

# 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  $E = \text{DTIME}(2^{O(n)})$  is hard for exponential size circuits with oracle access to  $\Sigma_6^P$ . 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  $\Sigma_i^P$  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  $C_{\text{hard}} \geq 1$ , there exists a constant  $C_{\text{easy}}$  and a problem in  $\text{DTIME}(2^{C_{\text{easy}} \cdot n})$  that is not in  $\text{DTIME}(2^{C_{\text{hard}} \cdot n} / 2^{\alpha \cdot n})$ . 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.

We show that under this assumption, there is an explicit extractor for samplable distributions for min-entropy  $k = O(\log n)$  (matching the min-entropy threshold achieved by the probabilistic method). More formally, we get that for every constant  $c \geq 1$ , and every constant  $\varepsilon > 0$ , there exists a constant  $c'$ , and explicit extractor  $\text{Ext} : \{0, 1\}^n \rightarrow \{0, 1\}$  with error  $\varepsilon$ , for distributions samplable by circuits of size  $n^c$ , that have min-entropy  $k \geq c' \cdot \log n$ .

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, ISF grant 1006/23, and also supported by the European Union (ERC, ECCC, 101076663). Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Research Council. Neither the European Union nor the granting authority can be held responsible for them.

†University of Haifa. ronen@cs.haifa.ac.il. Supported by ISF grant 1006/23, and also supported by the European Union (ERC, NFITSC, 101097959). Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Research Council. Neither the European Union nor the granting authority can be held responsible for them.

# Contents

|          |                                                                         |           |
|----------|-------------------------------------------------------------------------|-----------|
| <b>1</b> | <b>Introduction</b>                                                     | <b>1</b>  |
| 1.1      | Our Results . . . . .                                                   | 2         |
| 1.2      | Perspective: A Computational Analog of Two Source Extractors . . . . .  | 5         |
| <b>2</b> | <b>Technical Overview</b>                                               | <b>6</b>  |
| 2.1      | The Solution at a High Level . . . . .                                  | 6         |
| 2.2      | Trouble with Parameters . . . . .                                       | 9         |
| 2.3      | The Solution of [BDT19], and Its Computational Analogue . . . . .       | 10        |
| <b>3</b> | <b>Preliminaries</b>                                                    | <b>11</b> |
| 3.1      | Random Variables . . . . .                                              | 11        |
| 3.2      | Definition of Circuits of Various Types . . . . .                       | 12        |
| 3.3      | Hardness Assumptions . . . . .                                          | 13        |
| 3.4      | Pseudorandom Generators . . . . .                                       | 14        |
| 3.5      | Dispersers . . . . .                                                    | 15        |
| 3.6      | Majority as an Extractor for Non-oblivious Bit Fixing Sources . . . . . | 15        |
| 3.7      | Non-Malleable Extractors . . . . .                                      | 16        |
| <b>4</b> | <b>PRGs For Procedures With Large Time and Small Advice</b>             | <b>17</b> |
| <b>5</b> | <b>An Algorithm For Recognizing Bad Seeds</b>                           | <b>19</b> |
| <b>6</b> | <b>A “Somewhere Pseudoentropy Generator” for IsBad</b>                  | <b>22</b> |
| <b>7</b> | <b>The Extractor Construction</b>                                       | <b>26</b> |
| <b>8</b> | <b>Discussion and Open Problems</b>                                     | <b>30</b> |
| 8.1      | Evaluating the New Hardness Assumption . . . . .                        | 30        |
| 8.2      | Open Problems . . . . .                                                 | 32        |
| <b>9</b> | <b>Acknowledgments</b>                                                  | <b>33</b> |

# 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  $X : \{0, 1\}^s \rightarrow \{0, 1\}^n$  (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)  $(k, \varepsilon)$ -extractor for distributions samplable by circuits of size  $s$ , is a function:

$$\text{Ext} : \{0, 1\}^n \rightarrow \{0, 1\}^m$$

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  $\text{Ext}(X)$  is  $\varepsilon$ -close to  $U_m$ .<sup>1</sup>

[TV00] show that if  $\text{Ext}$  is an extractor for samplable distributions computable in time  $s$ , then  $\text{Ext}$  cannot be computed by circuits 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  $\text{poly}(n)$  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  $E = \text{DTIME}(2^{O(n)})$ . We say that “ $E$  is hard for exponential size circuits of type  $X$ ” if there exist constants  $0 < \alpha < C_{\text{easy}}$ , and a language  $L$  in  $\text{DTIME}(2^{C_{\text{easy}} \cdot n})$ , 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 the 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  $\text{BPP} = \text{P}$ . 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  $\Sigma_i$ -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  $\Sigma_i^P$  (namely, the  $i$ 'th level of the polynomial time hierarchy). See Definition 3.2 for a formal definition of  $\Sigma_i$ -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  $\Sigma_6$ -circuits. Under this

---

<sup>1</sup>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  $(k, \varepsilon)$ -extractor for distributions samplable by size  $n^c$  circuits, where  $k = (1 - \gamma) \cdot n$ ,  $\varepsilon = n^{-c}$  and the extractor runs in time  $\text{poly}(n^c)$ . 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  $\Sigma_6$ -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  $\Sigma_5$ -circuits. Shaltiel [Sha25] further improved the min-entropy parameter to  $k = n^{1/i}$  under the assumption that E is hard for exponential size  $\Sigma_{i+3}$ -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] and Li [Li23].

Using this new approach we obtain extractors with  $k = O_c(\log n)$  which matches the min-entropy threshold obtained by the probabilistic method [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 to previous ones. Chen and Tell [CT21] considered a 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  $2^{C_{\text{hard}} \cdot n}$  (for some constant  $C_{\text{hard}} > 1$ ) 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  $C_{\text{hard}} > 1$ , there exists a constant  $C_{\text{easy}} > C_{\text{hard}}$ , and a language  $L$  in  $\text{DTIME}(2^{C_{\text{easy}} \cdot n})$ , 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  $2^{C_{\text{hard}} \cdot n}$  and use  $2^{\alpha \cdot n}$  bits of advice. See also the 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 [\[CT21\]](#).<sup>2</sup> A recent work by [\[KS25\]](#) also uses a similar assumption that separates time from advice.<sup>3</sup>

**Comparing Hardness Assumptions.** The assumption that E is hard for large exponential time with exponential advice is obviously stronger than the assumption used by [\[IW97\]](#) that E is hard for exponential size (deterministic) circuits. The new assumption seems incomparable to the previous assumptions used in the context of extractors for samplable distributions [\[TV00, BDSGM23, Sha24, BSS25, Sha25\]](#). More specifically, 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  $\Sigma_i$ -circuits)” or vice versa. The new assumption is stronger as it allows the “adversary” a polynomial increase in time  $2^{C_{\text{hard}} \cdot n} = \text{poly}(2^{\alpha \cdot n})$ , and weaker in the sense that it does not give the “adversary” nondeterminism. A discussion on potential instantiations of this assumption appears in [Section 8.1](#).

### 1.1.2 Extractors With Min-Entropy That Match The Probabilistic Method

Trevisan and Vadhan [\[TV00\]](#) used the probabilistic method to show that for every constants  $c \geq 1$  and  $\varepsilon > 0$ , there exists a constant  $c' \geq c$ , such that for  $k = c' \cdot \log n$ , there exists a  $(k, \varepsilon)$ -extractor  $\text{Ext} : \{0, 1\}^n \rightarrow \{0, 1\}$  for distributions samplable by circuits of size  $n^c$ .

In this paper we give a matching *explicit* extractor, under the new hardness assumption that E is hard for large exponential time and exponential advice.

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

<sup>3</sup>The hardness assumption considered in [\[KS25\]](#) considered vastly larger bounds on time, such as  $2^{2^n}$ , and hardness against algorithms (with small advice) that also have access to an NP oracle.

**Theorem 1.2** (Extractors with min-entropy that matches the probabilistic method). *If  $E$  is hard for large exponential time with exponential advice, then for every constants  $c \geq 1$ , and  $\varepsilon > 0$ , there exists a constant  $c'$  such that for every sufficiently large  $n$ , there is a  $(k = c' \cdot \log n, \varepsilon)$ -extractor  $\text{Ext} : \{0, 1\}^n \rightarrow \{0, 1\}$  for distributions samplable by circuits of size  $n^c$ . Furthermore,  $\text{Ext}$  runs in time  $\text{poly}(n^c)$ .*

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 $\Sigma_6$ -circuits:<br>$\exists 0 < \alpha < 1 < C_{\text{easy}} \text{ s.t.}$<br>$\text{DTIME}(2^{C_{\text{easy}} \cdot n}) \not\subseteq \text{SIZE}_{\Sigma_6}^P(2^{\alpha \cdot n})$                                                                                                | $m \approx k,$<br>$\varepsilon = n^{-c}$      |
| [BDSGM23]                | $(1 - \gamma)n$ | $E$ is hard for exponential size nondeterministic circuits:<br>$\exists 0 < \alpha < 1 < C_{\text{easy}} \text{ s.t.}$<br>$\text{DTIME}(2^{C_{\text{easy}} \cdot n}) \not\subseteq \text{NSIZE}(2^{\alpha \cdot n})$                                                                                                       | $m \approx k,$<br>$\varepsilon = n^{-c}$      |
| [BSS25]                  | $n^{1-\gamma}$  | $E$ is hard for exponential size $\Sigma_5$ -circuits:<br>$\exists 0 < \alpha < 1 < C_{\text{easy}} \text{ s.t.}$<br>$\text{DTIME}(2^{C_{\text{easy}} \cdot n}) \not\subseteq \text{SIZE}_{\Sigma_5}^P(2^{\alpha \cdot n})$                                                                                                | $m \approx k,$<br>$\varepsilon = n^{-c}$      |
| [Sha25]                  | $n^{1/i}$       | $E$ is hard for exponential size $\Sigma_{i+3}$ -circuits:<br>$\exists 0 < \alpha < 1 < C_{\text{easy}} \text{ s.t.}$<br>$\text{DTIME}(2^{C_{\text{easy}} \cdot n}) \not\subseteq \text{SIZE}_{\Sigma_{i+3}}^P(2^{\alpha \cdot n})$                                                                                        | $m \approx k,$<br>$\varepsilon = n^{-c}$      |
| This work<br>Theorem 1.2 | $O_c(\log n)$   | $E$ is hard for large exponential time with exponential advice:<br>$\exists 0 < \alpha < 1, \text{ s.t. } \forall C_{\text{hard}} \geq 1, \exists C_{\text{easy}} > C_{\text{hard}} \text{ s.t.}$<br>$\text{DTIME}(2^{C_{\text{easy}} \cdot n}) \not\subseteq \text{SIZE}(2^{C_{\text{hard}} \cdot n})/2^{\alpha \cdot n}$ | $m = 1,$<br>any constant<br>$\varepsilon > 0$ |

**Comparison to Previous Work on Extractors For Samplable Distributions.** A comparison of our results and the previous results appears in Table 1.

**Advantages.** In 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 while the new assumption is incomparable to the assumptions used previously to obtain extractors for samplable distributions, this is the first construction of extractors for samplable distributions that *does not involve a lower bound for nondeterministic circuits*.

We also stress again that Theorem 1.2 gives extractors for samplable distributions for min-entropy  $k = O(\log n)$ , matching the min-entropy achieved by the probabilistic argument.

**Drawbacks.** Two drawbacks of our new result are 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  $m$  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 they are not inherent to the new approach, and in [Section 8.2](#) we elaborate on potential directions to overcome these issues.

### 1.1.3 A Reduction from Extractors for Samplable Distributions to Non-malleable Extractors

Our approach builds on the recent improvements in explicit constructions of two-source extractors [CZ16, BDT19, Li23]. Following these constructions we rely 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 3.15](#), 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  $d = O(\log \frac{n}{\varepsilon_{\text{nm}}})$  (at least when  $\varepsilon_{\text{nm}}$  is not too small). A precise formal definition of a “nice construction of non-malleable extractors” appears in [Definition 3.16](#). Our main contribution is the following theorem (stated more formally in [Theorem 7.1](#)).

**Theorem 1.3 (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  $(k, \varepsilon)$ -extractor  $\text{Ext} : \{0, 1\}^n \rightarrow \{0, 1\}$  for distributions samplable by circuits of size  $n^c$ . Furthermore,  $\text{Ext}$  runs in time  $\text{poly}(n^c)$ .*

A long line of works [CRS14, CGL16, Coh16b, Coh16a, CL16, Li17, Li23] give explicit constructions of non-malleable extractors, and our [Theorem 1.2](#) follows by plugging in a nice explicit construction of non-malleable extractors by Li [Li23], which works for min-entropy  $k = O(\log n)$ .

## 1.2 Perspective: A Computational Analog of Two Source Extractors

Thanks to Trevisan [Tre01], it is by now well known and widely celebrated wisdom in pseudo-randomness 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  $G^f : \{0, 1\}^d \rightarrow \{0, 1\}^m$  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  $\text{Ext}(x, y) = G^x(y)$  is an extractor.

Conversely (see e.g. [SU05]), given a construction of a “reconstructive extractor”  $\text{Ext} : \{0, 1\}^n \times \{0, 1\}^d \rightarrow \{0, 1\}^m$ , replacing the (true) randomness from the source  $X$  with the truth table of an appropriately hard function  $f$  yields a PRG  $G^f(y) = \text{Ext}(f, y)$ .

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. 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 (that is constructed under the new hardness assumption). A more detailed high level description appears in [Section 2](#).

## 2 Technical Overview

In this section we provide a high level explanation of the proofs of [Theorem 1.2](#), and [Theorem 1.3](#). 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 1.3](#) and infer [Theorem 1.2](#).

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 3.15](#) 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  $(t, k, \varepsilon_{\text{nm}})$ -non-malleable extractor

$$\text{nmExt}: \{0, 1\}^n \times \{0, 1\}^d \rightarrow \{0, 1\},$$

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

$$\Delta((h_y(X), h_{y_1}(X), \dots, h_{y_t}(X)); (U_1, h_{y_1}(X), \dots, h_{y_t}(X))) \leq \sqrt{\varepsilon_{\text{nm}}}. \quad (1)$$

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  $\sqrt{\varepsilon_{\text{nm}}} \cdot 2^d$  bad seeds. A precise formal statement

appears in [Theorem 3.19](#). Roughly speaking, with a bit more work, this implies that the distribution  $Z = \{\text{nmExt}(X, y)\}_{y \in \{0,1\}^d}$  on  $2^d$  bits has the property that the restriction of  $Z$  onto the good coordinates (corresponding to good seeds) is  $(\gamma = \sqrt{\varepsilon_{\text{nm}}})$ -almost  $t$ -wise independent; that is, further restricting to any subset of  $t$  good coordinates yields a distribution that is  $\gamma$ -close to  $U_t$ . There are at most  $q = \sqrt{\varepsilon_{\text{nm}}}$  bad coordinates. Such distributions are called  $(q, t, \gamma)$ -non-oblivious bit fixing sources (see [Definition 3.12](#) 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  $\gamma$ -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  $\gamma = \sqrt{\varepsilon_{\text{nm}}}$  and  $R = 2^d$ . However,  $d$  and  $\varepsilon_{\text{nm}}$  are *related*. Indeed, by the known lower bound of [\[RT00\]](#),  $d \geq \log \frac{n}{\varepsilon_{\text{nm}}^2} - O(1)$ . 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  $\varepsilon_{\text{nm}}$ . Specifically given a sample  $x_2 \sim X_2$ , the subset  $S$  they construct is the image of  $\text{Ext}(x_2, \cdot)$  for an appropriate  $\text{Ext}: \{0,1\}^n \times \{0,1\}^r \rightarrow \{0,1\}^d$ . Since  $\gamma$  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  $\text{nmExt}: \{0,1\}^n \rightarrow \{0,1\}^d \rightarrow \{0,1\}$  is a non-malleable extractor for constant  $t$ , with  $d = \log(n/\varepsilon_{\text{nm}})$ , then for any  $\varepsilon_{\text{nm}} = 1/\text{poly}(n)$  there is an algorithm  $\text{IsBad}_X$  that recognizes the set of bad seeds to  $\text{nmExt}$  (with respect to  $X$ ) in time  $\text{poly}(n)$ .*

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

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

The key properties that allow  $\text{IsBad}_X$  to run in polynomial time are threefold: the first is that  $t$  is constant, the second is that  $d = \log(n/\varepsilon_{\text{nm}})$  (and  $\varepsilon_{\text{nm}} = 1/\text{poly}(n)$ ), 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  $\varepsilon_{\text{nm}}$ . See Section 5 for details.

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

**A Crucial Step: Decoupling  $R$  from  $\varepsilon_{\text{nm}}$ .** 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  $\varepsilon_{\text{nm}}$  (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  $\varepsilon_{\text{nm}}$  will be inherently related. Indeed,  $\text{IsBad}_X$  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  $\text{SIZE}(\text{IsBad}_X)$ . 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  $\text{IsBad}_X$ .

We observe that  $\text{IsBad}_X$  is an algorithm that runs in time  $\text{poly}(n) \gg n^c$  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  $\text{poly}(n^c)$  of a function that is hard for algorithms using  $\approx n^c$  bits of advice and time a much larger  $\text{poly}(n)$  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.<sup>4</sup> The precise details are given in Section 4.

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

As a final note, notice that  $\text{IsBad}_X$  requires approximating the difference between the two distributions in Equation (1) to within  $\sqrt{\varepsilon_{\text{nm}}}$ , which in general takes  $1/\varepsilon_{\text{nm}}$  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  $\text{IsBad}_X$  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.

---

<sup>4</sup>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  $O(c \log n)$ .

**The Bird's Eye View So Far.** Overall, the current suggestion for our construction of the extractor is as follows. We can take a non-malleable extractor  $\text{nmExt} : \{0, 1\}^n \times \{0, 1\}^d \rightarrow \{0, 1\}$  with  $d = O(\log n / \varepsilon_{\text{nm}})$ . As long as  $\varepsilon_{\text{nm}} = 1/\text{poly}(n)$ , the set of bad seeds to  $\text{nmExt}$  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  $\text{nmExt}$  using a PRG  $G : \{0, 1\}^r \rightarrow \{0, 1\}^d$  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  $\varepsilon_{\text{nm}}$  sufficiently smaller than  $R^{-t} \approx n^{-ct}$ . Since  $t$  is constant, this means  $\varepsilon_{\text{nm}}$  only needs to be polynomially small. Then overall, the error of the final extractor construction will be  $\gamma R^t = \sqrt{\varepsilon_{\text{nm}}} R^t \ll 1$ .

## 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  $\text{nmExt}$ . This in turn produces a non-oblivious bit fixing source for constant  $t$ . The extractor for non-oblivious bit fixing sources from [CZ16] requires  $t = \text{poly} \log n$ . 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  $(q, t, \sqrt{\varepsilon_{\text{nm}}})$ -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  $q \leq \sqrt{R}$ . 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,  $\varepsilon_{\text{nm}}$  (which governs  $d$ ), and the error of the PRG  $\varepsilon_{\text{PRG}}$ . We did not explicitly discuss the error of the PRG in the previous discussion, as it wasn't a necessary detail until now. However, the seed length of the PRG is  $r = O(\log \frac{n}{\varepsilon_{\text{PRG}}})$  while the seed length of  $\text{nmExt}$  is  $d = O(\log \frac{n}{\varepsilon_{\text{nm}}})$ . What we learned from the previous discussion, is that  $r$  and  $d$  can be unrelated, and in particular, for any choice of  $\varepsilon_{\text{PRG}}$  we are free to choose  $\varepsilon_{\text{nm}}$  as sufficiently smaller than  $\varepsilon_{\text{PRG}}$  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  $\varepsilon_{\text{nm}}$  and  $\varepsilon_{\text{PRG}}$  make sense. First we observe that since the fraction of bad seeds overall is  $\sqrt{\varepsilon_{\text{nm}}}$ , the fraction of bad seeds in  $S$  is no more than  $\sqrt{\varepsilon_{\text{nm}}} + \varepsilon_{\text{PRG}} \approx \varepsilon_{\text{PRG}}$  for sufficiently small choice of  $\varepsilon_{\text{nm}}$ . Therefore, the total number of bad seeds in  $S$  is at most  $\varepsilon_{\text{PRG}} \cdot R$ . The question now, is whether  $\varepsilon_{\text{PRG}} \cdot R \leq \sqrt{R}$ ? Unfortunately, the answer is no. For a PRG with error  $\varepsilon_{\text{PRG}}$ , the seed length should be  $r \geq \log n + 2 \cdot \log 1/\varepsilon_{\text{PRG}}$ . A calculation confirms our fears that no choice of  $\varepsilon_{\text{PRG}}$  gives us what we want. We do note however, that if  $r = O(\log n) + (1 + \beta) \log 1/\varepsilon_{\text{PRG}}$ , for some  $\beta < 1/2$ , then there is a sufficiently small choice of  $\varepsilon_{\text{PRG}} = 1/\text{poly}(n)$  such that  $\varepsilon_{\text{PRG}} \cdot R < \sqrt{R}$  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  $\text{Ext}: \{0, 1\}^n \times \{0, 1\}^r \rightarrow \{0, 1\}^d$  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 $\varepsilon_{\text{PRG}}$ : 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  $\sqrt{\varepsilon_{\text{nm}}}$ . 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  $c_{\text{nm}} \cdot \log(n/\varepsilon_{\text{nm}})$  for some constant  $c_{\text{nm}}$ , then the total number of bad seeds is roughly  $2^{(1-1/c_{\text{nm}})d} = D^{1-\frac{1}{c_{\text{nm}}}}$  (for sufficiently small choice of  $\varepsilon_{\text{nm}}$ ). 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  $\text{lsBad}_X$ ) cannot be more than the probability any  $.99d$ -source does so (with an additional error  $\varepsilon_{\text{PEG}}$ ). In particular, this means that for the set of bad seeds, the fraction falling within  $S$  is:

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

Once again, by taking  $\varepsilon_{\text{nm}}$  sufficiently small, which in turn makes  $D$  sufficiently large, this fraction is  $\approx \varepsilon_{\text{PEG}}$ . 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  $1 \cdot \log 1/\varepsilon_{\text{PEG}}$  (as [DPW14, BDT19] point out), it is plausible to construct pseudoentropy generators with the same dependence on  $\varepsilon_{\text{PEG}}$ .<sup>5</sup>

**A Table of Seeds, and the XOR Trick** Unfortunately, since we are unaware of any suitable constructions of pseudoentropy generators with  $d \approx 1 \cdot \log 1/\varepsilon_{\text{PEG}}$ , 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:  $\text{SEPEG}: \{0, 1\}^{r'} \times [A] \rightarrow \{0, 1\}^d$ .

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

<sup>5</sup>A probabilistic argument, similar to that in [DPW14] can show that pseudoentropy generators (for some appropriate notion) with seed length  $\log \log |\mathcal{F}| + \log 1/\varepsilon_{\text{PEG}}$  exists for any family of tests  $\mathcal{F}$ . Additionally, [AIKS16] construct the similar notion of a hitting set generator with seed length  $\log 1/\varepsilon$ .

nmExt (for constant  $A$ ), with the property that all but an  $\varepsilon_{\text{PEG}}$  fraction of rows contains at least one good seed.

If nmExt is a  $(t' = t \cdot A, k, \varepsilon_{\text{nm}})$ -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  $(q = \varepsilon_{\text{PEG}} \cdot R', t, \sqrt{\varepsilon_{\text{nm}}})$  non-oblivious bit fixing source of length  $R'$ . As discussed before, this is exactly what we want, provided that  $r' = 1.1 \cdot \log \frac{1}{\varepsilon_{\text{PEG}}}$ , or  $R' = \left(\frac{1}{\varepsilon_{\text{PEG}}}\right)^{1.1}$ .

**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  $G: \{0, 1\}^r \rightarrow \{0, 1\}^d$  against  $\text{IsBad}_X: \{0, 1\}^d \rightarrow \{0, 1\}$  with the wrong dependence on  $\varepsilon_{\text{PRG}}$ . 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  $r = c_{\text{PRG}}(\log n + \log 1/\varepsilon_{\text{PRG}})$ . In other words, for parameter  $\varepsilon_{\text{PEG}}$  if we insist on only paying, say,  $r = .1 \cdot \log 1/\varepsilon_{\text{PEG}}$  seed, we obtain a PRG with error  $\varepsilon_{\text{PRG}} = \varepsilon_{\text{PEG}}^{1/(10 \cdot c_{\text{PRG}})}$ .

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  $R' = \left(\frac{1}{\varepsilon_{\text{PEG}}}\right)^{1.1}$  left nodes, left degree a constant  $A$ , and  $R = \left(\frac{1}{\varepsilon_{\text{PEG}}}\right)^{.1}$  right nodes. The property of this graph is that any subset of nodes on the left with fraction at least  $\varepsilon_{\text{PEG}}$  has a neighborhood on the right of fraction at least  $\varepsilon_{\text{PRG}} = \varepsilon_{\text{PEG}}^{1/(10 \cdot c_{\text{PRG}})}$ .

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  $\varepsilon_{\text{PEG}}$  fraction of rows in this table have all entries as bad seeds. This implies the neighborhood of these rows is a  $\varepsilon_{\text{PRG}} = \varepsilon_{\text{PEG}}^{1/(10 \cdot c_{\text{PRG}})}$  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  $\varepsilon_{\text{PRG}}$ -PRG to  $\text{IsBad}_X$  implies that there cannot be more than  $\sqrt{\varepsilon_{\text{nm}}} + \varepsilon_{\text{PRG}} \approx \varepsilon_{\text{PRG}}$  fraction of bad seeds  $y$ .

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.

The discussion above shows how to prove Theorem 1.3, and Theorem 1.2 now follows by plugging in the nice construction of non-malleable extractors of [Li23] (which is stated precisely in Theorem 3.18).

## 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  $\Omega$ , we use  $\Delta(A; B)$  to denote the statistical distance. That is,  $\Delta(A; B) = \frac{1}{2} \sum_{x \in \Omega} |A(x) - B(x)| = \max_{S \subseteq \Omega} |A(S) - B(S)|$ . 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  $A_1, \dots, A_t$  we use  $\{A_i\}_{i=1}^t$  to denote the joint distribution  $(A_1, \dots, A_t)$ .

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  $X_1, \dots, X_t, Y_1, \dots, Y_k$  be boolean random variables. Further suppose that for any  $i \in [t]$ ,*

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

*Then*

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

## 3.2 Definition of Circuits of Various Types

### 3.2.1 Nondeterministic Circuits And $\Sigma_i$ -Circuits

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

**Definition 3.2** (Nondeterministic circuits, oracle circuits and  $\Sigma_i$ -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  $\Sigma_i$ -circuit is an  $A$ -circuit where  $A$  is the canonical  $\Sigma_i^P$ -complete language. The size of all circuits is the total number of wires and gates.<sup>6</sup>*

### 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

<sup>6</sup>An alternative approach to define these circuit classes is using the Karp-Lipton notation for Turing machines with advice. For  $s \geq n$ , a size  $s^{\Theta(1)}$  deterministic circuit is equivalent to  $\text{DTIME}(s^{\Theta(1)})/s^{\Theta(1)}$ , a size  $s^{\Theta(1)}$  nondeterministic circuit is equivalent to  $\text{NTIME}(s^{\Theta(1)})/s^{\Theta(1)}$ , a size  $s^{\Theta(1)}$  NP-circuit is equivalent to  $\text{DTIME}^{\text{NP}}(s^{\Theta(1)})/s^{\Theta(1)}$ , and a size  $s^{\Theta(1)}$   $\Sigma_i$ -circuit is equivalent to  $\text{DTIME}^{\Sigma_i^P}(s^{\Theta(1)})/s^{\Theta(1)}$ .

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  $f : \{0, 1\}^n \rightarrow \{0, 1\}$  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  $\langle M \rangle$  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  $0 < \alpha < C_{\text{easy}}$ , and a language  $L$  in  $\text{DTIME}(2^{C_{\text{easy}} \cdot n})$ , 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  $\text{BPP} = \text{P}$  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  $\text{AM} = \text{NP}$ .

In their seminal paper on extractors for samplable distributions, Trevisan and Vadhan introduced a version of the assumption for  $\Sigma_i$ -circuits, and their extractor is constructed under the assumption that E is hard for exponential size  $\Sigma_6^{\text{P}}$ -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.

**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  $C_{\text{hard}} > 1$ , there exists a constant  $C_{\text{easy}} > C_{\text{hard}}$ , and a language  $L$  in  $\text{DTIME}(2^{C_{\text{easy}} \cdot n})$ , 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  $2^{C_{\text{hard}} \cdot n}$  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  $2^{C_{\text{hard}} \cdot n}$  for an arbitrary  $C_{\text{hard}} \geq 1$ , and we require that for every constant  $C_{\text{hard}} \geq 1$  there is a constant  $C_{\text{easy}} > C_{\text{hard}}$  such that the problem  $L$  can be computed in time  $2^{C_{\text{easy}} \cdot n}$ .

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 8.1](#).

### 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  $\varepsilon$ -pseudorandom for a class  $\mathcal{C}$  of functions, if for every  $C : \{0, 1\}^n \rightarrow \{0, 1\} \in \mathcal{C}$ ,*

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

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

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

*that is a seed-extending  $\frac{1}{s}$ -PRG for circuits of size  $s$ . Furthermore,  $G$  is computable in time  $\text{poly}(s)$ . (Here, the polynomial depends on the constants  $\alpha, C_{\text{easy}}$  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 sufficiently 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  $\tilde{p}$  such that*

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

*Furthermore,  $A'$  runs in time  $\text{poly}(s, \frac{1}{\varepsilon})$  where the polynomial depends on the constants 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 3.9.** A function  $\Gamma: [N] \times [D] \rightarrow [M]$  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  $\varepsilon$  is the parameter. However, for the setting that we are interested in the “error parameter”  $\varepsilon$  is close to one, rather than close to zero, and the parametrization in Definition 3.9 is more natural. This setting (where  $\varepsilon$  approaches one) was considered by Zuckerman [Zuc07]. The following theorem is a restatement of Zuckerman's theorem [Zuc07] using Definition 3.9.

**Theorem 3.10** ([Zuc07], Theorem 1.9). For every constant  $\delta > 0$  there exists a constant  $c_\delta$  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  $\Gamma: [N] \times [D] \rightarrow [M]$ , with  $D = c_\delta \cdot \frac{n}{\log \frac{1}{s}}$ . Furthermore,  $\Gamma$  is computable in time  $\text{poly}(n)$ .

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 3.10 is that setting  $s = M^{-\gamma}$  for a constant  $0 < \gamma < 1$ , one obtains that  $D$  is a constant that depends on  $\delta, \gamma$ .

### 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 3.11.** A distribution  $X$  is called  $(t, \gamma)$ -wise independent if the restriction of  $X$  to every  $t$ -coordinates is  $\gamma$ -close to  $U_t$ .

**Definition 3.12.** A source  $X$  over  $\{0, 1\}^n$  is a  $(q, t, \gamma)$  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  $(t, \gamma)$ -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 3.13** ([Vio14], See [BDT19], Lemma 2.17.). For  $\text{Maj}: \{0, 1\}^R \rightarrow \{0, 1\}$ , there exists a constant  $c_{\text{Maj}}$  such that for every  $\alpha > 0$  and a  $(q = R^{\frac{1}{2}-\alpha}, t, \gamma)$  non-oblivious bit-fixing source  $X$  on  $R$  bits,

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

For a fixed constant error  $\varepsilon$  of the final extractor, we will instantiate  $t$ , and  $\alpha$ , and choose  $R$  (and  $\gamma$ ) appropriately, so that the RHS is at most  $\varepsilon$ .

**Corollary 3.14.** For any  $\varepsilon > 0$ , let  $t = \frac{9c_{\text{Maj}}^2}{\varepsilon^2}$ , and let  $\alpha = 1/8$ . Suppose further that  $R \geq \frac{10^4 \cdot c_{\text{Maj}}^8}{\varepsilon^8}$ , and  $\gamma \leq \frac{\varepsilon}{3 \cdot c_{\text{Maj}} \cdot R^t}$ . Then for a  $(q = R^{\frac{1}{4}}, t, \gamma)$  non-oblivious bit-fixing source  $X$  on  $R$  bits,

$$\left| \Pr[\text{Maj}(X_1, \dots, X_R) = 1] - \frac{1}{2} \right| \leq \varepsilon.$$

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

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

### 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  $\varepsilon_{\text{nm}} = n^{-c_{\text{error}}}$  for a sufficiently large constant  $c_{\text{error}}$  that will be chosen in the proof.

In our reduction, the running time of the final extractor will be  $\text{poly}(n, 2^d, 2^{2^t}, \text{time}(\text{nmExt}))$ . 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  $\text{poly}(n)$ . We refer to such constructions of non-malleable extractors as “nice”, and the precise formulation is stated below.

**Definition 3.16** (Nice construction of non-malleable extractors). Let  $k(t, n, \varepsilon_{\text{nm}})$  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  $c_{\text{nm}}$  such that for every constant  $c_{\text{error}} \geq 1$ , and for every sufficiently large  $n$ , setting  $\varepsilon_{\text{nm}}(n) = n^{-c_{\text{error}}}$ , there is a function  $d(n) \leq c_{\text{nm}} \cdot \log \frac{n}{\varepsilon_{\text{nm}}(n)}$  and a  $(t, k(t, n, \varepsilon_{\text{nm}}), \varepsilon_{\text{nm}}(n))$ -non-malleable extractor  $\text{nmExt}: \{0, 1\}^n \times \{0, 1\}^{d(n)} \rightarrow \{0, 1\}$ , such that  $\text{nmExt}$  can be computed in time  $(\frac{n}{\varepsilon_{\text{nm}}(n)})^{c_{\text{nm}}}$ .

**Remark 3.17** (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 3.16](#)) should satisfy two properties:

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

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  $\varepsilon_{\text{nm}}(n) = n^{-\omega(1)}$ ), 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.

Nevertheless, the recent construction of non-malleable extractors by Li [[Li23](#)] (which is the best known in terms of min-entropy threshold, and achieves  $k = O(\log n)$ ) is nice.

**Theorem 3.18** (See Theorem 7.4, [Li23]). *For some constant  $C(t)$ , there is a nice construction of non-malleable extractors for min-entropy  $k = C(t) \cdot \left( \log \log n + \log \frac{1}{\varepsilon_{\text{nm}}} \right)$ .*

### 3.7.3 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 3.19 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 3.19** (See [CZ16], Lemma 3.4). *Let  $\text{nmExt}: \{0, 1\}^n \times \{0, 1\}^d \rightarrow \{0, 1\}$  be a  $(t, k, \varepsilon_{\text{nm}})$ -non-malleable extractor. For each  $y \in \{0, 1\}^d$ , define the function  $h_y: \{0, 1\}^n \rightarrow \{0, 1\}$  as  $h_y(x) = \text{nmExt}(x, y)$ . For any  $(n, k)$ -source  $X$ , and any  $\varepsilon'_{\text{nm}} \geq \varepsilon_{\text{nm}}$ , define:*

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

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

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

We note that Theorem 3.19 is only stated in [CZ16] for  $\varepsilon'_{\text{nm}} = \varepsilon_{\text{nm}}$ , however, the statement above is also true as any  $(t, k, \varepsilon_{\text{nm}})$ -non-malleable extractor is also a  $(t, k, \varepsilon'_{\text{nm}})$ -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*

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

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  $\text{poly}(s)$ . (Here, the polynomial depends on the constant  $\alpha$  in the hardness assumption, the constant  $c$ , and is also affected by the dependence between  $C_{\text{hard}}$  and  $C_{\text{easy}}$  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  $f : \{0, 1\}^\ell \rightarrow \{0, 1\}$  into a function  $G_f : \{0, 1\}^{d \cdot \ell} \rightarrow \{0, 1\}^{2^{\delta \cdot \ell}}$ . 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  $f : \{0, 1\}^\ell \rightarrow \{0, 1\}$ , the function  $G_f$  can be computed in time  $2^{O(\ell)}$  with oracle access to  $f$ .
- For any functions  $f : \{0, 1\}^\ell \rightarrow \{0, 1\}$ , and  $D : \{0, 1\}^{2^{\delta \cdot \ell}} \rightarrow \{0, 1\}$ , 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  $\ell$ . 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  $\alpha$ .

By the first item,  $G_f$  can be computed in time  $2^{O(\ell)} \cdot 2^{C_{\text{easy}} \cdot \ell} = \text{poly}(s)$  (where the polynomial depends on  $\alpha, C_{\text{easy}}$ ). If  $G_f$  is not a  $\frac{1}{s}$ -PRG for some circuit  $D : \{0, 1\}^s \rightarrow \{0, 1\}$  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  $2^{\frac{\alpha}{2} \cdot \ell} \cdot \text{size}(D) \leq 2^{\frac{\alpha}{2} \cdot \ell} \cdot 2^{\frac{\alpha}{2} \cdot \ell} = 2^{\alpha \ell}$  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  $\alpha$ , and so  $s^c = 2^{c \cdot \delta \cdot \ell}$  might be larger than  $2^\ell$ . We choose  $C_{\text{hard}} = 2 \cdot c \cdot \delta \cdot \ell$ , and the hardness assumption provides a constant  $C_{\text{easy}} > C_{\text{hard}}$  and a hard problem  $L$ . We take  $f$  to be the characteristic function of the hard problem on inputs of length  $\ell$ . 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  $\alpha$  (but not on  $c$ ).

By the first item,  $G_f$  can be computed in time  $2^{O(\ell)} \cdot 2^{C_{\text{easy}} \cdot \ell} = \text{poly}(s)$  (where the polynomial depends on  $\alpha, C_{\text{easy}}$  and recall that  $C_{\text{easy}}$  depends on  $C_{\text{hard}}$  that depends on  $c$ ). If  $G_f$  is not a  $\frac{1}{s}$ -PRG

for some nonuniform procedure  $D : \{0, 1\}^s \rightarrow \{0, 1\}$  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  $2^{\frac{\alpha}{2} \cdot \ell} \cdot \text{time}(D) \leq 2^{\frac{\alpha}{2} \cdot \ell} \cdot 2^{c \cdot \delta \ell} \leq 2^{2 \cdot c \cdot \delta \ell} = 2^{C_{\text{easy}} \cdot \ell}$  and number of advice bits that is used by  $C^D$  is  $2^{\frac{\alpha}{2} \cdot \ell} + \text{advice}(D) \leq 2^{\frac{\alpha}{2} \cdot \ell} + 2^{\frac{\alpha}{2} \cdot \ell} \leq 2^{\alpha \ell}$ . 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  $G : \{0, 1\}^d \rightarrow \{0, 1\}^n$  against a class  $\mathcal{C}$ , if  $n \geq d + 1$ , then the function  $f : \{0, 1\}^{d+1} \rightarrow \{0, 1\}$ , 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  $\text{poly}(2^d, n)$ , 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  $c_{\text{PRG}}$ , such that for every constant  $c_{\text{runtime}} > 1$ , every function  $\varepsilon_{\text{PRG}} = \varepsilon_{\text{PRG}}(n)$ , and every sufficiently large  $n$ , there is a*

$$G : \{0, 1\}^{c_{\text{PRG}} \cdot \log(n/\varepsilon_{\text{PRG}})} \rightarrow \{0, 1\}^{n^c}$$

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

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

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

## 5 An Algorithm For Recognizing Bad Seeds

A key component in recent constructions of two-source extractors [\[CZ16, BDT19\]](#) is applying a  $(t, k, \varepsilon_{\text{nm}})$ -non-malleable extractor  $\text{nmExt} : \{0, 1\}^n \times \{0, 1\}^d \rightarrow \{0, 1\}$  to transform a distribution  $X$  over  $\{0, 1\}^n$  with  $H_\infty(X) \geq k$ , into  $D = 2^d$  random variables  $(\text{nmExt}(X, y))_{y \in \{0, 1\}^d}$ . As stated formally in [Theorem 3.19](#), [\[CZ16\]](#) showed that there is a small set  $\text{BAD}_{X, \varepsilon_{\text{nm}}} \subseteq \{0, 1\}^d$  of “bad seeds”, such that the distribution  $Z = (\text{nmExt}(X, y))_{y \in \{0, 1\}^d \setminus \text{BAD}_{X, \varepsilon_{\text{nm}}}}$  has the property that every subset of  $t$  bits from  $Z$  is (close to) uniform.<sup>7</sup> In this section we show that if:

- $X$  is samplable by size  $n^c$  circuits,

<sup>7</sup>The property mentioned above in the text is oversimplified, and following [\[BDT19\]](#) we need the stronger property (stated formally in [Theorem 3.19](#)) 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.

- E is hard for large exponential time with exponential advice, and
- The non-malleable extractor nmExt comes from a nice construction (See [Definition 3.16](#)),

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

Jumping ahead, we mention that we will later use the theorem choosing a constant  $c_{\text{error}}$  that is much larger than  $c$ , so that this procedure will run in time  $n^{c_{\text{runtime}}}$  where  $c_{\text{runtime}}$  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  $c_{\text{runtime}}$ .

**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  $k = k(t, n, \varepsilon_{\text{nm}})$ .

Let  $t, c, c_{\text{error}} \geq 1$  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  $\text{nmExt} : \{0, 1\}^n \times \{0, 1\}^d \rightarrow \{0, 1\}$  to be a  $(k, t, \varepsilon_{\text{nm}}/16)$ -non-malleable extractor, where  $\varepsilon_{\text{nm}} = n^{-c_{\text{error}}}$ , and note that  $D = 2^d \leq n^{c_{\text{nm}} \cdot c_{\text{error}}}$ , let  $\text{IsBad}_{X,n,t,k,\varepsilon_{\text{nm}}}(y)$  be the procedure in [Figure 1](#) and let  $\text{BAD}'_X = \{y \mid \text{IsBad}_{X,n,t,k,\varepsilon_{\text{nm}}}(y) = 1\}$ .

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

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

Furthermore, for any  $y_1, \dots, y_t \notin \text{BAD}'_X$ :

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

2.  $\text{BAD}'_X \subset \text{BAD}_{X,\varepsilon_{\text{nm}}/16}$  and in particular,  $|\text{BAD}'_X| \leq \sqrt{\varepsilon_{\text{nm}}}D$ .
3.  $\text{IsBad}_{X,n,t,k,\varepsilon_{\text{nm}}}$  runs in time  $\text{poly}(n^c, n^{c_{\text{error}}} = \frac{1}{\varepsilon_{\text{nm}}})$  (where the polynomial depends on  $t, c_{\text{nm}}$  and the constants in the hardness assumption) and uses  $n^c$  bits of nonuniformity.

**Proof:**

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

$$\Delta((h_y(X); U_1) \mid \{h_{y_i}(X)\}_{i=1}^t) \leq \frac{\sqrt{\varepsilon_{\text{nm}}}}{2} + \frac{\sqrt{\varepsilon_{\text{nm}}}}{4} \leq \sqrt{\varepsilon_{\text{nm}}}.$$

Therefore  $y \in \{0, 1\}^d \setminus \text{BAD}_{X,\varepsilon_{\text{nm}}}$ . Applying [Theorem 3.19](#) yields the result.

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

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

Since nmExt is a  $(t, k, \varepsilon_{\text{nm}}/16)$ -non-malleable extractor applying [Theorem 3.19](#) yields the result.

Figure 1: A procedure that recognizes bad seeds

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

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

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

**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  $y_1, \dots, y_t \in \{0, 1\}^d$ ,  $y_i \neq y_j$ :
  - (a) For every subset  $S \subset \{0, 1\}^t$ :
    - i. Let  $\mathbf{1}_S: \{0, 1\}^t \rightarrow \{0, 1\}$  denote the algorithm determining membership in  $S$ .
    - ii. Define  $A_1: \{0, 1\}^{n^c} \rightarrow \{0, 1\}$  as:

$$A_1(z) = \mathbf{1}_S(h_y(X(z)), h_{y_1}(X(z)), \dots, h_{y_t}(X(z))).$$

Also define  $A_2: \{0, 1\}^{n^c} \times \{0, 1\} \rightarrow \{0, 1\}$  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  $s = t \cdot n^c \cdot \left(\frac{n}{\varepsilon_{\text{nm}}}\right)^{c_{\text{nm}}} = \text{poly}(n^c, n^{c_{\text{error}}})$ , for a polynomial that depends on  $t, c_{\text{nm}}$ . We will use the algorithm from [Theorem 3.8](#), compute  $\tilde{p}_1$  and  $\tilde{p}_2$ , that are  $\sqrt{\varepsilon_{\text{nm}}}/8$  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  $\text{poly}(s, \frac{1}{\varepsilon_{\text{nm}}}) = \text{poly}(n^c, n^{c_{\text{error}}})$  for a polynomial that depends on  $t, c_{\text{nm}}$ , and the constants in the hardness assumption.
- iv. If  $|\tilde{p}_1 - \tilde{p}_2| > \sqrt{\varepsilon_{\text{nm}}}/2$ , then break and output  $b = 1$ .

2. Output  $b = 0$ .

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  $O(\sqrt{\varepsilon_{\text{nm}}})$  on a circuit of size  $s = \text{poly}(n^c, n^{c_{\text{error}}})$ . By [Theorem 3.8](#) each such iteration can be done in time  $\text{poly}(n^c, n^{c_{\text{error}}})$  (for a polynomial that depends on  $t, c_{\text{nm}}$  and the constants in the hardness assumption).

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

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

(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 lsBad

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 lsBad 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 lsBad is a nonuniform procedure that runs in time  $n^{c_{runtime}}$  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 lsBad. 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  $\varepsilon$  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  $G: \{0, 1\}^r \rightarrow \{0, 1\}^d$  is a  $(\mu, \varepsilon_{\text{PEG}})$ -evasive pseudoentropy generator (evasive PEG) against a class  $\mathcal{C}$  if for every  $C: \{0, 1\}^d \rightarrow \{0, 1\} \in \mathcal{C}$ , with  $\Pr[C(U_c) = 1] \leq \mu$ , we have that:

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

Note that an  $\varepsilon$ -PRG is by definition, a  $(\mu, \mu + \varepsilon)$ -evasive PEG for every  $0 \leq \mu \leq 1$ . However, an evasive-PEG only needs to satisfy the latter property for some specific (typically very small)  $\mu$ . This weaker property allows an evasive-PEG to have seed length where the dependence on  $\varepsilon_{\text{PEG}}$  is  $(1 + \beta) \cdot \log \frac{1}{\varepsilon_{\text{PEG}}}$  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  $G: \{0, 1\}^r \times [A] \rightarrow \{0, 1\}^d$  is a  $(\mu, \varepsilon_{\text{PEG}})$ -somewhere-evasive PEG against a class  $\mathcal{C}$  if for every  $C: \{0, 1\}^d \rightarrow \{0, 1\} \in \mathcal{C}$ , such that  $\Pr[C(U_c) = 1] \leq \mu$ , we have that:

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

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  $c, c_{\text{runtime}} > 0$  and  $0 < \beta, \eta < 1$  there is a constant  $A = A(\beta, c) \geq 1$  (that does not depend on  $\eta$  and  $c_{\text{runtime}}$ ) such that for every sufficiently large  $n$ , and for every  $\varepsilon_{\text{PEG}} > 0$  and every integer  $d$  that satisfy:*

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

$$\frac{1}{\text{poly}(n)} \leq \varepsilon_{\text{PEG}} \leq \left(\frac{1}{n}\right)^{1/\nu}.$$

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

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

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

*there is an injective function*

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

*That is a  $(\mu = D^{-\eta}, \varepsilon_{\text{PEG}})$ -somewhere-evasive PEG for nonuniform procedures  $B: \{0, 1\}^d \rightarrow \{0, 1\}$  running in time  $n^{c_{\text{runtime}}}$  with  $n^c$  bits of advice. Furthermore, SEPEG is computable in time  $\text{poly}(n^c)$ .<sup>8</sup>*

*In particular, for  $\beta = 1/3$  this implies that the total number of seeds  $y'$  for which  $B(\text{SEPEG}(y', a)) = 1$  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  $n^{c_{\text{runtime}}}$  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  $\varepsilon_{\text{PEG}}$  is sufficiently small inverse polynomial in  $n$ , and  $D = 2^d$  is a sufficiently large polynomial in  $n$ ) we obtain a seed length  $r = (1 + \beta) \cdot \log \frac{1}{\varepsilon_{\text{PEG}}}$  (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 [\[BDT19\]](#).

<sup>8</sup>One may ask whether we can be more explicit about the exponent of the runtime, perhaps as a function of  $c_{\text{runtime}}$ . 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  $C_{\text{easy}}$  depends on  $C_{\text{hard}}$ .

**Construction:**

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

**Parameters:**

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

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

**Ingredients:**

**A PRG:** We will apply [Theorem 4.3](#) with the constant  $c$ , to obtain a constant  $c_{\text{PRG}} = c_{\text{PRG}}(c)$ . We choose  $\bar{c}_{\text{runtime}}$  so that  $n^{\bar{c}_{\text{runtime}}} = \frac{n^{c_{\text{runtime}}}}{\varepsilon_{\text{PEG}}^4}$ , and note that we can indeed choose such a constant  $\bar{c}_{\text{runtime}}$  as we are requiring that  $\varepsilon_{\text{PEG}} \geq \frac{1}{\text{poly}(n)}$ . We choose  $\bar{\varepsilon}_{\text{PRG}} = \varepsilon_{\text{PEG}}^\nu$ . Recall that  $\nu = \frac{\beta}{4c_{\text{PRG}}}$  and note that  $\bar{\varepsilon}_{\text{PRG}} = \varepsilon_{\text{PEG}}^\nu \leq \frac{1}{n}$ , where the last inequality follows because we are requiring that  $\varepsilon_{\text{PEG}} \leq \left(\frac{1}{n}\right)^{1/\nu}$ . Using [Theorem 4.3](#), we obtain a function

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

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

We have that:

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

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

**A Disperser:** We will apply [Theorem 3.10](#) choosing  $\delta = \frac{\beta}{1+\beta}$  to obtain a function

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

such that for  $R' = 2^{r'}$  and  $\bar{R} = 2^{\bar{r}}$ ,  $\Gamma$  is a  $\left(K = (R')^{\frac{\beta}{1+\beta}}, K' = 2\bar{\varepsilon}_{\text{PRG}}\bar{R}\right)$ -disperser.

Note that [Theorem 3.10](#) indeed gives  $K = (R')^{\frac{\beta}{1+\beta}}$ ,  $\bar{r} = \frac{\delta r'}{2} = \frac{\beta}{2} \cdot \log(1/\varepsilon_{\text{PEG}})$  and

choosing  $s = 2 \cdot \bar{\varepsilon}_{\text{PRG}}$ , we obtain that  $K' = 2\bar{\varepsilon}_{\text{PRG}}\bar{R}$  with

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

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

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

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

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

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

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:

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

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

Suppose that  $B: \{0, 1\}^d \rightarrow \{0, 1\}$  is computable by an  $n^{c_{\text{runtime}}}$  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  $(D^{-\eta}, \varepsilon_{\text{PEG}})$ -PEG for  $B$ , then  $|B| \leq D^{1-\eta}$ , and yet:

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

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:

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

We have that

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

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

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

which gives that

$$(D')^{\eta/2} \geq \varepsilon_{\text{PEG}}^{2 \cdot \eta} \cdot D^{\eta/2} \geq \varepsilon_{\text{PEG}}^2 \cdot D^{\eta/2} \geq \varepsilon_{\text{PEG}}^2 \cdot \left( \frac{1}{\varepsilon_{\text{PEG}}} \right)^{\frac{12 \cdot \eta}{\eta}} \geq \frac{1}{\varepsilon_{\text{PEG}}^4}. \quad (3)$$

We use (3) to conclude that:

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

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

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

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

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

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

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

## 7 The Extractor Construction

In this section we prove [Theorem 1.3](#) 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 1.3](#) in a formal way, relying on the precise definition on nice non-malleable extractors give in [Definition 3.16](#).

**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  $k(t, n, \varepsilon_{\text{nm}})$ .

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

$$\text{Ext}: \{0, 1\}^n \rightarrow \{0, 1\}$$

for distributions samplable by circuits of size  $n^c$ , for  $k = k(t, n, n^{-c_{\text{error}}})$ . Furthermore, the extractor runs in time  $\text{poly}(n^c)$ .<sup>9</sup>

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 3.18](#) yields [Theorem 1.2](#).

<sup>9</sup>As before, we cannot specify the exact dependence of the exponent of the polynomial on  $\alpha, c, \varepsilon$ , because we make no assumption on the relationship between  $C_{\text{hard}}$  and  $C_{\text{easy}}$  in [Definition 3.5](#). However, if  $C_{\text{easy}} = O(C_{\text{hard}})$ , then the runtime is  $n^{\text{poly}(1/\alpha, c, 1/\varepsilon)}$ .

**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  $\text{Ext} : \{0, 1\}^n \rightarrow \{0, 1\}$ , works as follows:

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

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

of Theorem 6.3 to fool the procedure  $\text{IsBad}$  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:

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

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

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  $y = \text{SEPEG}(y', z)$  is a bad seed for  $\text{nmExt}$  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  $\text{SEPEG}$  is  $r' = (1 + \beta) \cdot \log \frac{1}{\epsilon_{\text{PEG}}}$  and we can choose, say  $\beta = 1/3$ .
- For every  $y' \in \{0, 1\}^{r'}$  which is “good”, one can show that  $\oplus \text{NM}(X, y')$  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  $y = \text{SEPEG}(y', z^*)$  is a good seed, meaning that  $\text{nmExt}(X, y)$  is close to uniform (even conditioned on the output of  $\text{nmExt}$  on seeds of the form  $\text{SEPEG}(y', z)$  for  $z \neq z^*$ ).
- The  $R'$  input bits that we feed to  $\text{Maj}$ , are an oblivious bit-fixing source of the type described in Theorem 3.13. 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  $\text{Maj}$  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 3.16](#) (for constants  $t$  and  $c_{\text{error}}$  to be chosen shortly), and the somewhere-evasive PEG in [Theorem 6.3](#) (for constants  $c_{\text{runtime}}, \beta, \eta$  to be chosen shortly)
- Min-entropy parameter  $k = k(t, n, \varepsilon_{\text{nm}})$ , determined by the construction of the nice non-malleable extractor.

**Goal:** Construct  $\text{Ext}: \{0, 1\}^n \rightarrow \{0, 1\}$  such that for every  $X: \{0, 1\}^{n^c} \rightarrow \{0, 1\}^n$  computable by a circuit of size  $n^c$  such that  $H_\infty(X) \geq k, |\Pr[\text{Ext}(X) = 1] - \frac{1}{2}| \leq \varepsilon$ .

**Ingredients:**

**Some Initial Parameter Choices:** There are several parameters to consider when combining [Definition 3.16](#) and [Theorem 6.3](#), with many dependencies. We first need to define some initial parameters before defining the objects that depend on them:

- Define  $t' = \frac{9c_{\text{Maj}}^2}{\varepsilon^2}$ . For  $c_{\text{Maj}}$  according to [Theorem 3.13](#).
- 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 3.16](#), for our choice of constant  $t$ , we have a constant  $c_{\text{nm}} = c_{\text{nm}}(t)$  such that for every polynomially small error,  $\varepsilon_{\text{nm}} = n^{-c_{\text{error}}}$ , there is a  $(t, k, \varepsilon_{\text{nm}}/16)$ -non-malleable extractor:

$$\text{nmExt}: \{0, 1\}^n \times \{0, 1\}^d \rightarrow \{0, 1\}$$

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

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

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

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

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

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

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

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

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

**The Final Parameters,  $\varepsilon_{\text{nm}}$ ,  $c_{\text{error}}$ , and  $c_{\text{runtime}}$ :** . We finally set  $\varepsilon_{\text{nm}}$  sufficiently small to satisfy the constraints of both [Corollary 3.14](#) and [Theorem 6.3](#). Namely:

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

In particular this implies that  $\varepsilon_{\text{nm}} \leq \frac{\varepsilon^2}{(15tc_{\text{Maj}}R'^t)^2}$ , so that for  $\gamma = 5t\sqrt{\varepsilon_{\text{nm}}}$ , we have  $\gamma \leq \frac{\varepsilon}{3 \cdot c_{\text{Maj}} \cdot R'^t}$  as required by [Corollary 3.14](#). Additionally, since any extractor must have  $d \geq \log 1/\varepsilon_{\text{nm}}$ , we also have:

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

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

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

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

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

**Correctness:** Let  $k = k(t, n, \varepsilon_{\text{nm}})$  for the parameters  $n, t$  and  $\varepsilon_{\text{nm}}$  chosen above. Note that by our choices,  $t$  is a constant that depends on  $c, \varepsilon$ , and  $\varepsilon_{\text{nm}} = n^{-c_{\text{error}}}$  where  $c_{\text{error}}$  is a constant that depends on  $c, \varepsilon$ , 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:

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

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

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

Thus  $\Pr[\text{IsBad}_{X,n,t,k,\varepsilon_{\text{nm}}}(U_d) = 1] \leq D^{-\eta}$ . Therefore indeed by [Theorem 6.3](#), for all but  $\varepsilon_{\text{PEG}} R' \leq (\sqrt{R'})^{1/4}$  seeds  $y'$ , there is a  $z$  such that  $\text{SEPEG}(y', z) \notin \text{BAD}'_X$ . Call a row  $y'$  good if there is such a good  $z$  for it. Since  $\text{nmExt}$  is a  $(t = t' \cdot A, k, \varepsilon_{\text{nm}}/16)$ -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  $y'_1, \dots, y'_{t-1}$  and any  $z \in [A]$ . Formally:

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

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

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

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

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

This proves that  $\oplus \text{NM}(x, 1), \dots, \oplus \text{NM}(x, R')$  is a  $((R')^{1/4}, t, \gamma)$  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 3.14 yields the result. ■

We end this section by remarking that we obtain Theorem 1.2 by applying the non-malleable extractor in Theorem 3.18, which handles min-entropy  $k = C(t) \cdot \left( \log \log n + \log \frac{1}{\varepsilon_{\text{nm}}} \right)$ , with the choices of  $t = t(\varepsilon, c)$ , and  $\varepsilon_{\text{nm}} = n^{-c_{\text{error}}}$  for  $c_{\text{error}} = c_{\text{error}}(\varepsilon, c)$  that are specified above in the proof of Theorem 7.1.

## 8 Discussion and Open Problems

### 8.1 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 (namely, against algorithms with super exponential time and sub-exponential advice) were first introduced by Chen and Tell [CT21]. Our assumption has the same structure, but is quantitatively weaker. Another version of a hardness assumption with this flavor was recently suggested by Korten and Santhanam [KS25], and this version involves larger time bounds, as well as hardness against algorithms (with small advice) that also have access to an NP oracle.

In this section we discuss some justifications for the new assumption. For the purpose of comparison, let us discuss justifications for the more standard assumption that  $E$  is hard for exponential size circuits, defined in Definition 3.4.

### 8.1.1 Justification for: E is Hard for Exponential Size Circuits

**An assumption about the power of nonuniformity.** The assumption that E is hard for exponential size circuits, reflects a belief that adding  $2^{\alpha \cdot n}$  bits of nonuniformity to a deterministic algorithm running in time  $2^{\alpha \cdot n}$  where  $0 < \alpha < 1$ , does not give it a “polynomial speedup” and in particular, does not enable the algorithm to solve every problem in  $\text{DTIME}(2^{C_{\text{easy}} \cdot n})$  for every constant  $C_{\text{easy}} > 1$ .

**A completeness argument.** One can use an easy “completeness argument” to show that if the assumption holds, then one can identify a language  $L$  for which the assumption holds.

More specifically, suppose that the assumption holds for some *unknown* language  $L$  with respect to some specific *known* constants  $0 < \alpha < C_{\text{easy}}$ . It is easy to see that in this case, the assumption also holds (for some  $\alpha' = \alpha - o(1)$  and  $C'_{\text{easy}} = C_{\text{easy}} + o(1)$ ) with respect to the language:

$$L'_{C_{\text{easy}}} = \{(M, x) : \text{Turing machine } M \text{ accepts } x \text{ in } 2^{C_{\text{easy}} \cdot |x|} \text{ steps}\}.$$

This means that in this scenario we can instantiate the assumption *without* having a specific  $L$  in mind. The argument for this assertion is the following trivial argument:

- The language  $L'_{C_{\text{easy}}}$  can obviously be decided in time  $2^{C'_{\text{easy}} \cdot n}$ , for  $C'_{\text{easy}} = C_{\text{easy}} + o(1)$ .
- If  $L'_{C_{\text{easy}}}$  is not sufficiently hard, and there exists a circuit  $C'$  of size  $2^{\alpha' \cdot n}$ , that solves  $L'_{C_{\text{easy}}}$  on infinitely many input lengths, then, taking  $C(x) = C'(M, x)$  (where  $M$  is the machine that decides  $L$ ) gives a circuit  $C$  of size  $2^{(\alpha' + o(1)) \cdot n}$  that solves  $L$  on infinitely many input lengths, and this is a contradiction if  $\alpha' + o(1) \leq \alpha$ .

One may argue that a disappointing feature of this argument is that we still need to know (or at least guess) specific values for the constants  $\alpha$  and  $C_{\text{easy}}$ . We want to make the point that actual constructions that rely on the assumption that E is hard for exponential size circuits (e.g. [Theorem 3.7](#)) need to be supplied with specific values for the constants  $\alpha$  and  $C_{\text{easy}}$ .

Our point is that in order to utilize the hardness assumption, one must already have fixed choices for these two constants in mind, and the “completeness argument” above says that if one already has such choices in mind, then one does not need to evaluate specific candidates for  $L$ , and can simply use  $L'_{C_{\text{easy}}}$ .

While in the discussion above we considered hardness against *deterministic* circuits, the same justifications apply if we modify the that E is hard for exponential size (deterministic) circuits, to hold against nondeterministic circuits, or  $\Sigma_i$ -circuits. In fact, in the next section we observe that these two justifications also apply for the new hardness assumption that E is hard for large exponential time with exponential advice.

### 8.1.2 Justification for: E is Hard for Large Exponential Time with Exponential Advice

We now explain that the two justifications considered for the previous assumption also hold for the new assumption that E is hard for large exponential time with exponential advice.

**An assumption about the power of nonuniformity.** Similarly to the previous section, the assumption that  $E$  is hard for large exponential time with exponential advice, reflects a belief that adding  $2^{\alpha \cdot n}$  bits of nonuniformity to a deterministic algorithm running in time  $2^{C_{\text{hard}} \cdot n}$  where  $0 < \alpha < 1$ , and  $C_{\text{hard}} > 1$  does not give it a “polynomial speedup” and in particular, does not enable the algorithm to solve every problem in  $\text{DTIME}(2^{C_{\text{easy}} \cdot n})$  for every constant  $C_{\text{easy}} > C_{\text{hard}}$ .

**A completeness argument.** The completeness argument explained in the previous section also gives that if the assumption holds for some *unknown* language  $L$  with respect to some specific *known* constants  $0 < \alpha < C_{\text{easy}} < C_{\text{hard}}$ , then the assumption also holds (for some  $\alpha' = \alpha - o(1)$ ,  $C'_{\text{hard}} = C_{\text{hard}} - o(1)$  and  $C'_{\text{easy}} = C_{\text{easy}} + o(1)$ ) with respect to the same language

$$L'_{C_{\text{easy}}} = \{(M, x) : \text{Turing machine } M \text{ accepts } x \text{ in } 2^{C_{\text{easy}} \cdot |x|} \text{ steps}\},$$

that was considered in the previous section. This means (that as in the previous section) in this scenario we can instantiate the assumption *without* having a specific  $L$  in mind.

## 8.2 Open Problems

A drawback of [Theorem 1.3](#) (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 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.

It should be noted that other constructions of two-source extractors are able to obtain error  $\varepsilon$  that is polynomially small in  $n$ , with a larger output length  $m$  (specifically, the 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.

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 detail in [Section 2](#)), 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 (as is done by Ben-Aroya, Doron and Ta-Shma [[BDT19](#)]).

However, we could hope to circumvent these limitations if we could find a way to use super-constant  $t$  (as is done in [[CZ16](#), [Li17](#)]) 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.

We remark that in order to get large output length in the final extractor for samplable distributions, 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](#).

## 9 Acknowledgments

We thank Dean Doron for some clarifications about [BDT19]. We thank Mohit Gurumukhani for bringing the non-malleable extractor of [Li23] to our attention. We thank anonymous reviewers for helpful comments.

## 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.
- [KS25] Oliver Korten and Rahul Santhanam. How to Construct Random Strings. In Srikanth Srinivasan, editor, *40th Computational Complexity Conference (CCC 2025)*, volume 339 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 35:1–35:32, Dagstuhl, Germany, 2025. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.
- [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.
- [Li23] Xin Li. Two source extractors for asymptotically optimal entropy, and (many) more. In *2023 IEEE 64th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1271–1281. IEEE, 2023.
- [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.