf{PEG}}$ (rather than $\varepsilon_{\mathsf{PEG}}$), and so, is small (by a multiplicative factor of ν) when measured as a function of the target error $\varepsilon_{\mathsf{PEG}}$. In particular when $\varepsilon_{\mathsf{PEG}} \ll \frac{1}{n}$, the seed length $\bar{r}$ is significantly smaller than $1 \cdot \log \frac{1}{\varepsilon_{\mathsf{PEG}}}$ and is in fact $\frac{\beta}{2} \cdot \log(1/\varepsilon_{\mathsf{PEG}})$.

A Disperser: We will apply Theorem 3.11 choosing $\delta = \frac{\beta}{1+\beta}$ to obtain a function

$$\Gamma\colon \{0,1\}^{r'=(1+\beta)\log\frac{1}{\varepsilon_{\mathsf{PEG}}}} \times [A] \to \{0,1\}^{\bar{r}=\frac{\beta}{2}\cdot\log(1/\varepsilon_{\mathsf{PEG}})}$$

such that for $R' = 2^{r'}$ and $\bar{R} = 2^{\bar{r}}$, Γ is a $\left(K = (R')^{\frac{\beta}{1+\beta}}, K' = 2\bar{\varepsilon}_{\mathsf{PRG}}\bar{R}\right)$-disperser. Note that Theorem 3.11 indeed gives $K = (R')^{\frac{\beta}{1+\beta}}$, $\bar{r} = \frac{\delta r'}{2} = \frac{\beta}{2} \cdot \log(1/\varepsilon_{\mathsf{PEG}})$ and choosing $s = 2 \cdot \bar{\varepsilon}_{\mathsf{PRG}}$, we obtain that $K' = 2\bar{\varepsilon}_{\mathsf{PRG}}\bar{R}$ with

$$A = c_\delta \cdot \frac{r'}{\log(1/s)} = \frac{(1+\beta)\log\frac{1}{\varepsilon_{\mathsf{PEG}}}}{\log\frac{1}{2\bar{\varepsilon}_{\mathsf{PRG}}}} = \frac{(1+\beta)\log\frac{1}{\varepsilon_{\mathsf{PEG}}}}{\log\frac{1}{2\varepsilon_{\mathsf{PEG}}^{\nu}}},$$

which is a constant that depends on β and c_{PRG} (and the constant c_{PRG} is determined by c and the constant α in the hardness assumptions). These choices are made so that:

$$\frac{K}{R'} = (R')^{\frac{\beta}{1+\beta}-1} = (R')^{-\frac{1}{1+\beta}} = 2^{-\log(1/\varepsilon_{\mathsf{PEG}})} = \varepsilon_{\mathsf{PEG}}.$$

We also have that Γ is computable in time $\mathrm{poly}(r')$ which is some polynomial in n (as we are requiring that $\varepsilon_{\mathsf{PEG}} \geq \frac{1}{\mathrm{poly}(n)}$).

Definition of the Somewhere-Evasive-PEG: We define:

$$\mathsf{SEPEG}\colon \{0,1\}^{r'=(1+\beta)\log\frac{1}{\varepsilon_{\mathsf{PEG}}}} \times [A] \to \{0,1\}^d$$

as follows: we truncate the output length of $\bar{G}$ to length $d' = d - r' - \log|A|$ (and note that this is possible as by our requirements $d \leq n$ and the output length of G is larger than n). We define:

$$\mathsf{SEPEG}(y', z) = (y', z, G(\Gamma(y', z))).$$

Correctness: It's clear that SEPEG is injective. We now prove that SEPEG is indeed a $(D^{-\eta}, \varepsilon_{\mathsf{PEG}})$-somewhere-evasive PEG against algorithms running in time $n^{c_{\mathsf{runtime}}}$ with n^c bits of advice.

Suppose that $B\colon \{0,1\}^d \to \{0,1\}$ is computable by an $n^{c_{\mathsf{runtime}}}$ algorithm using n^c bits of advice. To simplify the notation we will also use B to also denote the set $\{x : B(x) = 1\} \subseteq \{0,1\}^d$.

If SEPEG is not a $(D^{-\eta}, \varepsilon_{\mathsf{PEG}})$-PEG for B, then $|B| \le D^{1-\eta}$, and yet:

$$\Pr_{y' \sim U_{r'}} \left[\forall z \in [A] : (y', z, \bar{G}(\Gamma(y', z))) \in B\right] > \varepsilon_{\mathsf{PEG}}.$$

Define

$$B' = \{s \in \{0,1\}^{d'} \mid \exists (y', z) \text{ s.t. } (y', z, s) \in B\}.$$

In other words, $B' \subseteq \{0,1\}^{d'}$ is the projection of B onto the last coordinates. We conclude that:

$$\Pr_{y' \sim U_{r'}} \left[\forall z \in [A] : G(\Gamma(y', z)) \in B'\right] > \varepsilon_{\mathsf{PEG}}. \tag{2}$$

We have that

$$R' \cdot A \le (R')^2 = 2^{2r'} = 2^{2\cdot(1+\beta)\cdot \log \frac{1}{\varepsilon_{\mathsf{PEG}}}} \le \frac{1}{\varepsilon_{\mathsf{PEG}}^4}.$$

It follows that B' is recognizable in time $R' \cdot A \cdot n^{c_{\mathsf{runtime}}} \le n^{c_{\mathsf{runtime}}}/\varepsilon_{\mathsf{PEG}}^4 = n^{\bar{c}_{\mathsf{runtime}}}$ with n^c bits of advice. Let $D' = 2^{d'}$ and note that

$$D' = \frac{D}{R' \cdot A} \ge \varepsilon_{\mathsf{PEG}}^4 \cdot D,$$

which gives that

$$(D')^{\eta/2} \ge \varepsilon_{\mathsf{PEG}}^{2\cdot\eta} \cdot D^{\eta/2} \ge \varepsilon_{\mathsf{PEG}}^2 \cdot D^{\eta/2} \ge \varepsilon_{\mathsf{PEG}}^2 \cdot \left(\frac{1}{\varepsilon_{\mathsf{PEG}}}\right)^{\frac{12}{\eta}\cdot\frac{\eta}{2}} \ge \frac{1}{\varepsilon_{\mathsf{PEG}}^4}. \tag{3}$$

We use (3) to conclude that:

$$|B'| \le |B| = D^{1-\eta} = (D' \cdot R' \cdot A)^{1-\eta} \le (D')^{1-\eta} \cdot R' \cdot A \le \frac{(D')^{1-\eta}}{\varepsilon_{\mathsf{PEG}}^4} \le (D')^{1-\frac{\eta}{2}}$$

Let $\bar{B} = \left\{y \in \{0,1\}^{\bar{r}} : \bar{G}(y) \in B'\right\}$. We have chosen the parameters of $\bar{G}$ so that

$$\frac{|\bar{B}|}{\bar{R}} = \Pr_{y \sim U_{\bar{r}}}[y \in \bar{B}] = \Pr_{y \sim U_{\bar{r}}}[\bar{G}(y) \in B'] \le \frac{|B'|}{D'} + \bar{\varepsilon}_{\mathsf{PRG}} \le (D')^{-\eta/2} + \bar{\varepsilon}_{\mathsf{PRG}} < 2\bar{\varepsilon}_{\mathsf{PRG}},$$

where the last inequality follows using (3) and recalling that $\bar{\varepsilon}_{\mathsf{PRG}} = \varepsilon_{\mathsf{PEG}}^{\nu} > \varepsilon_{\mathsf{PEG}}$.

We have that $|\bar{B}| < 2 \cdot \bar{\varepsilon}_{\mathsf{PRG}} \cdot |\bar{R}| = K'$. By the disperser property of Γ, and using (2), it follows that

$$\Pr_{y' \sim U_{r'}}[\forall z \in [A] : \Gamma(y', z) \in \bar{B}] = \Pr_{y' \sim U_{r'}}[\forall z \in [A] : \bar{G}(\Gamma(y', z)) \in B'] > \varepsilon_{\mathsf{PEG}} = \frac{K}{R'}.$$

However, this is a contradiction to the disperser property, as we have seen that $|\bar{B}| \le K'$. ■

7 The Extractor Construction

In this section we prove Theorem 1.2 asserting that under the hardness assumption that E is hard for large exponential time with exponential advice, there is a reduction that reduces the problem of constructing extractors for samplable distributions to the task of constructing nice non-malleable extractors.

We start by restating Theorem 1.2 in a formal way, relying on the precise definition on nice non-malleable extractors give in Definition 3.17.

Theorem 7.1. *Assume that:*

- E *is hard for large exponential time with exponential advice.*
- *There is a nice construction of non-malleable extractors for min-entropy k, for some function $k(t, n, \varepsilon_{\mathsf{nm}})$.*

Then for any constants $c \geq 1$ and $0 < \varepsilon \leq 1$, there exist constants t, c_{error} (where t depends on c, ε, and c_{error} depends on c, ε and α from the hardness assumption) such that for every sufficiently large n, there is a (k, ε)-extractor

$$\mathsf{Ext}\colon \{0,1\}^n \to \{0,1\}$$

for distributions samplable by circuits of size n^c, for $k = k(t, n, n^{-c_{\mathsf{error}}})$. Furthermore, the extractor runs in time $\mathrm{poly}(n^c)$.[8]

Before proceeding with the proof of Theorem 7.1, we note that using Theorem 7.1, together with the nice construction of non-malleable extractors specified in Theorem 3.19 immediately yields Theorem 1.3.

High level intuition for the proof of Theorem 7.1. The proof of Theorem 7.1 imitates the structure of the proof of [BDT19] (replacing information theoretic components with their analogues, as explained in Section 2). The actual proof is quite technical and involves many parameters. Loosely speaking, the construction of the extractor for samplable distribution $\mathsf{Ext} : \{0,1\}^n \to \{0,1\}$, works as follows:

- Let $\mathsf{nmExt} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}$ be a "nice" non-malleable extractor.
- We use the hardness assumption to set up the somewhere-evasive-PEG,

  $$\mathsf{SEPEG} : \{0,1\}^{r'} \times [A] \to \{0,1\}^d$$

  of Theorem 6.3 to fool the procedure IsBad from Section 5 (which checks whether as seed $y \in \{0,1\}^d$ is bad).
- Given input $x \in \{0,1\}^n$ and $y' \in \{0,1\}^{r'}$, we define:

  $$\oplus\mathsf{NM}(x, y') = \bigoplus_{z \in [A]} \mathsf{nmExt}(x, \mathsf{SEPEG}(y', z)).$$

[8] As before, we cannot specify the exact dependence of the exponent of the polynomial on α, c, ε, because we mke no assumption on the relationship between C_{hard} and C_{easy} in Definition 3.5. However, if $C_{\mathsf{easy}} = O\left(C_{\mathsf{hard}}\right)$, then the runtime is $n^{\mathrm{poly}(1/\alpha, c, 1/\varepsilon)}$.

- The output is given by: $\mathsf{Ext}(x) = \mathrm{Maj}_{y' \in \{0,1\}^r} \oplus \mathsf{NM}(x, y')$.

The analysis shows that:

- For every samplable distribution X with min-entropy k, the number of $y' \in \{0,1\}^{r'}$ that are "bad" in the sense that for every $z \in [A]$, the string $y = \mathsf{SRPEG}(y', z)$ is a bad seed for nmExt on the source X, is small. Here, it will be crucial that this number is less than $(R')^{1/2}$ for $R' = 2^{r'}$. This follows as the seed length of SEPEG is $r' = (1+\beta) \cdot \log \frac{1}{\varepsilon_{\mathsf{PEG}}}$ and we can choose, say $\beta = 1/3$.
- For every $y' \in \{0,1\}^{r'}$ which is "good", one can show that $\oplus\mathsf{NM}(X, y')$ is (close to) uniform. Loosely speaking, this follows using non-malleability, and that for a good y', there exists a $z^* \in [A]$ such that $y = \mathsf{SRPEG}(y', z^*)$ is a good seed, meaning that $\mathsf{nmExt}(X, y)$ is close to uniform (even conditioned on the output of nmExt on seeds of the form $\mathsf{SEPEG}(y', z)$ for $z \neq z^*$).
- The R' input bits that we feed to Maj, are an oblivious bit-fixing source of the type described in Theorem 3.14. This relies on the previous item and non-malleability (as well as the aforementioned bound on the number of bad y'). It follows that the function Maj outputs a bit that is close to uniform.

The exact details follow appear in the formal proof below.

Proof: The construction follows the structure of Theorem 5.1 in [BDT19].

Construction:

Hardness Assumption: We are assuming that E is hard for large exponential size with exponential advice.

"Input" Parameters:

- Constants $c > 0$, $0 < \varepsilon < 1$.
- An n that is sufficiently large according to the requirements of a nice non-malleable extractor in Definition 3.17 (for constants t and c_{error} to be chosen shortly), and the somewhere-evasive PEG in Theorem 6.3 (for constants c_{runtime}, β, η to be chosen shortly)
- Min-entropy parameter $k = k(t, n, \varepsilon_{\mathsf{nm}})$, determined by the construction of the nice non-malleable extractor.

Goal: Construct $\mathsf{Ext}\colon \{0,1\}^n \to \{0,1\}$ such that for every $X\colon \{0,1\}^{n^c} \to \{0,1\}^n$ computable by a circuit of size n^c such that $H_\infty(X) \ge k$, $\left|\Pr[\mathsf{Ext}(X) = 1] - \frac{1}{2}\right| \le \varepsilon$.

Ingredients:

Some Initial Parameter Choices: There are several parameters to consider when combining Definition 3.17 and Theorem 6.3, with many dependencies. We first need to define some inital parameters before defining the objects that depend on them:

- Define $t' = \frac{9c_{\mathrm{Maj}}^2}{\varepsilon^2}$. For c_{Maj} according to Theorem 3.14.
- In Theorem 6.3, define $\beta = 1/3$.

- We now have well-defined constant $A = A(\beta, c)$ from Theorem 6.3.
- We define $t = t' \cdot A$.

A Nice Non-malleable Extractor: According to Definition 3.17, for our choice of constant t, we have a constant $c_{\mathsf{nm}} = c_{\mathsf{nm}}(t)$ such that for every polynomially small error, $\varepsilon_{\mathsf{nm}} = n^{-c_{\mathsf{error}}}$, there is a $(t, k, \varepsilon_{\mathsf{nm}}/16)$-non-malleable extractor:

$$\mathsf{nmExt}\colon \{0,1\}^n \times \{0,1\}^d \to \{0,1\}$$

such that for sufficiently large n: $d \leq c_{\mathsf{nm}} \log n/\varepsilon_{\mathsf{nm}}$ computable in time $\left(\frac{n}{\varepsilon_{\mathsf{nm}}}\right)^{c_{\mathsf{nm}}}$.

A Somewhere-Evasive PEG: According to Theorem 6.3, for our choice of $\beta = 1/3$, and by choosing $\eta = \frac{1}{4c_{\mathsf{nm}}}$, for any c_{runtime}, and sufficiently large n, if we set

$$\varepsilon_{\mathsf{PEG}} = \left(\frac{1}{n}\right)^{1/\nu} = n^{-12c_{\mathsf{PRG}}},$$

where $c_{\mathsf{PRG}} = c_{\mathsf{PRG}}(c)$ is the constant from Theorem 4.3, then for any desired output length d, as long as:

$$\left(\frac{1}{\varepsilon_{\mathsf{PEG}}}\right)^{\frac{12}{\eta}} = n^{48\cdot c_{\mathsf{PRG}}\cdot c_{\mathsf{nm}}} \leq 2^d = D \leq \left(\frac{n}{\varepsilon_{\mathsf{nm}}}\right)^{c_{\mathsf{nm}}} \leq \mathrm{poly}(n)$$

there is an injective function computable in time $\mathrm{poly}(n)$

$$\mathsf{SEPEG}\colon \{0,1\}^{r'=(1+\beta)\log 1/\varepsilon_{\mathsf{PEG}}} \times [A] \to \{0,1\}^d$$

that is a $(D^{-1/(4c_{\mathsf{nm}})}, \varepsilon_{\mathsf{PEG}})$-somewhere-evasive PEG for algorithms $B\colon \{0,1\}^d \to \{0,1\}$ running in time $n^{c_{\mathsf{runtime}}}$ with n^c bits of advice. Notice we have now:

$$R' = 2^{r'} = \left(\frac{1}{\varepsilon_{\mathsf{PEG}}}\right)^{1+\beta} = n^{16c_{\mathsf{PRG}}}.$$

The Final Parameters, $\varepsilon_{\mathsf{nm}}$, c_{error}, and c_{runtime}: . We finally set $\varepsilon_{\mathsf{nm}}$ sufficiently small to satisfy the constraints of both Corollary 3.15 and Theorem 6.3. Namely:

$$\begin{aligned}\varepsilon_{\mathsf{nm}} &= \min\left(\frac{\varepsilon^2}{(15tc_{\mathrm{Maj}}(R')^t)^2}, \left(\frac{1}{\varepsilon_{\mathsf{PEG}}}\right)^{\frac{12}{\eta}}\right)\\ &= \min\left(\frac{\varepsilon^2}{225t^2c_{\mathrm{Maj}}^2}\cdot n^{\frac{288c_{\mathsf{PRG}}\cdot c_{\mathrm{Maj}}}{\varepsilon^2}}, n^{48\cdot c_{\mathsf{PRG}}\cdot c_{\mathsf{nm}}}\right) = n^{-c_{\mathsf{error}}}.\end{aligned}$$

In particular this implies that $\varepsilon_{\mathsf{nm}} \leq \frac{\varepsilon^2}{(15tc_{\mathrm{Maj}}R^t)^2}$, so that for $\gamma = 5t\sqrt{\varepsilon_{\mathsf{nm}}}$, we have $\gamma \leq \frac{\varepsilon}{3\cdot c_{\mathrm{Maj}}\cdot R^t}$ as required by Corollary 3.15. Additionally, since any extractor must have $d \geq \log 1/\varepsilon_{\mathsf{nm}}$, we also have:

$$n^{48\cdot c_{\mathsf{PRG}}\cdot c_{\mathsf{nm}}} \leq \frac{1}{\varepsilon_{\mathsf{nm}}} \leq D = \mathrm{poly}(n)$$

as required by Theorem 6.3. Thus, we set the output length of SEPEG to d. Finally, since $\mathsf{nmExt}\colon \{0,1\}^n \times \{0,1\}^d \to \{0,1\}$ is a $(t, k, \varepsilon_{\mathsf{nm}}/16)$-non-malleable extractor, we let c_{runtime} be the exponent such that the final (polynomial) runtime of $\mathsf{IsBad}_{X,n,t,k,\varepsilon_{\mathsf{nm}}}$ is $n^{c_{\mathsf{runtime}}}$, according to Theorem 5.1.

Definition of the Extractor: Given $x \sim X$, we define $\mathsf{Ext}(x)$ as follows.

- Ext first computes an $[R'] \times [A]$ table TABLE where the (y', z)-th entry is $h_{y',z}(x) = \mathsf{nmExt}(x, \mathsf{SEPEG}(y', z))$.
- It then computes, for every $y' \in [R']$, $\oplus\mathsf{NM}(x, y') = \bigoplus_{z\in[A]} \mathsf{TABLE}[y'][z]$
- Finally, it outputs $\mathrm{Maj}\left(\oplus\mathsf{NM}(x, 1), \ldots, \oplus\mathsf{NM}(x, R')\right)$

One can verify that the algorithm runs in time $\mathrm{poly}(n)$.

Correctness: Let $k = k(n, t, \varepsilon_{\mathsf{nm}})$ for the parameters n, t and $\varepsilon_{\mathsf{nm}}$ chosen above. Note that by our choices, t is a constant that depends on c, ε, and $\varepsilon_{\mathsf{nm}} = n^{-c_{\mathsf{error}}}$ where c_{error} is a constant that depends on c, ε, and on the constant $\alpha > 0$ in the hardness assumption.

Let X be a distribution over $\{0, 1\}^n$ that is samplable by circuits of size n^c, such that $H_\infty(X) \geq k$. First, we see, from Theorem 5.1 that:

$$|\mathsf{BAD}'_X| = |\{y \mid \mathsf{IsBad}_{X,n,t,k,\varepsilon_{\mathsf{nm}}}(y) = 1\}| \leq \sqrt{\varepsilon_{\mathsf{nm}}}D.$$

However, we can show that $\sqrt{\varepsilon_{\mathsf{nm}}}D \leq D^{1-\frac{1}{4c_{\mathsf{nm}}}}$. Indeed using the fact that $n \leq 1/\varepsilon_{\mathsf{nm}}$, we can show $\sqrt{\varepsilon_{\mathsf{nm}}} \leq D^{-\frac{1}{4c_{\mathsf{nm}}}}$ since:

$$D^{\frac{1}{4c_{\mathsf{nm}}}} \leq \left(\left(\frac{n}{\varepsilon_{\mathsf{nm}}}\right)^{c_{\mathsf{nm}}}\right)^{\frac{1}{4c_{\mathsf{nm}}}} \leq \varepsilon_{\mathsf{nm}}^{1/2}.$$

Thus $\Pr[\mathsf{IsBad}_{X,n,t,k,\varepsilon_{\mathsf{nm}}}(U_d) = 1] \leq D^{-\eta}$. Therefore indeed by Theorem 6.3, for all but $\varepsilon_{\mathsf{PEG}}R' \leq (\sqrt{R'})^{1/4}$ seeds y', there is a z such that $\mathsf{SEPEG(y', z)} \notin \mathsf{BAD}'_X$. Call a row y' good if there is such a good z for it. Since nmExt is a $(t = t' \cdot A, k, \varepsilon_{\mathsf{nm}}/16)$-non-malleable extractor, once again by Theorem 5.1, for any good row y', and any good z^* for it, $h_{y',z^*}(X)$ is close to a uniform bit, even conditioned on $h_{y',z}(X)$ for all $z \neq z^*$, and $h_{y'_i,z}(X)$ for any choice of $t-1$ other seeds $y'_1, \ldots, y'_{t-1}$ and any $z \in [A]$. Formally:

$$\Delta\left((h_{y',z^*}(X); U_1) \mid \{h_{y',z}(X)\}_{z\neq z^*}, \{h_{y'_i,z}(X)\}_{i\in[t-1],z\in A}\right) \leq \sqrt{\varepsilon_{\mathsf{nm}}}.$$

Notice here we used the fact that SEPEG is injective, and so every entry in the table represents the evaluation of nmExt on X and a different seed. We will use this fact to prove that $\oplus\mathsf{NM}(x, 1), \ldots, \oplus\mathsf{NM}(x, R')$ is a $\left((R')^{1/4}, t, \gamma\right)$ non-oblivious bit-fixing source (on R' bits) for $\gamma = 5t\sqrt{\varepsilon_{\mathsf{nm}}} = \frac{\varepsilon}{3c_{\mathrm{Maj}}(R')^t}$. The proof now proceeds identically to that in [BDT19]. Towards this end, fix t good seeds (rows) $y'_1, \ldots, y'_t$, with corresponding good $z_1, \ldots, z_t$. By Lemma 3.1, the good entries $(y'_1, z_1), \ldots, (y'_t, z_t)$ look uniform and independent of each other, even conditioned on all other entries in the rows of $y'_1, \ldots, y'_t$:

$$\Delta\left((h_{y'_1,z_1}(X), \ldots, h_{y'_t,z_t}(X); U_t) \mid \{h_{y'_i,z}(X)\}_{(y'_i,z)\notin\{(y'_1,z_1),\ldots,(y'_t,z_t)\}}\right) \leq 5t\sqrt{\varepsilon_{\mathsf{nm}}} = \gamma.$$

Therefore, the XOR of these rows must still look uniform and independent of each other:

$$\Delta(\oplus\mathsf{NM}(x, y'_1), \ldots, \oplus\mathsf{NM}(x, y'_t); U_t) \leq \gamma.$$

This proves that $\oplus\mathsf{NM}(x, 1), \ldots, \oplus\mathsf{NM}(x, R')$ is a $\left((R')^{1/4}, t, \gamma\right)$ non-oblivious bit-fixing source, where the "good" bits correspond to good rows, and the total number of bad rows is at most $(R')^{1/4}$. Applying Corollary 3.15 yields the result. ■

8 Nice Seeded Non-Malleable Extractors for Samplable Distributions

In this section we prove Theorem 1.4, which we restate below in a more precise way:

Theorem 8.1. *Assume that:*

- E *is hard for large exponential time with exponential advice.*
- E *is hard for exponential size nondeterministic circuits.*

Then, for every constants $c > 1$ and $\varepsilon > 0$, there exists a constant $C = C(c, \varepsilon)$ such that for every sufficiently large n, there is a $(k = C \cdot \log n \cdot \log \log n, \varepsilon)$-extractor

$$\mathsf{Ext} \colon \{0,1\}^n \to \{0,1\}$$

for distributions samplable by circuits of size n^c. Furthermore, the extractor runs in time $\mathrm{poly}(n^c)$.

In Theorem 8.1 we are assuming the additional assumption that E is hard for exponential size nondeterministic circuits. Loosely speaking, our plan is to use this assumption to convert the current known explicit constructions of non-malleable extractors [Li17] (and recall, that as explained in Section 1, these are not sufficiently nice for our purpose) into ones that *are* nice, and can be used in Theorem 7.1. More specifically, recall the best known constructions of non-malleable extractors for small min-entropy k [Li17] (stated in Theorem 3.20) have a seed length that is slightly too large for our purposes, and we would like to shorten their seed length to $O(\log n/\varepsilon_{\mathsf{nm}})$.

The key observation is that we can settle for a weaker extractor than a general non-malleable extractor. More specifically, it is sufficient for our purposes that the provided non-malleable extractor is *only guaranteed to work on samplable distributions*. This follows as when using Theorem 3.21, we only consider the case that X is a samplable distribution.

This means that it is sufficient for our purposes to have a "nice construction of non-malleable extractors for *samplable distributions*" defined below. This definition is similar to Definition 3.17, except that we restrict our attention to samplable distributions.

Definition 8.2 (Nice construction of non-malleable extractors for samplable distributions)**.** *Let $k(t, n, \varepsilon_{\mathsf{nm}})$ be some integer function. We say that "there is a nice construction of non-malleable extractors for distributions samplable by circuits of size n^c of min-entropy k" if for every constant t, there exists a constant c_{nm} (that can depend on c and t) such that for every constant $c_{\mathsf{error}} \geq 1$, and for every sufficiently large n, setting $\varepsilon_{\mathsf{nm}}(n) = n^{-c_{\mathsf{error}}}$, there is a function $d(n) \leq c_{\mathsf{nm}} \cdot \log \frac{n}{\varepsilon_{\mathsf{nm}}(n)}$ and a function*

$$\mathsf{nmExtComp}\, \{0,1\}^n \times \{0,1\}^d \to \{0,1\}$$

computable in time $(\frac{n}{\varepsilon_{\mathsf{nm}}(n)})^{c_{\mathsf{nm}}}$, such that for every distribution X samplable by a size n^c circuit of min-entropy k, the conclusion of Theorem 3.21 holds.

We note that we do not define a nice non-malleable extractor for samplable distributions according to the original Definition 3.16. This is mainly for convenience as the property we have used throughout this work is that of Theorem 3.21. We end this discussion by recording the following relevant observation.

Observation 3. *Theorem 5.1 holds even when there is only a nice construction* $\mathsf{nmExtComp}$ *of non-malleable extractors for distributions samplable by circuits of size n^c of min-entropy $k(t, n, \varepsilon_{\mathsf{nm}})$. The same is true for Theorem 7.1.*

The point here, is that constructing non-malleable extractors for samplable distributions is easier than improving the best known constructions of non-malleable extractors (because we only need our construction to work for samplable distributions), and is easier than constructing extractors for samplable distribtions (because we can use an independent seed). In fact, we can show that we can construct such $\mathsf{nmExtComp}$ by simply using the best known non-malleable extractor construction nmExt, and considering $\mathsf{nmExtComp}(x, y) = \mathsf{nmExt}(x, G(y'))$ for an appropriately chosen PRG G. If the seed length of G has the proper dependence on $\varepsilon_{\mathsf{nm}}$ then we are done. As we will explain below, it is sufficient that G is a PRG for nondeterministic circuits.

We start by defining an "almost nice" construction of non-malleable extractors, which are the same as nice non-malleable extractors, except the constraint on seed length is greatly reduced. (Loosely speaking we now allow seed length $d = (n/\varepsilon_{\mathsf{nm}})^{O(1)}$, whereas in Definition 3.17 we required that $d = O(\log(\frac{n}{\varepsilon_{\mathsf{nm}}}))$).

Definition 8.3. *Let $k(t, n, \varepsilon_{\mathsf{nm}})$ be some integer function. We say that "there is an almost-nice construction of non-malleable extractors for min-entropy k" if for every constant t, there exists a constant c_{nm} such that for every constant $c_{\mathsf{error}} \geq 1$, and for every sufficiently large n, setting $\varepsilon_{\mathsf{nm}}(n) = n^{-c_{\mathsf{error}}}$, there is a function $d(n) \leq (\frac{n}{\varepsilon_{\mathsf{nm}}(n)})^{c_{\mathsf{nm}}}$ and a $(t, k(t, n, \varepsilon_{\mathsf{nm}}), \varepsilon_{\mathsf{nm}}(n))$-non-malleable extractor $\mathsf{nmExt} : \{0,1\}^n \times \{0,1\}^{d(n)} \to \{0,1\}$, such that nmExt can be computed in time $(\frac{n}{\varepsilon_{\mathsf{nm}}(n)})^{c_{\mathsf{nm}}}$.*

The next theorem asserts that under a suitable hardness assumption, an almost nice construction of non-malleable extractors (for general distributions) yields a nice construction of non-malleable extractors for samplable distributions.

Theorem 8.4. *Assume that:*

- E *is hard for exponential size nondeterministic circuits.*
- *There is a construction of almost-nice non-malleable extractors for min-entropy k.*

Then for any constant c, there is a nice construction of non-malleable extractors for distributions samplable by circuits of size n^c of min-entropy k.

High level idea for the proof of Theorem 8.4. We will use Theorem 3.9, and the hardness assumption to obtain a PRG G for nonbdeterministic circuits. The main goal is to show that the subset of seeds to the non-malleable extractor in Theorem 3.20 chosen by this PRG preserves the properties of Theorem 3.21 whenever X is samplable by a circuit of size n^c. It turns out, that the ideas needed to do so are already in Section 5 and Section 6. We'll show that as long as for $\varepsilon_{\mathsf{nm}} = \frac{1}{\mathrm{poly}(n)}$, and the seed length of a non-malleable extractor is sufficiently short ($\mathrm{poly}(n)$), we can recognize the set of bad seeds with a *nondeterministic* circuit of size $\mathrm{poly}(n)$. (Note that in Section 5 we argued that IsBad can be implemented in time roughly $2^{d \cdot t}$, where this time came from a brute force search over all t tuples of seeds in $\{0,1\}^d$). The key observation is that while for $d \gg \log n$, we can not afford a brute force search over all such seeds (or t-tuples of seeds) in polynomial time, one can use nondeterminism to guess the right t seeds, if they exist. More concretely, we can change the procedure IsBad to instead guess the seeds $y_1, \ldots, y_t$ rather than brute force over all tuples. This leads to an implementation of IsBad by a $\mathrm{poly}(n)$ size nondeterministic circuit, which we can fool using the PRG G.

The precise argument proving Theorem 8.4 appears in Section 8.1. Observing that the non-malleable extractor from Theorem 3.20 is almost-nice, we get the following result:

Theorem 8.5. *Assume that* E *is hard for exponential size nondeterministic circuits. For some constant* $C = C(t)$*, for every constant* c*, there is a nice construction of non-malleable extractors for distributions samplable by circuits of size* n^c *of min-entropy* $k = C \cdot (\log n + \log 1/\varepsilon_{\mathsf{nm}} \cdot \log\log /\varepsilon_{\mathsf{nm}})$.

Proof: In Theorem 8.4, we use Theorem 3.20 as the almost-nice non-malleable extractor. Thus, for any desired constants t, c_{error}, and $\varepsilon_{\mathsf{nm}} = n^{-c_{\mathsf{error}}}$, we have constant $c_{\mathsf{Li}} = c_{\mathsf{Li}}(t)$ and utilize a $(t, k = d', \varepsilon_{\mathsf{nm}}/64)$-non-malleable extractor:

$$\mathsf{nmExt}\colon \{0,1\}^n \to \{0,1\}^{d' \le 16 c_{\mathsf{Li}}\left(\log n + \log \frac{1}{\varepsilon_{\mathsf{nm}}} \cdot \log\log \frac{1}{\varepsilon_{\mathsf{nm}}}\right)} \to \{0,1\}$$

Seting $C(t) = 16c_{\mathsf{Li}}$ and applying Theorem 8.4 yields the result. ∎

Equipped with this construction, and Observation 3, we can prove Theorem 8.1

Proof: (Proof of Theorem 8.1) The proof is identical to that of Theorem 7.1, and we only sketch the derivation of the constant C. By Theorem 8.5, for some constant $C' = C'(t)$, we have a nice construction of non-malleable extractors for distributions samplable by circuits of size n^c of min-entropy $k = C'(t) \cdot (\log n + \log 1/\varepsilon_{\mathsf{nm}} \cdot \log\log /\varepsilon_{\mathsf{nm}})$. In Theorem 7.1, we set $t = t' \cdot A = t(\varepsilon, c)$, and $\varepsilon_{\mathsf{nm}} = n^{-c_{\mathsf{error}}}$ for $c_{\mathsf{error}} = c_{\mathsf{error}}(t, c_{\mathsf{PRG}}, c_{\mathsf{nm}})$, where $c_{\mathsf{PRG}}, c_{\mathsf{nm}}$ are also functions of c and t. Thus setting $C = C' \cdot c_{\mathsf{error}}^2$ yields the result. ∎

8.1 Proof of Theorem 8.4

The proof will follow the high level explanation in Section 8. We start by describing the nondeterministic version of the procedure IsBad explicitly, and provide an analogous result to Theorem 5.1. Since we are not concerned with fooling the nondeterministic version of IsBad with a PRG of seed length independent of d (the seed length of nmExt) we only measure the complexity of IsBad by its size as a nondeterministic circuit. In other words, we do not need to take care to separate time and advice in this analysis, as this will happen ultimately later on. We now state a theorem that shows that Theorem 5.1 holds even for this nondeterministic version.

Theorem 8.6. *Assume that:*

- E *is hard for exponential size circuits.*
- *There is a construction of almost-nice non-malleable extractors for min-entropy* k.

Then:

Let $t, c, c_{\mathsf{error}} \ge 1$ *be constants. Let* n *be sufficiently large. Set* $\mathsf{nmExt} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}$ *to be the* $(t, k, \varepsilon_{\mathsf{nm}}/16)$*-non-malleable extractor from the almost-nice construction, where* $\varepsilon_{\mathsf{nm}} = n^{-c_{\mathsf{error}}}$*, with* $d \le (\frac{n}{\varepsilon_{\mathsf{nm}}(n)})^{c_{\mathsf{nm}}}$.

For any distribution X *over* $\{0,1\}^n$ *that is samplable by a size* n^c *circuit, and has* $H_\infty(X) \ge k$*, let* $\mathsf{NonDetIsBad}_{X,n,t,k,\varepsilon_{\mathsf{nm}}}(y)$ *be the procedure in Figure 2 and let*

$$\mathsf{BAD}'_X = \{y \mid \exists w \text{ s.t. } \mathsf{NonDetIsBad}_{X,n,t,k,\varepsilon_{\mathsf{nm}}}(y, w) = 1\}.$$

Figure 2: A nondeterministic procedure that recognizes bad seeds

```
Procedure NonDetIsBad$_{X,n,t,k,\varepsilon_{\mathsf{nm}}}$:

Ingredients: The procedure utilizes as a subroutine an almost-nice $(t, k, \varepsilon_{\mathsf{nm}}/16)$-non-malleable
    extractor $\mathsf{nmExt}\colon \{0,1\}^n \times \{0,1\}^d \to \{0,1\}$. Define $h_y(x) = \mathsf{nmExt}(x, y)$.

Input: $y \in \{0,1\}^d$, a seed to nmExt, and $w = (y_1, \ldots, y_t) \in (\{0,1\}^d)^t$, a choice of $t$ other seeds to
    nmExt.

Advice: A description of a circuit $X$ of size $n^c$.

Output: A bit $b \in \{0,1\}$.

Operation:

  1. Check that $y, y_1, \ldots, y_t$ are all distinct. If not, break about output $b = 0$.
     (a) For every subset $S \subset \{0,1\}^t$:
         i. Let $\mathbf{1}_S\colon \{0,1\}^t \to \{0,1\}$ denote the algorithm determining membership in $S$.
         ii. Define $A_1\colon \{0,1\}^{n^c} \to \{0,1\}$ as:
                 $A_1(z) = \mathbf{1}_S(h_y(X(z)), h_{y_1}(X(z)), \ldots, h_{y_t}(X(z)))$.
             Also define $A_2\colon \{0,1\}^{n^c} \times \{0,1\} \to \{0,1\}$ as:
                 $A_2(z, a) = \mathbf{1}_S(a, h_{y_1}(X(z)), \ldots, h_{y_t}(X(z)))$.
         iii. Note that $A_1, A_2$ are circuits of size $s = t \cdot n^c \cdot (\frac{n}{\varepsilon_{\mathsf{nm}}})^{c_{\mathsf{nm}}} = \mathrm{poly}(n^c, n^{c_{\mathsf{error}}})$, for a polyno-
              mial that depends on $t, c_{\mathsf{nm}}$. We will use the algorithm from Theorem 3.8, compute $\widetilde{p}_1$
              and $\widetilde{p}_2$, that are $\sqrt{\varepsilon_{\mathsf{nm}}}/8$ approximations to $\Pr[A_1(U_{n^c}) = 1]$ and $\Pr[A_2(U_{n^c+1}) = 1]$
              respectively. By Theorem 3.8 this can be done in time $\mathrm{poly}(s, \frac{1}{\varepsilon_{\mathsf{nm}}}) = \mathrm{poly}(n^c, n^{c_{\mathsf{error}}})$
              for a polynomial that depends on $t$, $c_{\mathsf{nm}}$, and the constants in the hardness assump-
              tion.
         iv. If $|\widetilde{p}_1 - \widetilde{p}_2| > \sqrt{\varepsilon_{\mathsf{nm}}}/2$, then break and output $b = 1$.
  2. Output $b = 0$.
```

1. $\mathsf{BAD}_{X,\varepsilon_{\mathsf{nm}}} \subset \mathsf{BAD}'_X$ *and in particular, for any* $y \notin \mathsf{BAD}'_X$*, and any distinct* $y_1, \ldots, y_t$*:*

$$\Delta\left((h_y(X); U_1) | \{h_{y_i}(X)\}_{i=1}^t\right) \le \sqrt{\varepsilon_{\mathsf{nm}}}.$$

Furthermore, for any $y_1, \ldots, y_t \notin \mathsf{BAD}'_X$*:*

$$\Delta(\{h_{y_i}\}_{i=1}^t; U_t) \le 5t\sqrt{\varepsilon_{\mathsf{nm}}}.$$

2. $\mathsf{BAD}'_X \subset \mathsf{BAD}_{X,\varepsilon_{\mathsf{nm}}/16}$ *and in particular,* $|\mathsf{BAD}'_X| \le \sqrt{\varepsilon_{\mathsf{nm}}} D$.

3. $\mathsf{NonDetIsBad}_{X,n,t,k,\varepsilon_{\mathsf{nm}}}$ *is computable by a nondeterministic circuit of size* $\mathrm{poly}(n^c, n^{c_{\mathsf{error}}} = \frac{1}{\varepsilon_{\mathsf{nm}}})$ *(where the polynomial depends on* t *and* $c_{\mathsf{nm}}(t)$*).*

Proof: The proof of the first two items is identical to that in Theorem 5.1. The final item follows from the fact that $\mathsf{NonDetIsBad}_{X,n,t,k,\varepsilon_{\mathsf{nm}}}(y,w)$ is a nondeterministic algorithm of size $\mathrm{poly}(n^c, n^{c_{\mathsf{error}}})$, and the input length is $|y| + |w| = (t+1)(\frac{n}{\varepsilon_{\mathsf{nm}}(n)})^{c_{\mathsf{nm}}}$. ■

We clarify that we only use Theorem 8.6 in order to show how to construct a nice non-malleable extractor for distributions samplable by circuits of size n^c by combining with a proper PRG.[9] That is, we wil show how to combine an almost-nice non-malleable extractor construction with a PRG to obtain a nice construction of non-malleable extractors for samplable distributions. Using Theorem 3.9, we can show that there is a PRG that fools NonDetIsBad.

We are finally ready to prove Theorem 8.4.

Proof: (of Theorem 8.4)

Construction:

Hardness Assumption: We are assuming that E is hard for exponential size nondeterministic circuits.

Parameters:

- Constants $t, c, c_{\mathsf{error}} > 0$.
- An n that is sufficiently large.

Goal: Construct $\mathsf{nmExtComp}\colon \{0,1\}^n \times \{0,1\}^d \to \{0,1\}$ such that for some constant c_{nm}, and for $\varepsilon_{\mathsf{nm}} = n^{-c_{\mathsf{error}}}$:

- $d \le c_{\mathsf{nm}} \log n/\varepsilon_{\mathsf{nm}}$.
- nmExtComp is computable in time $\left(\frac{n}{\varepsilon_{\mathsf{nm}}}\right)^{c_{\mathsf{nm}}}$.
- For every distribution X samplable by a size n^c circuit, the conclusion of Theorem 3.21 holds for nmExtComp.

Ingredients:

The Non-malleable Extractor: For our desired t, and $\varepsilon_{\mathsf{nm}} = n^{-c_{\mathsf{error}}}$, we have for constant $c'_{\mathsf{nm}} = c'_{\mathsf{nm}}(t)$ a $(t, k, \varepsilon_{\mathsf{nm}}/64)$-non-malleable extractor

$$\mathsf{nmExt}\colon \{0,1\}^n \to \{0,1\}^{d' \le (\frac{n}{\varepsilon_{\mathsf{nm}}(n)})^{c'_{\mathsf{nm}}}} \to \{0,1\}$$

A PRG Fooling Nondeterministic Circuits: For $s = 2 \cdot \mathrm{poly}\,(n^c, n^{c_{\mathsf{error}}})$, the size of $\mathsf{NonDetIsBad}_{X,n,t,k,\varepsilon_{\mathsf{nm}}/4}$ according to Theorem 8.6, let

$$G\colon \{0,1\}^{a \log s} \to \{0,1\}^s$$

be the $(1/s)$-PRG (computable in time $\mathrm{poly}(s)$) for nondeterministic circuits of size s guaranteed by Theorem 3.9, for some constant a. We will in particular note that $s \ge d'$ (since d' is the input length to $\mathsf{NonDetIsBad}_{X,n,t,k,\varepsilon_{\mathsf{nm}}/4}$) and $1/s \le \varepsilon_{\mathsf{nm}}/2$. Also note G is seed-extending and therefore injective.

[9] That is, once we construct such a function, the final result will consider yet another IsBad that is meant to determine whether a seed to the resulting construction is good or not.

Definition of the Extractor: We define:

$$\mathsf{nmExtComp}(x, y) = \mathsf{nmExt}(x, G(y))$$

where the output of $G(y)$ is truncated to d' bits to be of proper length. One can verify that the algorithm runs in time $\mathrm{poly}(n)$.

Correctness: We first verify that $d = a \log s \le a' \log n/\varepsilon_{\mathsf{nm}}$ for some constant a' (depending on c and t), and the runtime is $\mathrm{poly}(s) + (n/\varepsilon_{\mathsf{nm}})^{c'_{\mathsf{nm}}} \le (n/\varepsilon_{\mathsf{nm}})^{a''}$ for some constant a'' (also depending on c and t). Therefore we can take $c_{\mathsf{nm}} = \max(a', a'')$.

It remains to verify that the conclusion of Theorem 3.21 holds. Fix any X samplable by a size n^c circuit of min-entropy k (we set $C(t) = 16c_{\mathsf{Li}}$). Fix also any $\varepsilon'_{\mathsf{nm}} \ge \varepsilon_{\mathsf{nm}}$, and let $h_{y'} = \mathsf{nmExt}(\cdot, y')$. Define:

$$\mathsf{BAD}^* = \{r \in \{0,1\}^d \mid \exists \text{ distinct } r_1, \dots, r_t \in \{0,1\}^d \backslash \{y\}, \Delta\left((h_{G(r)}(X); U_1) | \{h_{G(r_i)}(X)\}_{i=1}^t\right) > \sqrt{\varepsilon'_{\mathsf{nm}}}\}.$$

We wish to show that $|\mathsf{BAD}^*| \le \sqrt{\varepsilon'_{\mathsf{nm}}} D$. First, since G is injective, we know that if $r \in \mathsf{BAD}^*$ then $G(r) \in \mathsf{BAD}_{X,\varepsilon_{\mathsf{nm}}}$. Furthermore, by Theorem 8.6, we know that $\mathsf{BAD}_{X,\varepsilon_{\mathsf{nm}}} \subset \mathsf{BAD}'_X$ and $|\mathsf{BAD}'_X| \le \frac{\sqrt{\varepsilon_{\mathsf{nm}}}}{2} D'$. Since G $(\varepsilon_{\mathsf{nm}}/2)$-fools BAD'_X, we have that:

$$\Pr_r[r \in \mathsf{BAD}^*] \le \Pr_r[G(r) \in \mathsf{BAD}_{X,\varepsilon_{\mathsf{nm}}}] \le \Pr_r[G(r) \in \mathsf{BAD}'_X] \le \frac{|\mathsf{BAD}'_X|}{D'} + \frac{\varepsilon_{\mathsf{nm}}}{2} \le \frac{\sqrt{\varepsilon_{\mathsf{nm}}}}{2} + \frac{\varepsilon_{\mathsf{nm}}}{2} \le \sqrt{\varepsilon'_{\mathsf{nm}}}.$$

■

8.2 A Final Remark: PRGs on PRGs on PRGs

We conclude by observing that there are four PRGs involved in Theorem 8.1. First, there is the PRG used as a subroutine to NonDetIsBad, which runs in some time $\mathrm{poly}(n)$. In the construction of Theorem 8.5, we require a PRG that fools NonDetIsBad, and thus this PRG, and the resulting nmExtComp, should run in an even larger $\mathrm{poly}(n)$. Next, nmExtComp is used as a subroutine to IsBad from Theorem 5.1, where it in turn must also be fooled by yet another PRG running in yet another larger polynomial. Finally, IsBad itself must be "fooled" by our final SRPEG.

It is entirely possible that some steps are ultimately redundant. Specifically it might be possible to flatten the construction by combining the PRG in nmExtComp and the PRG in SRPEG. We do not attempt to do so because we wish to highlight the potential of the technique given improvements to explicit constructions of non-malleable malleable extractors.

9 Discussion and Open Problems

9.1 Evaluating the New Hardness Assumption

In this paper we use a new hardness assumption (namely the assumption that E is hard for large exponential time with exponential advice, defined in Definition 3.5). To the best of our knowledge, hardness assumptions of this flavor where first introduced by Chen and Tell [CT21]. Our assumption has the same structure, but is quantitatively weaker.

Discussing the previous assumptions. Before evaluating this assumption, for the purpose of comparison, let us point out that for the more standard assumption introduced by Impagliazzo and Wigderson [IW97] (that E is hard for exponential size circuits) one can consider two possible justifications:

A *candidate based* justification is that it is commonly believed that there exists an $\alpha > 0$ such that $\mathsf{SAT} \notin \mathsf{SIZE}(2^{\alpha \cdot n})$, and so (if we add an "infinitely often") we can instantiate the hardness assumption with the problem SAT which is obviously in $\mathsf{E} = \mathsf{DTIME}(2^{O(n)})$.

A *completeness based* justification is that the language

$$L = \left\{(M, x, 1^t) : \text{Turing machine } M \text{ accepts } x \text{ in time } 2^t\right\}$$

is in E and is obviously complete for E under linear time reductions. This implies that if the hardness assumption holds (and some problem in E is sufficiently hard) then it can be instantiated with L (even in case SAT is not sufficiently hard).

We can continue this rationale to the stronger assumption that E is hard for exponential time nondeterministic circuits. Here, SAT will obviously not do, but a potential candidate is $\mathsf{co} - \mathsf{SAT}$. Furthermore, the completeness justification holds.

One can easily extend this discussion to even stronger hardness assumptions against Σ_i-circuits, where the candidate will be $\Sigma_{i+1} - \mathsf{SAT}$.

Discussing the new assumption. We now turn our attention to the new assumption that E is hard for large exponential time with exponential advice. For this assumption, we can define the language:

$$L_c = \left\{(M, x) : \text{Turing machine } M \text{ accepts } x \text{ in time } 2^{c \cdot |x|}\right\}.$$

Continuing the previous discussion, note that as before, for every constant c, L_c is in E, and furthermore, it is natural to conjecture that nonuniform procedures with time $2^{C_{\mathsf{hard}}}$ that use $2^{\alpha \cdot n}$ bits of advice (for a sufficiently small $\alpha > 0$) cannot accept L_c, if c is chosen to be a constant that is sufficiently larger than C_{hard}. This means that in the assumption that E is hard for large exponential time with exponential advice, when given a constant C_{hard} we can hope to take c to be a constant that is significantly larger than C_{hard} and instantiate the assumption with L_c.

Furthermore, by the completeness argument explained earlier, if the new hardness assumption holds, then the candidate choice above can be used.

9.2 Open Problems

The first natural open problem is to improve the seed length of currently known non-malleable extractors to $O(\log n/\varepsilon_{\mathsf{nm}})$. The recent line of work that constructs such extractors for $k = \Omega(d)$ [CGL16, Coh16b, Coh16a, CL16, Li17] do not quite have the right dependence between d and $\varepsilon_{\mathsf{nm}}$. In fact, as Section 8 suggests, even constructing nice non-malleable extractors for samplable distributions under, say the asssumption that E is hard for exponential time and exponential advice, rather than E is hard for exponential size nondeterministic circuits would offer an improvement, by removing the nondeterministic assumption.

Reducing the min-entropy in Theorem 1.3. As explained in Section 1, the reason that we only get k slightly larger than $n/2$ in Theorem 1.3 is that current best constructions of non-malleable extractors are not "sufficiently nice" for our purposes. We remark once again, that the current best explicit constructions by Li [Li17], *do* achieve a very low k of $O(\log n \cdot \log\log n)$, and if the seed length d could be improved from $O(\log n + \log \frac{1}{\varepsilon_{\mathsf{nm}}} \cdot \log\log \frac{1}{\varepsilon_{\mathsf{nm}}})$ to $O(\log n + \log \frac{1}{\varepsilon_{\mathsf{nm}}})$, then Theorem 7.1 immediately implies an extractor for samplable distributions for the same k, under the sole assumption that E is hard for large exponential size with exponential advice.

An alternative approach is to try and extend the argument in Theorem 1.4 which achieves a low k, but assumes the additional assumption that E is hard for exponential size nondeterministic circuits. As explained in Section 8 this assumption is used to "reduce the seed length" of the non-malleable extractor of [Li17] (in the case that the input distribution is samplable). More concretely, as explained Section 8, a "nice non-malleable extractors for samplable distributions" (in the precise meaning of Definition 8.2) is sufficient for our purposes. It is natural to ask whether such seeded non-malleable extractors for samplable distributions can be obtained with a weaker hardness assumption than the one used in Theorem 8.1.

Improving the output length and error of the extractor. A drawback of Theorem 1.2 (and its more formal restatement in Theorem 7.1) is that the obtained extractor for samplable distributions only outputs a single bit, and has error $\varepsilon > 0$ that is constant.

As explained Section 2, the output length and error that we get, *match* the output length and error that is obtained by Ben-Aroya, Doron and Ta-Shma [BDT19] in their construction of two-source extractors. It should be noted that other constructions of two-source extractors are able to obtain error ε that is polynomially small in n, with a larger output length m (specifically, the breakthrough construction of Chattopadhyay and Zuckerman [CZ16] and Li [Li17]). A natural approach is to try and imitate the structure of these constructions in our reduction. See discussion in Section 2.5

We remark that in order to get large output length it may be sufficient to achieve low error, with logarithmic output length. This is because Shaltiel [Sha08] showed that extractors for samplable distributions with short output length $m = O(\log n)$ and error $\varepsilon < 2^{-m}$, can be improved to give output length $m \approx k$ (if the initial extractor has some additional properties) and indeed this transformation is used in some previous extractor constructions mentioned in Table 1.

10 Acknowledgments

We thank Dean Doron for some clarifications about [BDT19].

References

[AIKS16] Sergei Artemenko, Russell Impagliazzo, Valentine Kabanets, and Ronen Shaltiel. Pseudorandomness when the odds are against you. In *31st Conference on Computational Complexity, CCC*, pages 9:1–9:35, 2016.

[BDSGM23] Marshall Ball, Dana Dachman-Soled, Eli Goldin, and Saachi Mutreja. Extracting randomness from samplable distributions, revisited. In *2023 IEEE 64th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1505–1514. IEEE, 2023.

[BDT19] Avraham Ben-Aroya, Dean Doron, and Amnon Ta-Shma. An efficient reduction from two-source to nonmalleable extractors: achieving near-logarithmic min-entropy. *SIAM Journal on Computing*, (0):STOC17–31, 2019.

[BSS25] Marshall Ball, Ronen Shaltiel, and Jad Silbak. Extractors for samplable distributions with low min-entropy. In *Proceedings of the 57th Annual ACM Symposium on Theory of Computing*, pages 596–603, 2025.

[CGL16] Eshan Chattopadhyay, Vipul Goyal, and Xin Li. Non-malleable extractors and codes, with their many tampered extensions. In *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing*, pages 285–298, 2016.

[Cha20] Eshan Chattopadhyay. Guest column: A recipe for constructing two-source extractors. *ACM SIGACT News*, 51(2):38–57, 2020.

[CL16] Eshan Chattopadhyay and Xin Li. Explicit non-malleable extractors, multi-source extractors, and almost optimal privacy amplification protocols. In *Foundations of Computer Science (FOCS), 2016 IEEE 57th Annual Symposium on*, pages 158–167. IEEE, 2016.

[Coh16a] Gil Cohen. Making the most of advice: New correlation breakers and their applications. In *Foundations of Computer Science (FOCS), 2016 IEEE 57th Annual Symposium on*, pages 188–196. IEEE, 2016.

[Coh16b] Gil Cohen. Non-malleable extractors-new tools and improved constructions. In *LIPIcs-Leibniz International Proceedings in Informatics*, volume 50. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2016.

[CRS14] Gil Cohen, Ran Raz, and Gil Segev. Nonmalleable extractors with short seeds and applications to privacy amplification. *SIAM Journal on Computing*, 43(2):450–476, 2014.

[CT21] Lijie Chen and Roei Tell. Simple and fast derandomization from very hard functions: eliminating randomness at almost no cost. In *Proceedings of the 53rd Annual Symposium on Theory of Computing (STOC)*, pages 283–291. ACM, 2021.

[CT23] Lijie Chen and Roei Tell. When arthur has neither random coins nor time to spare: Superfast derandomization of proof systems. In *Proceedings of the 55th Annual ACM Symposium on Theory of Computing*, pages 60–69, 2023.

[CZ16] Eshan Chattopadhyay and David Zuckerman. Explicit two-source extractors and resilient functions. In *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing*, pages 670–683. ACM, 2016.

[DPW14] Yevgeniy Dodis, Krzysztof Pietrzak, and Daniel Wichs. Key derivation without entropy waste. In *Advances in Cryptology–EUROCRYPT 2014*, pages 93–110. Springer, 2014.

[DW09] Yevgeniy Dodis and Daniel Wichs. Non-malleable extractors and symmetric key cryptography from weak secrets. In *Proceedings of the forty-first annual ACM symposium on Theory of computing*, pages 601–610. ACM, 2009.

[ISW99] Russell Impagliazzo, Ronen Shaltiel, and Avi Wigderson. Near-optimal conversion of hardness into pseudo-randomness. In *40th Annual Symposium on Foundations of Computer Science (Cat. No. 99CB37039)*, pages 181–190. IEEE, 1999.

[IW97] Russell Impagliazzo and Avi Wigderson. P = BPP if E requires exponential circuits: Derandomizing the XOR lemma. In *Proceedings of the 29th Annual ACM Symposium on Theory of Computing (STOC 1997)*, pages 220–229. ACM, 1997.

[KvM02] Adam R. Klivans and Dieter van Melkebeek. Graph nonisomorphism has subexponential size proofs unless the polynomial-time hierarchy collapses. *SIAM Journal on Computing*, 31(5):1501–1526, 2002.

[Li17] Xin Li. Improved non-malleable extractors, non-malleable codes and independent source extractors. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, pages 1144–1156, 2017.

[LP23] Yanyi Liu and Rafael Pass. Leakage-resilient hardness vs randomness. In *Proceedings of the conference on Proceedings of the 38th Computational Complexity Conference*, pages 1–20, 2023.

[MV05] Peter B. Miltersen and N. Vinodchandran Variyam. Derandomizing Arthur–Merlin games using hitting sets. *Computational Complexity*, 14(3):256–279, 2005.

[RT00] Jaikumar Radhakrishnan and Amnon Ta-Shma. Bounds for dispersers, extractors, and depth-two superconcentrators. *SIAM Journal on Discrete Mathematics*, 13(1):2–24, 2000.

[Sha08] Ronen Shaltiel. How to get more mileage from randomness extractors. *Random Structures & Algorithms*, 33(2):157–186, 2008.

[Sha24] Ronen Shaltiel. Multiplicative extractors for samplable distributions. In *Electronic Colloquium on Computational Complexity (ECCC), TR24-168*, 2024.

[Sha25] Ronen Shaltiel. Extractors for samplable distribution with polynomially small min-entropy. In *Electronic Colloquium on Computational Complexity (ECCC), TR25-054*, 2025.

[STV01] Madhu Sudan, Luca Trevisan, and Salil Vadhan. Pseudorandom generators without the XOR lemma. *Journal of Computer and System Sciences*, 62(2):236–266, 2001.

[SU05] Ronen Shaltiel and Christopher Umans. Simple extractors for all min-entropies and a new pseudorandom generator. *Journal of the ACM*, 52(2):172–216, 2005.

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

[Tre01] Luca Trevisan. Extractors and pseudorandom generators. *Journal of the ACM*, 48(4):860–879, 2001.

[TV00] Luca Trevisan and Salil Vadhan. Extracting randomness from samplable distributions. In *Proceedings of the 41st Annual IEEE Symposium on Foundations of Computer Science (FOCS 2000)*, pages 32–42. IEEE, 2000.

[Uma03] Christopher Umans. Pseudo-random generators for all hardnesses. *Journal of Computer and System Sciences*, 67(2):419–440, 2003.

[Vio14] Emanuele Viola. Extractors for circuit sources. *SIAM Journal on Computing*, 43(2):655–672, 2014.

[Zuc07] David Zuckerman. Linear degree extractors and the inapproximability of Max Clique and Chromatic Number. *Theory of Computing*, 3:103–128, 2007.