Electronic Colloquium on Computational Complexity, Report No. 104 (2025)

How to Construct Random Strings

Oliver Korten* and Rahul Santhanam†

July 29, 2025

Abstract

We address the following fundamental question: is there an efficient deterministic algorithm that, given 1^n, outputs a string of length n that has polynomial-time bounded Kolmogorov complexity $\tilde{\Omega}(n)$ or even $n - o(n)$?

Under plausible complexity-theoretic assumptions, stating for example that there is an $\epsilon > 0$ for which $\mathsf{TIME}[T(n)] \not\subseteq \mathsf{TIME}^{\mathsf{NP}}[T(n)^\epsilon]/2^{\epsilon n}$ for appropriately chosen time-constructible T, we show that the answer to this question is positive (answering a question of [RSW22]), and that the Range Avoidance problem [KKMP21, Kor21, RSW22] is efficiently solvable for uniform sequences of circuits with close to minimal stretch (answering a question of [ILW23]).

We obtain our results by giving efficient constructions of pseudo-random generators with almost optimal seed length against algorithms with small advice, under assumptions of the form mentioned above. We also apply our results to give the first complexity-theoretic evidence for explicit constructions of objects such as rigid matrices (in the sense of Valiant) and Ramsey graphs with near-optimal parameters.

*Department of Computer Science, Columbia University, New York, NY, USA. oliver.korten@columbia.edu
†Department of Computer Science, Oxford University, UK. rahul.santhanam@cs.ox.ac.uk

ISSN 1433-8092

1 Introduction

1.1 Motivation

Constructing Random Strings: Time-bounded Kolmogorov complexity K^T [LV19] is a fundamental measure of complexity of a Boolean string. Given a string x and a time bound T, $\mathsf{K}^T(x)$ is the size of the smallest program p such that $\mathcal{U}(p)$ outputs x within T steps, where $\mathcal{U}$ is a universal Turing machine fixed in advance. Intuitively, for feasible time bounds $T(n)$, $\mathsf{K}^{T(n)}(x)$ measures the inherent compressibility or "structure" in x from the point of view of efficient algorithms, in the sense that any x with low $\mathsf{K}^{T(n)}$ complexity has a short description from which it can be recovered efficiently. Thus a string with high K^T complexity can be considered unstructured or random-like.

Given functions $T, \ell : \mathbb{N} \to \mathbb{N}$ and a sequence $(x_n)_{n\in\mathbb{N}}$ of strings, where each x_n is of length n, let us call the sequence (K^T, ℓ)-hard if $\mathsf{K}^{T(n)}(x_n) \geq \ell(n)$ for all n. We are interested in the complexity of producing a (K^T, ℓ)-hard sequence of strings, for polynomial time bounds T and for ℓ as large as possible. Three particularly important settings we focus on here will be $\ell(n) = n^{\Omega(1)}$, $\ell(n) = \tilde{\Omega}(n)$, and $\ell(n) = n - o(n)$. A simple argument shows that K^T-hard strings cannot be produced in time $T' = o(T/\log(T))$ by an algorithm A which outputs x_n when given n in unary. Indeed, the existence of such an algorithm would imply that $K^T(x_n) = O(\log(n))$, as the universal machine $\mathcal{U}$ can output x_n in less than time T when given n in binary and a constant-size program for $\mathcal{A}$ [1]. But is it possible for such a sequence to be produced by an algorithm running in time poly(T)?

Question 1: Let $T : \mathbb{N} \to \mathbb{N}$ be any time-constructible function and let $\ell(n) \in \{n^{\Omega(1)}, \tilde{\Omega}(n), n - o(n)\}$. Is there a deterministic algorithm $\mathcal{A}$, which given input 1^n, runs in time poly(T) and outputs a (K^T, ℓ)-hard string?

Question 1 is a fundamental question about Kolmogorov complexity, for which we currently do not have clear positive or negative complexity-theoretic evidence. A positive answer to Question 1 would be very interesting, as it would demonstrate the power of having polynomially more time: a poly(T) time algorithm would be capable of producing highly T-incompressible strings, while a $o(T/\log(T))$ time algorithm can only produce strings that are highly T-compressible. Indeed Hypothesis 1.19 in a recent work of [RSW22] asks a very similar question[2] to Question 1, and states that "The plausibility of Hypothesis 1.19 remains to be investigated". Question 1 also has implications for the theory of meta-complexity, which has seen much recent work [All17, Hir22, LO22].

A natural approach to Question 1 is via pseudo-randomness. Under the assumption that E requires non-deterministic circuits of size $2^{\epsilon n}$ for some constant $\epsilon > 0$, it is known [IW97, KvM99] that for any constant $c > 0$ there is a pseudo-random generator G_n with seed length $O(\log(n))$ and error $o(1)$ against co-non-deterministic circuits of size n^c, computable in time poly(n). Consider $T(n) = n^d$, where $d < c$ is any constant. The property of being a K^T-incompressible string, i.e., a length-n string of K^T-complexity $\geq n - 1$, can be checked by a co-non-deterministic circuit of size at most n^c, and moreover at least half of all strings of length n satisfy the property. Hence close to half of the outputs of G_n satisfy the property of K^T-incompressibility, which means that the range of G_n is a poly(n) size set S_n computable in time poly(n) and containing at least one K^T-incompressible string for large enough n. However, it is unclear how to identify efficiently *which* of the strings in S_n is incompressible - this seems to require calls to an NP oracle. We could try

[1] Here we use the standard fact that any algorithm $\mathcal{A}$ running in time T' can be simulated by $\mathcal{U}$ in time $O(T' \log(T'))$.

[2] The difference is that instead of only asking about producing K^T-hard strings for time-constructible functions T, [RSW22] asks for an algorithm that gets T and n as input in unary, and outputs a K^T-hard string of length n.

and *combine* the strings in S_n together into a string that is K^t-hard, for example by concatenating them. This approach has two drawbacks:

1. Concatenating the strings yields a new string of length $>> T(n)$ rather than length n, since the set S_n has size $>> n^c$. Hence for any reasonable setting of $\ell(n)$, even $\ell(n) = n^{\Omega(1)}$, we do not get any interesting consequences for K^T-hardness when we are interested in time bounds as a *function* of input length.

2. Even if we don't mind destroying the relation between the time bound and input length, concatenating poly(n) strings which are maximally incompressible can, at best, achieve incompressibility $\ell(n) = n^{\Omega(1)}$. If we are in the more stringent compressibility settings $\ell(n) = \tilde{\Omega}(n)$ or $\ell(n) = n - o(n)$ which occur more frequently in explicit construction applications, concatenating poly(n) many strings appears to be useless.

Like most of the interesting problems in derandomization, answering Question 1 *unconditionally* would be difficult, as any $(\mathsf{K}^T, n^{\Omega(1)})$-hard string for $T > n^2$ can be transformed easily into the truth table of a hard Boolean function in E, i.e., a function with circuit complexity at least $2^{\epsilon m}$ for some ϵ, and hence would imply $\mathsf{E} \not\subseteq \mathsf{P}/\text{poly}$. Instead, we would simply like complexity-theoretic evidence in favor of or against Question 1. Preferably, this evidence should avoid exotic assumptions, and involve standard beliefs about the difficulty of simulating one resource efficiently by another, e.g., the belief that time cannot be simulated by much smaller space or by much smaller non-uniformity.

Explicit Constructions and Range Avoidance: Recently there has been a lot of interest [Kor21, KKMP21, RSW22, GLW22, ILW23, GGNS23, CGL$^+$23, CHLR23, CL24, CHR24, Li24, KP24] in the Range Avoidance problem, where the input is a circuit C from ℓ bits to n bits, where $n > \ell$, and the task is to find a non-output of C. This task is efficiently doable with randomness - we can just output a random string of length n, and this will be a non-output of C with probability at least $1/2$. The question is whether this can be done efficiently deterministically.

The problem was motivated in [Kor21, KKMP21] by its applications to *explicit constructions* of combinatorial objects. Let Π be a property, such as the property of a graph being Ramsey or of a matrix being rigid, that is satisfied by a random object with probability very close to 1. For many natural such Π, including the Ramsey and Rigidity properties, it can be shown that objects *not satisfying* Π can be efficiently recovered from a compressed representation. This recovery process can be modeled by a small Boolean circuit C for which any non-output is the representation of an object satisfying Π. In other words, solving Range Avoidance on C enables an efficient explicit construction for Π.

Given that explicit constructions for properties such as Ramsey and Rigidity have been long sought after, this motivates the study of the complexity of Range Avoidance, and in particular the search for evidence that Range Avoidance is feasible. Unfortunately it was shown in [ILW23] that under plausible cryptographic and complexity-theoretic assumptions, Range Avoidance is *intractable*.

However the intractability of Range Avoidance doesn't give evidence for the intractability of the explicit construction problems mentioned above - it could be that a *weaker* assumption than solving Range Avoidance is sufficient for efficient explicit constructions. Indeed such an assumption was identified in [ILW23] - solving Range Avoidance for *uniformly generated* circuits. It turns out that all of the explicit construction problems studied in [Kor21] can be captured by Range Avoidance on uniform circuits. This now raises the question of how tractable it is to solve Range Avoidance in this special case.

Question 2: Is there compelling complexity-theoretic evidence in favour of efficient explicit constructions for properties such as Ramsey and Rigidity, and more generally in favour of the tractability of Range Avoidance on uniform circuits?

Indeed the tractability of Range Avoidance on uniform circuits is raised explicitly in [ILW23]. It is also stated there that "some of the authors are skeptical that this special case of Avoid is easy."

The pseudorandomness approach to Question 1 can be applied to Question 2 as well. Essentially the same argument as we used there gives that, for properties such as Ramsey and Rigid, under standard derandomization assumptions, there is an efficient construction of a list of objects at least one of which satisfies the property. However, as we do not know polynomial-time algorithms for verifying the Ramsey or Rigidity properties, it is unclear how to pick out a single object satisfying the property from such a list. Indeed the pseudo-randomness approach applies to general Range Avoidance as well, but in that case we now have evidence that the problem is not tractable.

1.2 Our Results

We give positive answers to Questions 1 and 2, under believable complexity assumptions. Our hardness assumptions are of the following two forms, each involving a parameter $d \in \mathbb{N}$:

Hypothesis (Hardness Assumptions for d^{th}-Exponential Time Bounds)**.**

1. *(Strong Form) There is an absolute constant $\epsilon > 0$ so that the following holds. For any time constructible $T(n), m(n)$, with $T(n) \geq n$ having at most d^{th}-order exponential growth rate and $n \leq m(n) \leq \mathrm{poly}(n)$, there is a function $f : \mathbb{N} \mapsto \{0,1\}^*$, $|f(n)| = m(n)$ computable in $\mathrm{poly}(T(n))$ time, which cannot be computed in time $T(n)^\epsilon$ with $m(n)^\epsilon$ bits of advice by any NP-oracle machine for more than finitely many n.*

2. *(Weak Form) Let $T(n) = \exp^{[d+1]}(n)$. There is some $\epsilon > 0$ and a language computable in time $T(n)$ which is not computable in time $T(n)^\epsilon$ with an NP oracle and $2^{\epsilon n}$ bits of advice even infinitely often.*

We are using $\exp^{[d]}(n)$ for the d-fold iteration of the exponential function $\exp(n) = 2^n$. Such assumptions, particularly of the second kind, are fairly standard in complexity theory. Indeed, when $d = 0$, the second assumption is a standard derandomization assumption, asserting that there is a language in $\mathsf{TIME}[2^n]$ which does not have NP oracle circuits of size $2^{o(n)}$. The main novelty in our case is that we use these assumptions for larger values T, in particular for time bounds T which have iterated-exponential growth rate. While we state the strong form assumption by quantifying over all time constructible $T(n), m(n)$, in fact we only require the assumption to hold for quite a limited class of bounds: the bounds $T(n)$ needed are representable by constant length arithmetic expressions with $\exp, \log$, and ceiling/floor functions, while the bounds $m(n)$ needed are simply of the form $m(n) = \exp(c\lceil \log n \rceil)$ for constants c.

The first assumption is analogous to assumptions made in the recent literature on hardness vs randomness [CT21], and generalizes the second assumption in a natural way. Both the first and second assumptions involve natural beliefs about the relationships between the fundamental resources of time, non-determinism and advice. Both assumptions formalize the intuition that time cannot be sped up by an arbitrary polynomial amount using non-determinism and advice. Indeed, note that there is no known way to speed up time even by a *super-constant* amount by using non-determinism and advice.

We believe that our assumptions hold relative to a *random oracle*, and here is an informal argument for the case of the second assumption. Given oracle A, define the time T bounded Turing machine M^A to accept x of length n if the x'th string of length $T(n)$ is in A (where we consider the lexicographic order on strings). For a random oracle A, the truth table of A at length $T(n)$ is Kolmogorov-random even conditioned on truth tables below length $T(n)$. Since $T(n)^\epsilon$ time nondeterministic oracle machines can only access strings of A at length $T(n)^\epsilon$ or below, it should not be possible to compute the first 2^n bits of A at length $T(n)$ from this information together with arbitrary $T(n)^\epsilon$ bits of advice.

Our approach to using these assumptions in service of constructing random strings is via pseudorandomness, and is a variation on the strategy we critiqued a few paragraphs ago: construct a pseudorandom generator, enumerate its range, and concatenate the constituent strings. The problems with this approach discussed in the last section boiled down to one issue: for a standard complexity-theoretic PRG fooling polynomial size nondeterministic circuits, we can at best achieve a seed length of $O(\log n)$, and hence will end up needing to concatenate a very long list of $\mathrm{poly}(n)$ strings. To remedy the situation, it would suffice to use pseudorandom generators with much smaller seed length of $o(\log n)$ or even $\log \log n$. While it is known that $O(\log n)$ seed length is information-theoretically optimal for fooling *nonuniform* circuits of size $\mathrm{poly}(n)$, we aim to exploit the uniformity of the adversary we are fooling to get seed length *as small as possible*; for uniform adversaries the lower bound of $\log n$ no longer applies. This yields a result which is interesting in its own right: efficient constructions of small pseudo-random sets for uniform (or even slightly non-uniform) algorithms, under assumptions of the form discussed in the previous paragraph.

Theorem 1 (Informal, see Theorems 5 and 7)**.** *Under the Hypotheses above with $d = 1$, there is a pseudorandom generator with seed length $O(\log \log n)$ which fools $\mathsf{TIME}^{\mathsf{NP}}[\mathrm{poly}(n)]$; if we assume the strong form the generator runs in polynomial time, and if we assume the weak form it runs in quasipolynomial time.*

More generally, under the above hardness assumption with large parameters d (i.e. for higher-exponential time bounds), we obtain pseudorandom generators fooling $\mathsf{TIME}^{\mathsf{NP}}[\mathrm{poly}(n)]$ with seed length $O(\log^{[d+1]}(n))$ (for infinitely many n). If we assume the strong form hypothesis our generators will run in polynomial time, and if we assume the weak form they will run in iterated quasipolynomial time. When $d > 1$, our construction will require in addition that the input length n is of the form $n = \exp^{[d-1]}(n')$ for some n', or more generally that $\mathsf{K}^{\mathrm{poly}(n)}(n) \leq \log^{[d]}(n)$.

Some terms in the above require clarification. We are using $\log^{[d]}(\cdot)$ to denote the d-fold iteration of the logarithmic function. Our notion of "iterated quasipolynomial time" will be defined in the preliminaries - it describes a family of runtime bounds which grows faster than polynomial and quasipolynomial, but is much closer to polynomial than to exponential. Finally, the notation $\mathsf{K}^{\mathrm{poly}(n)}(n)$ denotes the time bounded Kolmogorov complexity of the *number* n, when written as a string in standard binary notation.

Our PRG construction can actually handle advice (nonuniformity) up to $\log^{[d]}(n)$ when the seed length is $\log^{[d+1]}(n)$, which we observe is optimal via a standard argument (Lemma 4). It also holds for fooling larger time bounds $\mathsf{TIME}^{\mathsf{NP}}[T(n)]$ for $T(n) >> \mathrm{poly}(n)$ (at the cost of the time complexity of the generator being low as a function of $T(n)$ rather than of n), and for other oracles $\mathcal{O}$ in place of the NP oracle (provided we use $\mathcal{O}$ in place of NP in the hardness assumption).

We then turn our attention briefly to PRGs fooling uniform algorithms *without* NP oracles: what is the minimal seed length such that we can fool $\mathsf{TIME}[T(n)]$ by a PRG with running time $T(n)^{O(1)}$? For this problem we show in Theorem 9 that the machinery developed above is unnecessary: if the PRG is capable of simulating the distinguishers it is trying to fool, there is a straighforward method which can reduce seed length $O(\log n)$ to arbitrarily small seed length. Hence under

the standard assumptions that achieve logarithmic seed length for $\mathsf{TIME}[T(n)]/n$ (namely that E requires exponential circuit size) we can obtain an arbitrarily strong reduction in seed length for $\mathsf{TIME}[T(n)]$. Like our NP-oracle result, this result applies to distinguishers with mild nonuniformity, and the achievable seed length scales with the uniformity in a way that is provably optimal according to Lemma 4.

Random Strings and Explicit Constructions: We next use our short-seed pseudorandom generators for P^{NP} (Theorem 1) to give (conditional) constructions of $\mathsf{K}^{\mathrm{poly}}$-random strings. Our first construction of random strings is the following, which achieves incompressibility $\tilde{\Omega}(n)$:

Theorem 2 (Informal, see Theorems 10 and 11)**.** *Under our main hardness assumption with $d = 1$, for any $k \in \mathbb{N}$ there is an algorithm $\mathcal{A}$ such that $\mathcal{A}(1^n)$ outputs an n-bit string x satisfying $\mathsf{K}^{n^k}(x) \geq \frac{n}{\mathsf{polylog}(n)}$ for all n; the algorithm will run in polynomial time if we use the strong form of the assumption, and quasipolynomial time if we use the weak form.*

More generally under our main hardness assumption for larger values of d, we obtain constructions of strings with $\mathsf{K}^{n^k}(x) \geq \frac{n}{\mathrm{poly}(\log^{[d]}(n))}$. Using the strong form of the assumption the construction runs in polynomial time, and under the weak form it runs in iterated quasipolynomial time. When $d > 1$ we require the integer n to be of the form $n = \exp^{[d-1]}(n')$ for some integer n', or more generally to satisfy $\mathsf{K}^{\mathrm{poly}(n)}(n) \leq \log^{[d]} n$.

If we are satisfied with our construction algorithms succeeding on some unknown infinite set of input lengths, we are able to bootstrap the above constructions so that the $\log^{[d]}(n)$ loss in K-complexity becomes additive rather than multiplicative:

Theorem 3 (Informal, see Theorem 12)**.** *Assume the first version of our main hardness assumption for all d. Then for any $k, d \in \mathbb{N}$ there is a polynomial time algorithm $\mathcal{A}$ such that $\mathcal{A}(1^n)$ outputs an n-bit string x satisfying $\mathsf{K}^{n^k}(x) \geq n - \log^{[d]} n$ for infinitely many n.*

In Section 4.2 we then use the previous results to give conditional polynomial time explicit constructions of various important combinatorial objects shown reducible to Range Avoidance in [Kor21]. We will defer most of the specifics to Section 4.2, but highlight the following two results:

Theorem (See Theorems 13 and 14)**.**

1. *Under the strong form of our main hardness assumption with $d = 2$, there is a $\mathrm{poly}(n)$ time algorithm which produces a matrix $M \in \mathbb{F}_2^{n \times n}$ which is Valiant-Rigid whenever n is a power of 2.*

2. *Assuming that there is a language in $\mathsf{TIME}[2^{2^n}]$ that is not computable in $\mathsf{TIME}^{\mathsf{NP}}[2^{\epsilon 2^n}]/2^{\epsilon n}$ (even infinitely often)[3], there is a language in EXP that requires Boolean circuits of size $\frac{2^n}{\mathrm{poly}(n)}$ for all n.*

Crucially, our results give the first standard-form hardness assumptions under which the above explicit construction problems (and others in Section 4.2) have an efficient algorithm, and they do so via a universal approach: we have a single object, $\mathsf{K}^{\mathrm{poly}}$-random strings, whose efficient explicit construction follows from plausible hardness assumptions, and which automatically satisfies various other pseudorandom properties, e.g. rigidity as a matrix or maximal hardness as a Boolean function.

Further Applications: In Sections 4.3 and 4.4 we discuss two further applications of our random

[3]This is a particular case of the weak form hardness assumption.

string constructions. The first shows a "hardness condensation" phenomenon for the main hardness assumptions we use in this paper (with the lower bound relativized to a PSPACE oracle), whereby we may amplify the degree of nonuniformity in the lower bounds from 2^{n^ϵ} to $2^{n-o(n)}$. The second addresses the question of the relative difficulty of *uniform* range avoidance and general (nonuniform) range avoidance posed in [RSW22, ILW23], where we observe using our main results and some previous hardness results ([ILW23, ABK24]) that under plausible hardness assumptions, nonuniform range avoidance is significantly harder than the uniform variant.

Barriers to Improving Our Main Results: In the final section we give two sets of results indicating barriers to improving our main PRGs and random string constructions. Our first result casts our iterated-logarithmic seedlength PRG fooling P^{NP} (Theorem 1) as a reconstructive *multi-source* extractor, similar to Trevisan's analysis [Tre01] of the classical hardness randomness connection in [NW94, IW97] in terms of reconstructive single-source extractors. We show that any such extractor must have iterated-logarithmic seed length, and thus that any improvement to the seed length of our PRG in Theorem 1 must use substantially different techniques.

Second, we give an oracle separation showing that it is not possible via black-box arguments to obtain our main results from more standard assumptions like $\mathsf{E} \not\subseteq \mathsf{SIZE}[2^{\epsilon n}]$. In [Kor21] it is shown that explicit construction of $(\mathsf{K}^{\mathrm{poly}}, n-1)$ random strings and $2^{\epsilon n}$ hard truth tables are equivalent with respect to NP oracle reductions. If such an equivalence held without NP oracles it would supersede our main results here: we could start with an assumption that E requires $2^{\epsilon n}$-size circuits, and then use the reduction to produce from the hard truth table a string with high time bounded Kolmogorov complexity. We show that no such reduction exists which is *relativizing*. This justifies in some sense the need for more high-end hardness assumptions to achieve our main results.

1.3 Overview of Main Construction

We describe at a high level our construction of pseudorandom generator with small seed length secure against any $L \subseteq \{0,1\}^n$ computable in $\mathsf{TIME}^{\mathsf{NP}}[\mathrm{poly}(n)]$. Using the classical hardness-randomness connection, under the assumption that $\mathsf{TIME}[2^{O(n)}]$ requires exponential size NP-oracle circuits we can obtain a generator $G^0 : \{0,1\}^{s_0(n)} \to \{0,1\}^n$ with seed length $s_0(n) \leq O(\log n)$. The key observation is that this first attempt at a generator significantly overshoots our primary goal in one respect: the generator G^0 obtained from the generic hardness-randomness connection would in fact be secure against $\mathsf{TIME}^{\mathsf{NP}}[\mathrm{poly}(n)]/n$ machines which have access to n bits of advice. On the other hand we only want a generator which fools *uniform* algorithms.

To use this to our advantage, we consider a recursive argument. Let $n_1 = s_0(n)$, and consider n_1 as our *new input length*. Define the language $L_1 \subseteq \{0,1\}^{n_1}$ consisting of the seeds z of G^0 such that $G^0(z) \in L$. We know that $\Pr_{z\sim\{0,1\}^{n_1}}[z \in L_1] \approx \Pr_{x\sim\{0,1\}^n}[x \in L]$ by the security of our first generator G_0, and therefore if we could find a second generator $G_1 : \{0,1\}^{s_1(n_1)} \to \{0,1\}^n$ which fools the language L_1, we would find that the composition $G_0 \circ G_1$ fools the original language L. If $s_1(n_1) = O(\log n_1) = O(\log \log n)$ we would have made significant progress. For such a generator G^1 to fool L_1, we would need it to fool NP-oracle algorithms with running time $\mathrm{poly}(n)$; however the input length of L_1 is $n_1 = O(\log n)$, so as a function of the new input length we need G^1 to fool algorithms running in *exponential time* with an NP oracle. We may proportionally increase the allowable runtime of G^1 to exponential as well; so overall we have scaled the time complexities of our generator/distinguisher by an exponential. However, crucially, the *advice complexity* of deciding L_1 remains $O(1)$: this is where we use crucially the uniformity of our distinguisher. If L required n bits of advice compute, then the best advice upper bound we could place on L_1 is $n \approx 2^{n_1}$ which is trivial.

Continuing in this way, we are able to iteratively reduce the seed length of our original generator by a logarithmic composition, for an arbitrary constant number of phases. The cost is that we need to obtain, in each step, psuedorandom generators with logarithmic seed length that fool algorithms running in $\exp(\exp(\cdots \exp(n) \cdots))$ time with an NP oracle, and which are computable in a comparable amount of time (without an NP oracle). To achieve such generators we rely on the standard toolkit of hardness-randomness transformations [NW94, IW97, Uma02]. In this way, we can argue the security of our construction based on the kind of hardness assumptions described above.

There are a few additional intricacies that we are skimming over here, which contribute to the two caveats in our main theorem statements: the weaker runtime bounds when using the second version of our hardness assumptions, and the requirement that the number n is of the form $n = \exp^{[d-1]}(n')$ for some n' when using the first version of our hardness assumptions. The first of these issues roughly stems from the fact that, when trying to apply our hardness assumption to get an $O(\log n_j)$ seed length generator on input length $n_j \approx \log^{[j]}(n)$ (which we will then compose with the outer generator to reduce the total seed length by another logarithm), we will actually need to apply our hardness assumption on some other input length $m_j = O(n_j)$; this will mean that we can at best hope for our generator to run in time $\mathrm{poly}(\exp^{[j]}(O(n_j))) \approx \exp^{[j]}(O(\log^{[j]} n))$, which will be superpolynomial when $j > 1$. This issue can be avoided by considering the more refined first version of our hardness hypothesis.

For the second issue, it turns out that we can only guarantee the security of our generator on inputs lengths n such that the *number* n itself has time-bounded Kolmogorov complexity $\log^{[d]}(n)$. For $d = 1$ this changes nothing, since every number n has $\mathsf{K}^{O(n)}(n) \leq \log n$ via its binary representation, so we achieve a generator with seed length $O(\log \log n)$ which is valid on all input lengths, but for large values of d our construction is only valid for infinitely many n (in particular it will work for all n of the form $\exp^{[d-1]}(n')$ for some n'). The reason for this issue is as follows: in the above exposition, we considered the language $L_1 \subseteq \{0,1\}^{n_1}$ of seeds mapping to strings in the base language L, and argued it is computable in time exponential in its input length. When we continue this argument for more steps and obtain languages $L_2, L_3, \ldots$, we would like to argue at each step that they are uniformly computable with a small amount of advice (comparable to the input length they are defined on). However, the definition of these languages depends on the original input length n that we started on, and since $\log^{[d]}(\cdot)$ is not injective for any $d > 0$, we cannot determine n solely from the smaller input lengths, but will need to somehow supply the number n as advice. If we assume the number n is sufficiently compressible, we are able to skirt this issue and complete the argument.

1.4 Related Work

We have already mentioned connections of our work to meta-complexity and Range Avoidance. In terms of derandomizing uniform algorithms, there has been a lot of work on uniform hardness-randomness tradeoffs, beginning with [IW98], but much of that work is not relevant to our setting as the emphasis is on uniform assumptions for derandomization, while we are interested instead in decreasing the seed length, and are not concerned with uniformity of the assumption.

However the recent work of [CT21] on super-fast derandomization is indeed very relevant from a technical point of view. They give a (conditional) derandomization of $\mathsf{BPTIME}[T(n)]$ into $\mathsf{TIME}[T(n)^{1+\epsilon} n]$. The classical approach to derandomizing $\mathsf{BPTIME}[T(n)]$ would be to model it as a circuit of size $T(n)$ with input length $T(n)$ (the input represents the randomness) and use a PRG fooling this class. To fool this class of circuits requires a PRG with seed length $\log T(n)$ and running time $T(n)$, and if $T(n) = n^3$ we could therefore not hope to achieve derandomization in

time $T(n)^{1+\epsilon}n$ no matter how fast the PRGs runtime is. A key observation in [CT21] is that that this approach to derandomizing $\mathsf{BPTIME}[T(n)]$ actually *overshoots* whats required: if we model the $\mathsf{BPTIME}[T(n)]$ machine more precisely as a $\mathsf{TIME}[T(n)]$ machine on input length $T(n)$ with nonuniformity $n << T(n)$, then we only need a PRG which fools $\mathsf{TIME}[T(n)]/n$; for this task, it is possible (at least information-theoretically) to achieve a seed length of $\log n$ or more generally $(1+\epsilon)\log n$ rather than $\log T(n)$, which means that the enumeration of seeds will only cost us $n^{1+\epsilon}$ time.

They then proceed to construct a PRG for $\mathsf{TIME}[T(n)]/n$ with seed length $(1+\epsilon)\log n$ and computable in time $T(n)^{1+\epsilon}$. In their case a great deal of work must be dedicated to optimizing the runtime of the PRG which has no relevant parallel in our work; however the question of reducing the seed length down to the true level of uniformity, and the assumptions under which they are able to achieve it, have similarities to ours. In particular, they require assumptions of the form $\mathsf{TIME}[T(n)] \not\subseteq \mathsf{TIME}[T(n)^{1-\delta}]/2^{(1-\delta)n}$ where $T(n) = 2^{kn}$ for a large constant k.

The innovation in our work is the use of recursion to achieve dramatically smaller seed length against inefficient uniform adversaries using a small amount of non-uniformity, which enables us to address Questions 1 and 2.

2 Preliminaries

We start with some notation for the classes of growth rates appearing in this work:

Definition 1 (Growth Rates)**.** *Both $\log(\cdot), \exp(\cdot)$ are base 2. For any function $f : \mathbb{N} \to \mathbb{N}$, $d \in \mathbb{N}$ we use $f^{[d]}$ to denote the d-ary composition of f with itself, with $f^{[0]}$ being the identity $n \mapsto n$. We say that a function f has elementary growth rate, or that it is "elementary," if $f(n) \leq \exp^{[d]}(n)$ for some absolute constant d. We say that a funciton $f : \mathbb{N} \to \mathbb{N}$ is time-constructible if there is a Turing machine M such that $M(1^n)$ prints $f(n)$ in binary, and M has running time $O(f(n))$ on all inputs.*

For each $d \in \mathbb{N}$ and $\alpha \in \mathbb{R}_+$ we define the function $\Phi_{d,\alpha}(n) = \exp^{[d]}(\log(\alpha \log^{[d-1]}(n)))$. Using this function we define a hierarchy of $O(\cdot)$ notations more relaxed than the standard as follows; for $f, g : \mathbb{N} \to \mathbb{N}$, we say $f(n) = O^{\langle d \rangle}(g(n))$ if $f(n) \leq \Phi_{d,C}(g(n))$ for some $C \in \mathbb{N}$ and sufficiently large n, and $f(n) = \Omega^{\langle d \rangle}(g(n))$ if $f(n) \geq \Phi_{d,\epsilon}(g(n))$ for some $\epsilon > 0$ and sufficiently large n. $o^{\langle d \rangle}(g(n))$, $\omega^{\langle d \rangle}(g(n))$ are defined analogously.

In this notation we have $O^{\langle 1 \rangle}(n) = O(n)$, $O^{\langle 2 \rangle}(n) = \mathrm{poly}(n)$, $O^{\langle 3 \rangle}(n) = \mathrm{quasipoly}(n)$. This hierarchy of growth rates enumerates a class of strongly subexponential growth rates having magnitude much closer to n than to 2^n: indeed for each fixed d, $O^{\langle d \rangle}(n)$ is closed under composition and grows slower than $\Omega^{\langle d' \rangle}(2^n)$ for any fixed d'. Similarly, $\Omega^{\langle 1 \rangle}(n) = \Omega(n)$, $\Omega^{\langle 2 \rangle}(n) = n^{\Omega(1)}$, $\Omega^{\langle 3 \rangle}(n) = 2^{\log^{\Omega(1)} n}$, and for each fixed d the class $\Omega^{\langle d \rangle}(n)$ has a growth rate much closer to n than to $\log n$, and in particular far exceeds the growth rate of $O^{\langle d' \rangle}(\log n)$ for any fixed d'.

Definition 2 (Languages and Complexity Classes)**.** *For functions $f, g : \mathbb{N} \to \mathbb{N}$ and language $A \subseteq \{0,1\}^*$ we define the complexity class $\mathsf{TIME}^A[f(n)]/g(n)$ consisting of those languages decidable in time $O(f(n))$ with an oracle for the language A and using $O(g(n))$ bits of advice on inputs of length n. When $g(n) = 0$ we omit this argument.*

For a language $L \subseteq \{0,1\}^*$ we use $L_n : \{0,1\}^n \to \{0,1\}$ to denote the restriction of L to $\{0,1\}^n$, and $\mathsf{SIZE}[f(n)]$ to denote the set of languages L so that L_n is computed by Boolean circuits of size $O(f(n))$ for all n.

We next define our notation for time-bounded Kolmogorov complexity:

Definition 3 (Kolmogorov Complexity)**.** *We fix an efficient universal oracle turing machine $\mathcal{U}$ once and for all. For an oracle language $\mathcal{O} \subseteq \{0,1\}^*$, time bound $T \in \mathbb{N}$, $x, y \in \{0,1\}^*$ we use $\mathsf{K}^{T,\mathcal{O}}(x \mid y)$ to denote the length of the shortest program $\pi \in \{0,1\}^*$ so that $\mathcal{U}^{\mathcal{O}}(\pi, y)$ halts with output x in at most T time steps. If $\mathcal{O} = \{\}$ or y is the empty string we omit them from the notation as in $\mathsf{K}^T(x \mid y)$, $\mathsf{K}^{T,\mathcal{O}}(x)$ respectively.*

For a number $n \in \mathbb{N}$, we use $\mathsf{K}^{T,\mathcal{O}}(n)$ to denote $\mathsf{K}^T(\mathrm{bin}(n))$ where $\mathrm{bin}(n) \in \{0,1\}^{\lceil \log n \rceil}$ is the canonical binary representation of n.

Finally we introduce relevant notation for pseudorandom generators:

Definition 4 (Pseudorandom Generators)**.** *For a family of "distinguishers" $\mathcal{D} \subseteq \{0,1\}^{\{0,1\}^n}$ and $\epsilon \in [0,1]$, we say that $G : \{0,1\}^s \to \{0,1\}^n$ is a pseudorandom generator (PRG) secure against $\mathcal{D}$ with error ϵ if, for all $D \in \mathcal{D}$, we have*

$$|\Pr_{x \sim \{0,1\}^n}[D(x) = 1] - \Pr_{z \sim \{0,1\}^s}[D(G(z)) = 1]| \leq \epsilon$$

We say that G is a hitting set generator (HSG) if it satisfies the weaker condition

$$(\Pr_{x \sim \{0,1\}^n}[D(x) = 1] > \epsilon) \Rightarrow (\Pr_{z \sim \{0,1\}^s}[D(G(z)) = 1] > 0)$$

Let $s : \mathbb{N} \to \mathbb{N}$, $\epsilon : \mathbb{N} \to [0,1]$. If $\mathcal{C} \subseteq \{0,1\}^{\{0,1\}^*}$ is a complexity class (set of languages) and $G = (G_n : \{0,1\}^{s(n)} \to \{0,1\}^n)_{n \in \mathbb{N}}$ is an ensemble of generators, we say that G is a PRG (resp. HSG) secure against $\mathcal{C}$ if, for all $L \in \mathcal{C}$, there is $n_0 \in \mathbb{N}$ so that for all $n > n_0$, G_n is a PRG (resp. HSG) secure against $\{L_n\}$ with error $\epsilon(n)$.

2.1 Range Avoidance and Construction of Random Strings

We formalize a general notion of explicit construction problems as follows:

Definition 5 (Explicit Construction Problems)**.** *An explicit construction problem is defined by a language $\Pi \subseteq \{0,1\}^*$, such that $\Pi_n \neq \emptyset$ for all n. The computational task is: given 1^n as input, output a string $x \in \Pi_n$.*

If Π is an explicit construction problems, we say that a function $S : \{0,1\}^* \to \{0,1\}^*$ is a "list solution" to Π if $S(1^n) \cap \Pi_n \neq \emptyset$ for all n. The "list size" $\ell(\cdot)$ is defined as $\ell(n) = |S(1^n)|$.

As discussed in the introduction, an important class of explicit construction problems are those reducible to the problem *range avoidance*:

Definition 6 (Range Avoidance [KKMP21, Kor21, RSW22])**.** *Range avoidance, or "Avoid," is the following search problem: given a Boolean circuit $C : \{0,1\}^m \to \{0,1\}^n$ with $m < n$, output a string $x \in \{0,1\}^m \setminus \mathrm{range}(C)$. We say that this Avoid instance has "stretch" $m \mapsto n$.*

We say that an explicit construction problem Π reduces to Avoid in polynomial time with stretch function $\ell(n)$, if there is a polynomial time algorithm which, given 1^n, outputs an Avoid instance $C_n : \{0,1\}^{\ell(n)} \to \{0,1\}^n$ so that whenever x is a solution for C_n, we have $x \in \Pi$.

The key connection between $\mathsf{K}^{\mathrm{poly}}$ random strings and explicit construction problems reducible to Range Avoidance is the following:

Observation 1 ([RSW22])**.** *Say that for every $k \in \mathbb{N}$, there is a polynomial time algorithm which, for every n (resp. infinitely many n) outputs a string $x \in \{0,1\}^n$ with $\mathsf{K}^{n^k}(x) \geq \ell(n)$. Then every explicit construction problem reducible to Avoid with stretch function $\ell(n)$ is solvable in polynomial time .*

Proof. The definition of the reduction implies that there is a polynomial time algorithm $C_n : \{0,1\}^{\ell(n)} \to \{0,1\}^n$, so that whenever $x \notin \text{range}(C_n)$ we have $x \in \Pi$. If k is such that the algorithm constructing C_n runs in time n^k, then we observe that every string x in the range of C_n satisfies $\mathsf{K}^{n^k}(x) \leq \ell(n)$. □

Recall that our approach to constructing K^{poly} random strings will consist of two steps. First, we will try only to construct a short list of strings, one of which is guaranteed to have K^{poly} complexity $\geq n-1$: we then concatenate the list to obtain a single string whose complexity degrades by a factor proportional to the length of the list. The second step (concatenation) is justified by the following standard claim:

Observation 2. *Let $x^1, \ldots, x^m \in \{0,1\}^n$ and let $\hat{x} = (x^1, \ldots, x^m)$ be their concatenation. Then for any time bound T and $i \leq m$ we have $\mathsf{K}^{T-O(mn)}(\hat{x}) \geq \mathsf{K}^T(x^j) - O(\log m)$.*

For the first step (obtaining a short list of candidate solutions), we rely on the following observation:

Observation 3. *Let $(G_n : \{0,1\}^{s(n)} \to \{0,1\}^n)_{n\in\mathbb{N}}$ be a hitting set generator secure against $\mathsf{TIME}^{\mathsf{NP}}[O(T(n))]$ with error $\frac{1}{2}$. Then there exists a seed $z \in \{0,1\}^{s(n)}$ so that $\mathsf{K}^{T(n)}(G_n(z)) \geq n-1$; in other words the range of G_n is a list solution for the set of strings with $\mathsf{K}^{T(n)}(\cdot) \geq n-1$.*

Proof. At least half of n bit strings have $\mathsf{K}^{T(n)}(\cdot) \geq n-1$. On the other hand $\mathsf{K}^{T(n)}(\cdot)$ is computable in time $O(T(n))$ with an NP oracle. □

Hence, we have reduced the problem of constructing random strings to constructing hitting generators against P^{NP} with short seed length. The next section is dedicated to the construction of such a generator under plausible hardness assumptions. We will in fact produce a pseudorandom generator (despite only needing a hitting set generator).

3 PRGs for Uniform Classes with Near-Optimal Seed Length

The main goal in this section is to produce a pseudorandom generator computable in a $\text{poly}(n)$ time, or more generally $O^{\langle d \rangle}(n)$ time for some fixed constant d, which is secure against uniform P^{NP} algorithms and has seed length significantly smaller than $\log n$. We will phrase all of our assumptions/generator construction with respect to an arbitrary oracle $\mathcal{O}$ in place of NP for the sake of generality.

We will consider here two qualitatively distinct classes of hardness assumption used to instantiate our generators, each parameterized by an oracle $\mathcal{O}$ (typically we set $\mathcal{O} = \mathsf{NP}$) and a constant $d \in \mathbb{N}$:

Hypothesis 1 (Strong Assumption for $\mathcal{O}, d$, abbreviated $\mathsf{SH}(\mathcal{O}, d)$)**.** *There is $\epsilon > 0$ so that for all time constructible $T(n) \leq (\exp^{[d]}(n))^{O(1)}$ and all time constructible $m : \mathbb{N} \to \mathbb{N}$, $n \leq m(n) \leq \text{poly}(n)$ the following holds: there is a sequence of strings $(f_n \in \{0,1\}^{m(n)})_{n\in\mathbb{N}}$ so that the map $n \mapsto f_n$ is computable uniformly in $T(n)$ time, but no machine running in time $T(n)^\epsilon$ with an $\mathcal{O}$ oracle and $m(n)^\epsilon$ bits of advice can compute f_n for more than finitely many n. In the case $T(n) \leq \text{poly}(n)$, we only require the assumption to hold in the case $m(n) = n$.*

We refer to $m(n)$ as the "length bound" in the above.

Hypothesis 2 (Weak Assumption for parameters $\mathcal{O}, d, v,\ d \geq 1$, abbreviated $\mathsf{WH}(\mathcal{O}, d, v)$)**.** *Let $1 \leq d' \leq d$ and $T(n) = \exp^{[d']}(n)$. There is some constant $\epsilon > 0$ and a language L computable in $\mathsf{TIME}[T(n)]$ which is not computable in $\mathsf{TIME}^{\mathsf{NP}}[\Phi_{v,\epsilon}(T(n))]/\Phi_{v,\epsilon}(2^n)$ on more than finitely many input lengths.*

Note that $\mathsf{WH}(\mathcal{O}, 1, 2)$ translates to the assumption that E requires $2^{\Omega(n)}$ circuit complexity with $\mathcal{O}$ oracles, which is the standard regime in which polynomial time generators fooling $\mathsf{P}^{\mathcal{O}}$ with logarithmic seedlength can be constructed by known methods.

3.1 Fooling $\mathsf{TIME}^{\mathsf{NP}}[T(n)]$ with $O(\log n)$ Seed Length

The first step in our construction is to use classical hardness-randomness constructions to give pseudorandom generators which fool $\mathsf{TIME}^{\mathsf{NP}}[T(n)]$ with $O(\log n)$ seed length and runtime $\approx T(n)$ in the case $T(n)$ is very large, under the appropriate hardness assumptions. Depending on which assumption we use we get a generator with different parameters:

Lemma 1. *Assume $\mathsf{SH}(\mathcal{O}, d)$. Then for every time constructible $T(n) \leq (\exp^{[d]}(n))^{O(1)}$ there is a PRG $(G_n : \{0,1\}^{s(n)} \to \{0,1\}^n)_{n\in\mathbb{N}}$ computable uniformly in time $\mathrm{poly}(T(n))$ which fools $\mathsf{TIME}^{\mathcal{O}}[T(n)]/3n$.*

Lemma 2. *Assume $\mathsf{WH}(\mathcal{O}, d+1, v)$, $d \geq 0$, $v \geq 2$. Then for any $k \in \mathbb{N}$ there is a PRG $(G_n : \{0,1\}^{s(n)} \to \{0,1\}^n)_{n\in\mathbb{N}}$ computable uniformly in time $O^{\langle d+v\rangle}(n)$ which fools $\mathsf{TIME}^{\mathcal{O}}[n^k]/3n$ and has seed length $s(n) \leq O^{\langle v-1\rangle}(\log n)$.*

We will prove the first case (Lemma 1) here, and reserve Lemma 2 for the Appendix as it's proof is similar. We require the following standard tool from complexity-theoretic pseudorandomness:

Theorem 4 (Black-Box Hardness-Randomness Connection [IW97])**.** *For each $\epsilon > 0$ exists a constant $c \in \mathbb{N}, c \geq \frac{3}{\epsilon}$ and a uniform algorithm IW with the following behavior. On input n and given oracle access to a function $f : \{0,1\}^{\ell(n)} \to \{0,1\}$ with $\ell(n) = c \cdot \lceil \log n \rceil$, IW^f computes a function $\mathsf{IW}^f : \{0,1\}^{s(n)} \to \{0,1\}^n$ in $\mathrm{poly}(n)$ time for $s(n) \leq O(\ell(n))$ such that for any $D : \{0,1\}^n \to \{0,1\}$ with*

$$|\Pr_{x\sim\{0,1\}^n}[D(x) = 1] - \Pr_{z\sim\{0,1\}^{s(n)}}[D(\mathsf{IW}^f(z)) = 1]| \geq \frac{1}{n}$$

there exists a circuit C with D oracle gates computing f whose total size is at most $2^{\epsilon\ell(n)}$.

Proof of Lemma 1. We will invoke $\mathsf{SH}(\mathcal{O}, d)$ on some yet to be determined time bound $T'(n)$ and length bound $m(n)$, with respect to oracle $\mathcal{O}$ and constant d; let $\epsilon > 0$ be the implied constant guaranteed by the hypothesis (which does not depend on the choice of T', m).

Next, invoke Theorem 4 with parameter $\delta := \epsilon/2$, and let $c \in \mathbb{N}$ be the guaranteed constant from this Theorem. So there is a function $\mathsf{IW}^f : \{0,1\}^{s(n)} \to \{0,1\}^n$ which, given oracle access to a function $f : \{0,1\}^{\ell(n)} \to \{0,1\}$ with $\ell(n) = c \cdot \lceil \log n \rceil$, runs in $\mathrm{poly}(n)$ time with $\mathrm{poly}(n)$ oracle calls to f, and such that whenever $D : \{0,1\}^n \to \{0,1\}$ distinguishes IW^f from uniform, there is a D-oracle circuit C of size $2^{\delta\ell(n)}$ computing f.

Now we set $m(n) = 2^{\ell(n)} = 2^{c\cdot\lceil\log n\rceil}$ and let $T'(n) = T(n)^{3k}$, which are both time constructible, and use these in our specific invocation of $\mathsf{SH}(\mathcal{O}, d)$ as hinted previously. We then obtain an algorithm which, in time $T(n)^{3k}$, computes a string $f_n \in \{0,1\}^{m(n)}$, which cannot be computed by by any machine running in time $T'(n)^\epsilon = T(n)^3$ with $m(n)^\epsilon = 2^{\epsilon\cdot\ell(n)}$ bits of advice. We then claim that, setting $G_n = \mathsf{IW}^{f_n} : \{0,1\}^{\ell(n)} \to \{0,1\}^n$ gives the required generator. By assumption it is computable in time $T(n) \cdot \mathrm{poly}(n) \leq \mathrm{poly}(T(n))$. On the other hand if it were distinguished in

time $T(n)$ with $3n$ bits of advice, we'd obtain a circuit computing every bit of f_n of size $3n + 2^{\frac{\epsilon}{2}\ell(n)}$ with oracle gates that can be evaluated in time $T(n)$, hence overall we would be able to compute f_n in time $T(n) \cdot m(n)^{\frac{\epsilon}{2}} + O(n \cdot T(n)) = T(n) \cdot n^{\frac{1}{2}} + O(T(n) \cdot n) < n^3$ with $3n + m(n)^{\frac{\epsilon}{2}} < m(n)^{\epsilon}$ bits of advice, provided $n = o(m(n)^{\epsilon})$. Recalling from our application Theorem 4 that $c\delta \geq 3$, we have that $m(n)^{\frac{\epsilon}{2}} \geq \Omega(n^{\frac{c\epsilon}{2}}) = \Omega(n^{\frac{3}{2}})$ and we are done. □

3.2 Recursive Generator Construction

We give here our main construction of a generator fooling languages decidable in polynomial time with an NP oracle under a natural computational hardness assumption, specifically the assumptions $\mathsf{SH}(\mathsf{NP}, d)$, $d \in \mathbb{N}$. Our construction will have seed iterated-logarithm type seed length length $\log^{[O(1)]} n$ for infinitely many n; in particular it will work for all n such that the number n has time-bounded Kolmogorov complexity comparable to the seed length. Afterwords we show that a similar construction gives a generator fooling the same class with a slower runtime under the weak form hardness assumptions; the proof of this construction will be relegated to the appendix.

Theorem 5. *Let $\mathcal{O} \subseteq \{0,1\}^*$ be any oracle, $d, v \in \mathbb{N}$, and assume Hypothesis $\mathsf{SH}(\mathcal{O}, d+v)$. Let $T(n) \leq (\exp^{[v]}(n))^{O(1)}$ be time constructible. There exists a generator $(\mathcal{G}_n^d : \{0,1\}^{s_d(n)} \to \{0,1\}^n)_{n\in\mathbb{N}}$ computable uniformly in $\mathrm{poly}(T(O(n)))$ time, so that the following hold:*

1. *For all n, $s_d(n) \leq O(\log^{[d+1]}(n))$*
2. *For all $L \in \mathsf{TIME}^{\mathcal{O}}[T(n)]/\log^{[d]}(n)$, we have*

$$\left| \Pr_{z\sim\{0,1\}^{s(n)}}[\mathcal{G}_n(z) \in L] - \Pr_{x\sim\{0,1\}^n}[x \in L] \right| \leq O\left(\frac{1}{\log^{[d]}(n)}\right)$$

for all but finitely many $n \in \{n \mid \mathsf{K}^{T(n)}(n) \leq \log^{[d]}(n)\}$.

Proof. Using our hardness assumption $\mathsf{SH}(\mathcal{O}, d+v)$ in combination with Lemma 1, for each time constructible bound $R : \mathbb{N} \to \mathbb{N}$ bounded above by $(\exp^{[d+v]}(n))^{O(1)}$ there is a constant $c(R) \in \mathbb{N}$ and a pseudorandom generator $(G_n^{(R)} : \{0,1\}^{c(R)\lceil \log n \rceil} \to \{0,1\}^n)_{n\in\mathbb{N}}$ computable in $\mathsf{TIME}(R(n)^{c(R)})$ which fools $\mathsf{TIME}^B[R(n)]/3n$ with error $\frac{1}{n}$. Using $c(\cdot)$ we may define a sequence of constants $(c_i)_{i\in\mathbb{N}}$ and functions $(T_d, f_d : \mathbb{N} \to \mathbb{N})_{d\in\mathbb{N}\cup\{0\}}$ as follows:

1. $T_0 = T$, $f_0 = (x \mapsto x)$.
2. Set $c_d = c(T_{d-1})$, $f_d = c_d \cdot \lceil \log f_{d-1}(\cdot) \rceil$, $T_d = 4 \cdot T_{d-1}(\exp(\lceil \frac{\cdot}{c_d} \rceil))^{c_d}$

Observe that $3T_{d-1}(f_{d-1}(n)) + T_{d-1}(f_{d-1}(n))^{c_d} \leq T_d(f_d(n))$.

Then for each d, we have the generator $(G_n^{(T_d)})_{n\in\mathbb{N}}$ defined for every input, which we abbreviate as G^d. We use this family of generators to construct the generators $(\mathcal{G}_n^d)_{n\in\mathbb{N}}$ promised in the theorem statement. In particular, we define for every $d \in \{0, \ldots, \}$ the generator

$$(\mathcal{G}_n^d : \{0,1\}^{f_{d+1}(n)} \to \{0,1\}^n)_{n\in\mathbb{N}}, \quad \mathcal{G}_n^d = G_n^0 \circ G_{f_1(n)}^1 \circ \cdots \circ G_{f_i(n)}^d$$

We prove its security by induction, using the following stronger induction hypothesis for each $d \geq 0$:

1. $\mathcal{G}_n^d$ is computable in time $\leq T_{d+1}(f_{d+1}(n)) - 3T(n)$ with the number n and $O(1)$ additional bits as advice

2. $\mathcal{G}_n^d$ fools any $L \subseteq \{0,1\}^n$, $L \in \mathsf{TIME}^B[T_d(f_d(n))]/f_d(n)$ with error $\leq \sum_{j \leq d} f_j(n)^{-1}$ for all sufficiently large n satisfying $K^{T(n)}(n) \leq f_d(n)$

For the base case, we know that $\mathcal{G}_n^0 = G_n^0$ which, by assumption, fools $\mathsf{TIME}^B[T(n)]/n$ with error $\frac{1}{n}$. Recall that $T_1(f_1(n)) \geq 3T_0(f_0(n)) + T_0(f_0(n))^{c_1} = 3T(n) + T(n)^{c_1}$, hence $\mathcal{G}_n^0 = G_n^0$ is computable in time $T(n)^{c_1} \leq T_1(f_1(n)) - 3T(n)$ (with no advice). Now assume the induction hypothesis up to $d-1$. Let $L \subseteq \{0,1\}^n$ be given, n sufficiently large, so that $L \in \mathsf{TIME}^B[T_d(f_d(n))]/f_d(n)$ and and $\mathsf{K}^{T(n)}(n) \leq f_d(n)$. Define $L_d \subseteq \{0,1\}^{f_d(n)}$, $L_d = \{x \mid \mathcal{G}^{d-1}(x) \in L\}$. By induction, $\mathcal{G}^{d-1}$ is computable in time $T_d(f_d(n)) - 3T(n)$ given the number n as advice, and $O(1)$ additional advice bits; hence L_d is computable in $\mathsf{TIME}^B[T_d(f_d(n))]/3f_d(n)$: if we are given the number n as advice, then we may determine $f_0(n), \ldots, f_d(n)$ using the number d as advice which costs $O(1)$, after which we can compute the generator $\mathcal{G}_n^{d-1}$, and the language L using $\leq f_d(n)$ additional bits of advice (together with some $O(1)$ bits to specify the algorithm described in the current sentence). Under the assumption $\mathsf{K}^{T(n)}(n) \leq \log^{[i]}(n) \leq f_i(n)$ we may produce the number n from an additional $f_i(n)$ bits of advice, for a total advice cost $\leq 3f_i(n)$, and the time of the additional operations (beyond the original computation of $\mathcal{G}^{d-1}$) is bounded by $3T(n)$.

Now, using the second part of our induction hypothesis, we determine that $\mathcal{G}_n^{i-1}$ fools L, in particular we have:

$$|\Pr_{z \in \{0,1\}^{f_i(n)}}[\mathcal{G}_n^{i-1}(z) \in L] - \Pr_{x \in \{0,1\}^n}[x \in L]| \leq \sum_{j<i} f_j(n)^{-1}$$

On the other hand, using the security of the generator G_n^d on input length $f_{d+1}(n)$ (which is sufficiently large), we have that G_n^d fools L_d, in particular:

$$|\Pr_{w \in \{0,1\}^{f_{d+1}(n)}}[G_n^d(w) \in L_i] - \Pr_{z \in \{0,1\}^{f_d(n)}}[z \in L_d]| \leq f_d(n)^{-1}$$

Recalling that $\mathcal{G}_n^d = \mathcal{G}_n^{d-1} \circ G_n^d$ and combining the previous two inequalities using the triangle inequality we have:

$$|\Pr_{z \in \{0,1\}^{f_{d+1}(n)}}[\mathcal{G}_n^d(w) \in L] - \Pr_{x \in \{0,1\}^n}[x \in L]| \leq \sum_{j \leq d} f_j(n)^{-1}$$

which establishes the second condition in our inductive hypothesis. For the first, note that to compute $\mathcal{G}_n^d$, we need only recover the number n from its shortest $\mathsf{K}^T(\cdot)$ description in time $T(n)$, compute $\mathcal{G}_n^{i-1}$ which takes time $\leq T_i(f_i(n)) - 3T(n)$ by induction, and finally compute G_n^i which takes time $T_i(f_i(n))^{c_{i+1}}$; overall this is bounded by $T_{i+1}(f_{i+1}(n)) - 3T(n)$.

At this point the theorem is proven; it remains to verify a few bounds:

1. $\sum_{j \leq d} f_j(n)^{-1} \leq O(f_d(n)^{-1})$ so that the error bound is as stated in the theorem.
2. $f_d(n) \leq O(\log^{[d]}(n))$
3. $T_d(f_d(n)) \leq \mathrm{poly}(T(O(n)))$

The first and second hold trivially. For the last, we prove it by induction; more specifically we will show by induction that $T_d(O(f_d(n))) \leq T(O(n))^{O(1)}$. In the base case, $T_0(O(f_0(n))) = T(O(n))$. We then have that

$$T_d(O(f_d(n))) = 4 \cdot T_{d-1}\Big(O(\exp(\lceil \frac{c_d \lceil \log f_{d-1} \rceil}{c_d} \rceil))\Big)^{c_d} \tag{1}$$

$$\leq \Big(T_{d-1}(O(\exp(\log f_{d-1}(n) + O(1)))\Big)^{O(1)} \tag{2}$$

$$= T_{d-1}(O(f_{d-1}(n)))^{O(1)} \leq T(O(n))^{O(1)} \tag{3}$$

where the last step uses the induction hypothesis. □

We highlight an important special case of the above:

Theorem 6. *Assume that for some $\epsilon > 0$ and every time constructible $T(n) \leq 2^{O(n)}$, $m(n) \leq \mathrm{poly}(n)$, there is a function $n \mapsto \{0,1\}^{m(n)}$ computable uniformly in time $T(n)$ which is not computable in $\mathsf{TIME}^{\mathsf{NP}}[T(n)^{\epsilon}]/m(n)^{\epsilon}$ even infinitely often. Then for every $k \in \mathbb{N}$, there is a pseudorandom generator family $(\mathcal{G}_n)_{n\in\mathbb{N}}$ with seed length $O(\log\log n)$ which fools $\mathsf{TIME}^{\mathsf{NP}}[n^k]/\log n$ for all sufficiently large n.*

Proof. We will set $v = 0$, $d = 1$, $T(n) = n^k$ in Theorem 5. Observe that for all n, we have $\mathsf{K}^{O(n)}(n) \leq \log n$ since we may encode the number n in binary. □

If we rely instead on the weak form hypotheses $\mathsf{WH}(\cdot)$ we get:

Theorem 7. *Assume Hypothesis $\mathsf{WH}(\mathcal{O}, d+1, v)$ with $d \geq 0$, $v \geq 2$ fixed constants, and $k \in \mathbb{N}$ a fixed constant. Then there is a pseudorandom generator with seed length $O^{\langle v-1\rangle}(\log^{[d+1]}(n))$ and runtime $O^{\langle d+v\rangle}(n)$ that fools $\mathsf{TIME}^{\mathcal{O}}[n^k]/\log^{[d]}(n)$ with error $(2\log^{[d]}(n))^{-1}$ whenever $\mathsf{K}^{n^k}(n) \leq \log^{[d]}(n)$.*

As in the case of Lemma 2, we relegate the proof to the appendix since it uses essentially the same ideas as that of Theorem 5 and is in fact a bit simpler. As before we highlight an important special case, obtained immediately from Theorem 7 by setting $\mathcal{O} = \mathsf{NP}$, $d = 1$, $v = 2$:

Theorem 8. *Assume that there is some $\epsilon > 0$ and a language $L \in \mathsf{TIME}[2^{2^n}]$ which is not computable in $\mathsf{TIME}^{\mathsf{NP}}[2^{\epsilon 2^n}]/2^{\epsilon n}$ on more than finitely many input lengths. Then for any $k \in \mathbb{N}$ there is a pseudorandom generator fooling $\mathsf{TIME}^{\mathsf{NP}}[n^k]$ with seedlength $O(\log\log n)$ and quasipolynomial runtime.*

3.3 Fooling Uniform Deterministic Time

For our application to construction of random strings, we will use the generator in the previous section with oracle setting $\mathcal{O} = \mathsf{NP}$. It is nonetheless natural to consider the implications of our generator in the case $\mathcal{O} = \emptyset$; in this case we obtain a generator with $o(\log n)$ seed length fooling $\mathsf{TIME}[T(n)]$ under plausible hardness assumptions. However, we demonstrate here that in the regime of a deterministic distinguisher, once $O(\log n)$ seed length is achieved for $\mathsf{TIME}[\mathrm{poly}(n)]$, we can reduce the seed length all the way down to an arbitrarily small super-constant value; more generally, we can fool $\mathsf{TIME}[\mathrm{poly}(n)]/a(n)$ with essentially optimal seed length $O(\log a(n))$ for any time constructible $a(n) \leq \log n$. The discrepancy between the simplicity of the result here and the work required in the previous section is due to the key distinction that in this regime, the pseudorandom generator has enough resources to *simulate* the distinguishers it is trying to fool.

Theorem 9. *Let $a(n) \leq \log n$ be time constructible. Assuming E requires $2^{\Omega(n)}$-size Boolean circuits for sufficiently large n, there is a polynomial time computable generator $\mathcal{G}^a : \{0,1\}^{O(\log a(n))} \to \{0,1\}^n$ which fools $\mathsf{TIME}[n]/a(n)$ with error $\leq \frac{1}{3}$.*

Proof. Using the hardness assumption we obtain (uniformly in n) a pseudorandom generator with seed length $c\log n$ for some constant c. Setting $m = n^c$ and enumerating the outputs of our PRG, we obtain a list of n bit strings $x^1, \ldots, x^m \in \{0,1\}^n$ which is a pseudorandom generator for $\mathsf{TIME}[n]/n$. Let $r = 2^{a(n)}$, let $L_1, \ldots, L_r$ enumerate $\mathsf{TIME}[n]/a(n)$ machines. Define the matrix $M \in \{0,1\}^{m\times r}$ with $M(i,j) = L_j(x^i)$; we may compute M in $\mathrm{poly}(n)$ time.

We now deterministically construct a set $I \subseteq [m]$ of size $a(n)^{O(1)}$ so that for every j we have

$$|\Pr_{i\sim I}[M(i,j)=1] - \Pr_{i\sim[m]}[M(i,j)=1]| \leq \frac{1}{3}$$

If we can accomplish this, we output the condensed PRG whose range consists of the strings $\{x^i \mid i \in I\}$ and has seed length $\lceil \log |I| \rceil = O(\log a(n))$ and are finished. The algorithm to find the set I is given in the next lemma. □

Lemma 3. *There is a polynomial time algorithm which, given $M \in \{0,1\}^{m\times r}$ with $r \leq m^{O(1)}$, outputs a set $I \subseteq [m]$ of rows so that $|I| \leq O(\log r)$, and for every column $j \in [r]$, we have*

$$|\Pr_{i\sim I}[M(i,j)=1] - \Pr_{i\sim[m]}[M(i,j)=1]| \leq \frac{1}{3}$$

Proof. Using known results on the so-called "set balancing problem" [MSN94], for any $I \subseteq [m]$ we may efficiently compute $I' \subseteq I$, $|I'| \leq \frac{|I|}{2} + \sqrt{|I|\log r}$ so that

$$|\Pr_{i\sim I}[M(i,j)=1] - \Pr_{i\sim I'}[M(i,j)=1]| \leq \sqrt{\frac{\log r}{|I|}}$$

for every j. In particular, provided $\sqrt{|I|\log r} \leq \frac{1}{4}|I|$, i.e. $|I| \geq 16\log r$, we have that $|I'| \leq \frac{3}{4}|I|$. Initializing $I_0 = [m]$, we may apply the above to obtain $I_0 \supseteq I_1 \supseteq \cdots \supseteq I_q$, $|I_q| = \Theta(\log r)$, and for every j

$$|\Pr_{i\sim I_\ell}[M(i,j)=1] - \Pr_{i\sim I_{\ell+1}}[M(i,j)=1]| \leq \sqrt{\frac{\log r}{|I_\ell|}}$$

Hence

$$|\Pr_{i\sim[m]}[M(i,j)=1] - \Pr_{i\sim I_q}[M(i,j)=1]| \leq \sum_{\ell\leq q}\sqrt{\frac{\log r}{|I_\ell|}}$$

For a suitable choice of q we will have $\sqrt{\frac{\log r}{|I_q|}} = \epsilon$ for an arbitrarily small constant ϵ, and $\sqrt{\frac{\log r}{|I_\ell|}} \leq \sqrt{\frac{3}{4}}\cdot\sqrt{\frac{\log r}{|I_{\ell+1}|}}$, hence

$$|\Pr_{i\sim[m]}[M(i,j)=1] - \Pr_{i\sim I_q}[M(i,j)=1]| \leq \sum_{\ell\leq q}\sqrt{\frac{\log r}{|I_\ell|}} \leq \epsilon\sum_{\ell=0}^{\infty}\Big(\sqrt{\frac{3}{4}}\Big)^\ell < \frac{1}{3}$$

□

3.4 Optimality of the Seed Length

We include a simple (and basically standard) argument which indicates that the seed lengths obtained in the previous are essentially optimal with respect to the amount of advice they can handle:

Lemma 4. *Let $s(n) \leq \log n$ be time-constructible, and $G = (G_n : \{0,1\}^{s(n)} \to \{0,1\}^n)_{n\in\mathbb{N}}$ an arbitrary family of generators which fools $\mathsf{TIME}[n]/a(n)$ almost everywhere with error $\frac{1}{2}$. Then $s(n) \geq \log a(n)$.*

In particular, if G fools $\mathsf{TIME}[n]$ then $s(n) \geq \lambda(n)$ for some inverse-total computable $\lambda(n) = \omega(1)$.

Proof. Let $\ell(n) = s(n) + 1$, and consider the family $\mathcal{L}$ of languages L with $L_n(x)$ depending only on the first $\ell(n)$ bits of x, and $|L_n| = 2^{n-1}$. For each $n \in N$, we have that $|\text{range}(G_n)| \leq 2^{\ell(n)-1}$, hence there exists a language $L \in \mathcal{L}$ so that for all n sufficiently large, we have $\text{range}(G_n) \cap L_n = \emptyset$. On the other hand for every n we have $\Pr_{x \sim \{0,1\}^n}[x \in L] = \frac{1}{2}$, hence G_n fails to fool the language L_n for all n sufficiently large. Every language $L \in \mathcal{L}$ is decidable in deterministic time $n + 2^{\ell(n)}$ with $2^{\ell(n)}$ bits of advice (we only need that $s(n)$, and hence $\ell(n)$ are time constructible) so we get a contradiction if $2^{\ell(n)} \leq O(n)$ or $2^{\ell(n)} \leq a(n)$. □

In the case of deterministic time (Section 3.3) this implies that the stated construction is optimal for advice complexity $\leq \log n$. For our main construction fooling $\mathsf{TIME}^{\mathsf{NP}}$, we have no indication that to fool uniform languages (no advice) requires iterated-logarithmic seed length. However Theorem 5 is able to fool languages with advice complexity $\log^{[d]}(n)$ with seed length $O(\log^{[d+1]}(n))$, so in this sense we are able to achieve the maximum advice complexity for the given seed length $O(\log^{[d+1]}(n))$. It is consistent with our current knowledge that fooling completely uniform algorithms, even with an NP oracle, is achievable with arbitrarily slow-growing time constructible seed lengths, as we can achieve in the case of deterministic adversaries (under standard hardness assumptions).

4 Construction of Random Strings and Applications

In this section we discuss the applications of our main result to explicit constructions of random strings and circuit lower bounds.

4.1 Random String Construction Via PRG Concatenation

We start by applying our PRG constructions to give polynomial time explicit constructions of highly random strings. The first is a direct combination of our main PRG constructions with Observations 2 and 3 from the preliminaries. We start with the situation in which we use the strong form of our hardness assumptions, and hence Theorem 5 as our PRG:

Theorem 10. *Let $d \in \mathbb{N}$, $T(n) \leq (\exp^{[c]}(n))^{O(1)}$, and assume Hypothesis $\mathsf{SH}(\mathsf{NP}, d + c)$. Then for any constant $k \in \mathbb{N}$, there is a polynomial time algorithm $\mathcal{A}$ so that, for all n with $\mathsf{K}^{T(n)}(n) \leq \log^{[d]}(n)$, $\mathcal{A}(1^n) \in \{0,1\}^n$ is an n-bit string x with $\mathsf{K}^{T(n)}(x) \geq \Omega(\frac{n}{(\log^{[d]}(n))^{O(1)}})$.*

In the special case $d = 1$, $c = 0$, under a hardness assumption for singly exponential time bounds and NP oracles we obtain an algorithm $\mathcal{A}$ which produces, for every $n \in \mathbb{N}$, a string x of length n with $\mathsf{K}^{T(n)}(x) \geq \frac{n}{\text{poly}(\log n)}$ for all n.

More generally, a hardness assumption for exponential time bounds and $\mathcal{O}'$ oracles yields an efficient algorithm for computing strings with $\tilde{\Omega}(n)$ $\mathcal{O}$-oracle time-bounded Kolmogorov complexity, provided there is a $\text{poly}(T(n))$ time $\mathcal{O}'$ oracle algorithm which can test the $\mathsf{K}^{T(n),\mathcal{O}}$ complexity of an n-bit string.

Proof. This is a rather direct corollary of Theorem 5. Let n be given with $\mathsf{K}^{T(n)}(n) \leq \log^{[d]}(n)$, and let $s_d(\cdot) \leq O(\log^{[d+1]}(\cdot))$ be the seed length bound of the generator $\mathcal{G}^d$ from Theorem 5 using the base time bound $T(n)$. Let n' be the largest integer such that $n' 2^{s_d(n')} \leq n$; clearly $\mathsf{K}^{T(n)}(n') \leq \mathsf{K}^{T(n)}(n) + O(1) \leq \log^{[d]}(n)$, hence we may apply Theorem 5 to compute (uniformly) a pseudorandom generator $\mathcal{G}^d_n : \{0,1\}^{s_d(n')} \to \{0,1\}^{n'}$ which fools $\mathsf{TIME}^{\mathsf{NP}}[T(n)]/O(1)$ on input length n'. By Observation 3 we conclude that some string z in the range of $\mathcal{G}_n$ has $\mathsf{K}^{T(n)}(z) \geq n' - 1$, hence the concatenation of all strings in the range of $\mathcal{G}^d_n$ (padded at the end with 0s to

bring the length up to n) will have $\mathsf{K}^{T(n)}(\cdot)$ complexity at least $n' - s_d(n') \geq \frac{n}{(\log^{[d]}(n))^{O(1)}}$ (recall Observation 2).

The second part of the theorem is the special parameter setting given by Theorem 6, and for the last sentence in the statement, we use the fact that Theorem 6 works for any oracle $\mathcal{O}'$. □

If we instead rely on the weaker form hardness assumptions and the associated PRG from Theorem 7 we obtain via the exact same argument:

Theorem 11. *Assume Hypothesis* $\mathsf{WH}(\mathsf{NP}, d+1, v)$ *with* $d \geq 0$, $v \geq 2$. *Then for every* $k \in \mathbb{N}$, *there is a* $O^{\langle d+v \rangle}(n)$ *time algorithm* $\mathcal{A}$ *such that for all sufficiently large* n, $\mathcal{A}(1^n) \in \{0,1\}^n$ *is a string* x *satisfying* $\mathsf{K}^{n^k}(x) \geq \frac{n}{O^{\langle v \rangle}(\log^{[d]}(n))}$.

In particular, assuming that there is a language in $\mathsf{TIME}[2^{2^n}]$ *which cannot be decided in* $\mathsf{TIME}^{\mathsf{NP}}[2^{\epsilon 2^n}]/2^{\epsilon n}$ *infinitely often, there is a quasipolynomial time algorithm which outputs strings* $x \in \{0,1\}^n$ *with* $\mathsf{K}^{n^k}(x) \geq \frac{n}{\mathrm{poly}(\log n)}$.

More generally, a hardness assumption for exponential time bounds and $\mathcal{O}'$ *oracles yields an efficient algorithm for computing strings with* $\tilde{\Omega}(n)$ $\mathcal{O}$*-oracle time-bounded Kolmogorov complexity, provided there is a* $\mathrm{poly}(T(n))$ *time* $\mathcal{O}'$ *oracle algorithm which can test the* $\mathsf{K}^{T(n),\mathcal{O}}$ *complexity of an* n*-bit string.*

We now move on to a second class of constructions which achieves a far superior degree of randomness, but the cost is that our construction only works for infinitely many n, and unlike the previous we have no control over which specific values of n our construction succeeds on.

Theorem 12. *Assume Hypothesis* $\mathsf{SH}(\mathsf{NP}, d)$. *Then for any constant* $k \in \mathbb{N}$, *there is a polynomial time algorithm* $\mathcal{A}$ *so that, for infinitely many* n, $\mathcal{A}(1^n) \in \{0,1\}^n$ *is an* n*-bit string* x *with* $\mathsf{K}^{n^k}(x) \geq n - \mathrm{poly}(\log^{[d]} n)$.

Proof. Applying the assumption with Theorem 5 and the argument in the previous proof, we have an algorithm $\mathcal{A}$ which prints, for all sufficiently large n in $N_d := \{n \mid n = \exp^{[d+1]}(n') \text{ for some } n'\}$, a list of $\mathrm{poly}(\log^{[d]}(n))$ strings, one of which must have $\mathsf{K}^{n^k}(n) \geq n - 1$. We now define a second algorithm $\mathcal{A}'$, which, given n, computes the largest value $m \leq n$ with $m \in N$, and prints the $(n-m)^{th}$ element of the list $\mathcal{A}(1^m)$ (if $(n-m)$ is larger than the length of the list, it does something arbitrary). For every $m \in N$, there is some $n \leq m + \mathrm{poly}(\log^{[d]}(n))$ which prints a string x of length m with $\mathsf{K}^{n^k}(x) \geq m - 1 \geq n - \mathrm{poly}(\log^{[d]}(n))$. □

As before, a modified variant of this construction can be done in the case of the alternative weak form hardness assumptions, where the cost of using the weaker kind of assumption is that the run time of the construction will degrade correspondingly; we leave the details to the interested reader as we will not use this variant in what follows.

4.2 Explicit Constructions Under Plausible Hardness Assumptions

By our main result, under reasonable hardness assumptions we are able to obtain explicit constructions of strings with time-bounded Kolmogorov complexity $\frac{n}{\beta(n)}$, where $\beta(n)$ grows like an arbitrarily slow iterated logarithm function. We show here how to use this construction to build various important pseudorandom objects under our main hardness assumptions. We start with a list of fundamental explicit construction problems, focusing on those studied originally in [Kor21], in addition to so-called "incompressible functions" considered in [AASY15]. For background on substantial literature dedicated to these problems see [Kor21, AASY15],

Definition 7 (List of Explicit Construction Problems)**.**

- (s,r)-*Rigidity: A matrix* $M \in \{0,1\}^{n\times n}$ *such that* $M \neq S+R$ *whenever* $S,R \in \mathbb{F}_2^{n\times n}$*,* S *has at most* s *nonzero entries,* R *has rank at most* r
- s-*Hard Boolean Functions: A Boolean function* $f : \{0,1\}^n \to \{0,1\}$ *not computed by any Boolean circuit of size* $\leq s$
- (s,k)-*Incompressible Boolean Functions: A Boolean function* $f : \{0,1\}^n \to \{0,1\}$ *which cannot be expressed in the form* $f(x) = g(C(x))$ *where* $C : \{0,1\}^n \to \{0,1\}^k$ *is a circuit of size* s *and* $g : \{0,1\}^k \to \{0,1\}$ *is an arbitrary Boolean function*
- s-$\mathsf{PSPACE}^{\mathsf{cc}}$-*Complexity: A communication matrix* $M \in \{0,1\}^{2^n\times 2^n}$ *such that* M *cannot be solved by space-*s *communication protocols*
- (s,t)-*Bit-Probe Complexity: A data structure problem* $D \in \{0,1\}^{2^n\times 2^n}$ *which requires space* $\geq s$ *or time* $\geq t$ *in the bit-probe model.*
- *Ramsey: A graph* $G \in \binom{n}{2}$ *with no clique or independent set of size* $\geq 2.1 \cdot \log n$

We then have:

Lemma 5. *Say that a Range Avoid instance has "stretch* $n \mapsto m$*" if it is of the form* $C : \{0,1\}^n \to \{0,1\}^m$*. The following uniform reductions exist from the problems in Definition 7 to Range Avoidance:*

1. (s,r)-*Rigidity reduces uniformly to Avoid with stretch* $(2s\log n + 2nr) \mapsto n^2$
2. s-*Hard Boolean Function reduces uniformly to Avoid with stretch* $(1+o(1))s\log s \mapsto 2^n$
3. (s,k)-*Incompressible Boolean Functions reduces uniformly to Avoid with stretch* $2^k + (1+o(1))s\log s \mapsto 2^n$
4. s-$\mathsf{PSPACE}^{\mathsf{cc}}$-*Complexity reduces uniformly to Avoid with stretch* $2^{O(s)+n} \mapsto 2^{2n}$
5. (s,t)-*Bit-Probe Complexity reduces uniformly to Avoid with stretch* $s2^n + 2^{t+n} \mapsto 2^{2n}$
6. *Ramsey reduces uniformly to Avoid with stretch* $n - \Omega(\log n) \mapsto n$

Proof. All of these proofs occur in [Kor21], with the exception of Incompressible Functions. For this, if $f(x) = g(C(x))$ for a circuit $C : \{0,1\}^n \to \{0,1\}^k$ of size s and $g : \{0,1\}^k \to \{0,1\}$, we may represent the circuit C (gate by gate) using $(1+o(1))s\log s$ bits via a standard encoding, and the function g via its 2^k-bit truth table. From these it is clear how we may reproduce the function f efficiently. □

Combining this with our first generator construction from the previous section and Observation 1, we have:

Theorem 13. *Assume Hypothesis* $\mathsf{SH}(\mathsf{NP},d)$*. Then for all* n *with* $\mathsf{K}^{\mathrm{poly}(n)}(n) \leq \log^{[d]}(n)$*, we have the following:*

1. *Polynomial time construction of matrices in* $\mathbb{F}_2^{n\times n}$ *which are* $(\frac{n^2}{\log n\cdot \mathrm{poly}(\log^{[d]}(n))}, \frac{n}{\mathrm{poly}(\log^{[d]}(n))})$*-rigid. In particular, for any* $d > 2$ *such matrices achieve Valiant rigidity.*

2. *Boolean functions in* $\mathsf{E} = \mathsf{TIME}[2^{O(n)}]$ *with circuit complexity at least* $\frac{2^n}{n \cdot \mathrm{poly}(\log^{[d-1]}(n))}$

3. *Boolean functions in* $\mathsf{E} = \mathsf{TIME}[2^{O(n)}]$ *which are* $(\Omega(\frac{2^n}{n \cdot \mathrm{poly}(\log^{[d-1]}(n))}), n - O(\log^{[d]} n))$ *incompressible*

4. *Polynomial time construction of communication matrices* $M \in \{0,1\}^{2^n \times 2^n}$ *which require space* $\Omega(n)$ *(for any setting of* d*)*

5. *Polynomial time construction of data structure problems* $D \in \{0,1\}^{2^n \times 2^n}$ *which require space* $\Omega(2^n)$ *or time* $\Omega(n)$ *in the bit probe model (for any setting of* d*)*

In the important case $d = 1$*, each of the constructions works for all* n *and requires the hardness assumption only for singly-exponential time bounds.*

We arrive at similar conclusions if we use the weak form hypothesis in combination with Theorem 7 instead, at the cost of a slow-down in our construction algorithms. We highlight below the results obtained by using the special case of Theorem 7 given in Theorem 8:

Theorem 14. *Assume that there is some* $\epsilon > 0$ *and a language* $L \in \mathsf{TIME}[2^{2^n}]$ *which is not computable in* $\mathsf{TIME}^{\mathsf{NP}}[2^{\epsilon 2^n}]/2^{\epsilon n}$ *on more than finitely many input lengths (this is Hypothesis* $\mathsf{WH}(\mathsf{NP}, 1, 2)$*). Then we have the following for all* n*:*

1. *Quasipolynomial time construction of matrices in* $\mathbb{F}_2^{n \times n}$ *which are* $(\frac{n^2}{(\log n)^{O(1)}}, \frac{n}{(\log n)^{O(1)}})$*-rigid.*

2. *Boolean functions in* $\mathsf{EXP} = \mathsf{TIME}[2^{n^{O(1)}}]$ *with circuit complexity at least* $\frac{2^n}{\mathrm{poly}(n)}$

3. *Boolean functions in* EXP *which are* $(\Omega(\frac{2^n}{\mathrm{poly}(n)}), n - O(\log n))$ *incompressible*

4. *Quasipolynomial time construction of communication matrices* $M \in \{0,1\}^{2^n \times 2^n}$ *which require space* $\Omega(n)$ *(for any setting of* d*)*

5. *Quasipolynomial time construction of data structure problems* $D \in \{0,1\}^{2^n \times 2^n}$ *which require space* $\Omega(2^n)$ *or time* $\Omega(n)$ *in the bit probe model (for any setting of* d*)*

The above results cover the vast majority of explicit construction problems considered originally in [Kor21]. A notable exception is the construction of near-optimal Ramsey graphs; here the stretch $n - \Omega(\log n) \mapsto n$ given by Lemma 5 is to small to apply Theorem 10, and we must appeal instead to Theorem 12. Using this approach, we will be able to construct near-optimal Ramsey graphs infinitely often, provided we modify appropriately our encoding of graphs by strings to be well-defined on every input length:

Definition 8 (Ramsey Graphs of Every Length)**.** *We associate every every string* $x \in \{0,1\}^*$ *to a graph* G_x *as follows. Let* $n = |x|$ *and set* n' *to be the greatest integer so that* $\binom{n'}{2} \leq n$*. Set* $m = \binom{n'}{2}$*, truncate* x *to the first* m *bits and interpret it as a graph on* $\binom{n'}{2}$ *vertices canonically. Under this encoding, we say that* x *encodes a Ramsey graph if* G_x *contains no clique or independent set of size* $\geq 2.1 \cdot \log n'$*.*

We then have:

Lemma 6. *Assume Hypothesis* $\mathsf{SH}(\mathsf{NP}, 2)$*. Then there is a polynomial time algorithm which, for infinitely many* n*, outputs a string* $x \in \{0,1\}^n$ *so that* G_x *is Ramsey.*

Proof. We claim that we may uniformly construct a Range Avoidance instance $C : \{0,1\}^{n-\Omega(n)} \to \{0,1\}^n$ so that for all n, range(C_x) contains every x such that G_x fails to be Ramsey. In particular, given n we may compute efficiently the parameters n', m being the number of vertices and edges for the graphs $\{G_x \mid x \in \{0,1\}^n\}$; we then apply Lemma 5 to obtain (uniformly) a Range Avoidance instance with stretch $m - \Omega(\log m) \mapsto m$ covering all strings $z \in \{0,1\}^m$ which fail to encode Ramsey graphs. If $x \in \{0,1\}^n$ fails to be Ramsey, then $x = zy$ for some $z \in \{0,1\}^m$ which fails to be Ramsey, hence we may uniformly construct an Avoid instance covering every non-Ramsey string $x \in \{0,1\}^n$ with stretch $n - \Omega(\log n') \mapsto n$, which is $n - \Omega(\log n) \mapsto n$. We thus conclude that $\mathsf{K}^{\mathrm{poly}(n)}(x) \leq n - \Omega(\log n)$ whenever G_x fails to be Ramsey. Applying Theorem 12 and our hardness assumption yields the lemma. □

4.3 Hardness Condensation

We also apply Theorem 10 to derive a new *hardness condensation* result. Hardness condensation is a phenomenon where a mild hardness assumption can be transformed into a much stronger hardness assumption of the same type. It was first studied in [BS06], who showed hardness condensation results for complexity classes with advice. It is more interesting to get hardness condensation for uniform classes, and indeed [Kor21] achieves this for E^{NP} - he shows that E^{NP} requires exponential-size circuits iff it requires almost maximum-size circuits.

We obtain a new hardness condensation result for deterministic time *without* an NP-oracle. Here the condensation is with respect to non-uniform complexity - the stronger hardness assumption can handle almost a maximum non-trivial amount of non-uniformity, while the weaker hardness assumption only involves an exponential amount of non-uniformity.

Theorem 15. *The following ar equivalent:*

1. *For every d there exists v so that, for all time constructible $T(n) \leq \exp^{[d]}(n)$, we have $\mathsf{TIME}[T(n)]$ is not contained even infinitely often in $\mathsf{SPACE}[o^{\langle v \rangle}(T(n))]/o^{\langle v \rangle}(2^n)$*
2. *For every d there exists v so that, for all time constructible $T(n) = \exp^{[d]}(n)$, we have $\mathsf{TIME}[T(n)]$ is not contained even infinitely often in $\mathsf{SPACE}[o^{\langle v \rangle}(T(n))]/2^{n-\omega(\log n)}$*

Proof. Clearly the second item implies the first. We show that the first item implies the second. Assuming (1), we have that every every d there exists v so that $\mathsf{WH}(\mathsf{QBF}, d, v)$ holds. Applying Theorem 11, for every d there is v so that for any time constructible $T \leq \exp^{[d]}(n)$, there is an algorithm A running in time $O^{\langle v \rangle}(T(2^n))$ which, given 1^{2^n}, outputs a string of length 2^n which cannot be produced by any algorithm running in space $T(2^n)$ with $2^{n-\omega(\log n)}$ bits of advice [4]. Interpreting this string as a function $f : \{0,1\}^n \to \{0,1\}$, we obtain a language in $\mathsf{TIME}[O^{\langle v \rangle}(T(2^n))]$ which is not contained even infinitely often in $\mathsf{SPACE}[T(2^n)]/2^{n-\omega(\log n)}$. Reparameterizing the time bounds yields (2). □

4.4 Range Avoidance vs Uniform Range Avoidance

We use our results together with previous work to argue that the Range Avoidance problem becomes much more tractable for uniformly computable maps. While there is compelling complexity-theoretic evidence in various settings that Range Avoidance is intractable in general, our results show that Uniform Range Avoidance is tractable under plausible complexity-theoretic assumptions.

The first setting we consider is where the Range Avoidance instance is an arbitrary polynomial-size circuit $C : \{0,1\}^n \to \{0,1\}^m$, and we wish to solve Range Avoidance in polynomial time.

[4] We are using here the fact that $\mathsf{SPACE}[T(n)]$ can be simulated in $\mathsf{TIME}^{\mathsf{QBF}}[\mathrm{poly}(T(n))]$

Ilango, Li and Williams [ILW23] showed that in this setting, Range Avoidance is infeasible, under the standard assumption that $\mathsf{NP} \neq \mathsf{coNP}$ and the plausible assumption that subexponentially-secure indistinguishability obfuscation exists [JLS21].

Theorem 16. *[ILW23] If* $\mathsf{NP} \neq \mathsf{coNP}$ *and subexponentially-secure indistinguishability obfuscation exists, then for any* $c \geq 0$*, there is no polynomial-time algorithm which solves Range Avoidance for polynomial-size circuits mapping* n *to* n^c *bits.*

Corollary 1. *Assume that* $\mathsf{NP} \neq \mathsf{coNP}$*, subexponentially-secure indistinguishability obfuscation, and Hypothesis* $\mathsf{SH}(\mathsf{NP}, 1)$*. There is a constant* c *such that for all* $d \geq c$*, there is a polynomial-time algorithm which solves Range Avoidance on uniform circuits of size* n^d *mapping* n *bits to* n^d *bits but no polynomial-time algorithm which solves Range Avoidance on all circuits of size* n^d *mapping* n *bits to* n^d *bits.*

Proof. We show that we can take c to be $1 + \delta$, for arbitrarily small $\delta > 0$. Indeed, for such c and $d \geq c$, the intractability of general Range Avoidance follows from Theorem 16, under the given assumptions. We show that uniform Range Avoidance can be solved by applying Theorem 10 and using the third assumption. Indeed, by this assumption, for every constant k, there exists a polynomial-time algorithm A_k which for all n, on input 1^n outputs a length-n string of K^{n^k} complexity at least $n/\mathsf{polylog}(n)$.

Let $\{C_n\}$ be a sequence of uniform circuits mapping n bits to n^d, where d is any constant greater than 1. Here our notion of uniformity is standard $\mathsf{DLOGTIME}$-uniformity, but the argument can be adapted to work for more relaxed notions of uniformity. Note that any output $y = C_n(x)$ has K^{n^k} complexity at most $n + O(\log(n))$ for any $k > d$, as we can first generate C_n given n using the uniformity condition and then generate y from C_n and x by simulating C_n. By running A_k on input 1^{n^d}, we obtain a string of length n^d and $\mathsf{K}^{n^{kd}}$ complexity at least $n^d/\mathsf{polylog}(n)$, which is therefore a non-output of C_n when n is large enough. □

We also consider the setting where the Range Avoidance instance is an arbitrary polynomial-size TQBF-oracle circuit $C : \{0,1\}^n \to \{0,1\}^m$, and we wish to solve Range Avoidance in polynomial time. The negative evidence for this is even stronger - the task is intractable under the standard assumption $\mathsf{PSPACE} \neq \mathsf{NP}$. Somewhat surprisingly, we show that uniform Range Avoidance is tractable under plausible assumptions even though the uniform Range Avoidance instance C is allowed to use a TQBF-oracle.

Theorem 17. *[ABK24] Suppose* $\mathsf{PSPACE} \neq \mathsf{NP}$ *(resp.* $\mathsf{PSPACE} \not\subseteq \mathsf{NTIME}[2^{\log^{O(1)} n}]$*). There is a constant* c *such that for all* $d \geq c$*, there is no polynomial-time (resp. quasipolynomial time) algorithm which solves Range Avoidance on all* TQBF*-oracle circuits of size* n^d*.*

In fact, [ABK24] showed that computing an KS-incompressible string x of length n conditional on a given string y is hard for polynomial time if $\mathsf{PSPACE} \neq \mathsf{NP}$, where KS is Kolmogorov space-bounded complexity. This is easily seen to imply the result above by considering the Range Avoidance instance which takes a TQBF-oracle program as input and evaluates it. The extension to quasipolynomial time bounds is not stated in [ABK24] but follows immediately from the proof.

Corollary 2. *Assume* $\mathsf{SH}(\mathsf{PSPACE}, 1)$*, and that* $\mathsf{PSPACE} \not\subseteq \mathsf{NTIME}[2^{\log^{O(1)} n}]$*. Then there is a constant* c *such that for all* $d \geq c$*, there is a polynomial-time algorithm which solves Range Avoidance on uniform* TQBF*-oracle circuits of size* n^d *but no polynomial-time algorithm which solves Range Avoidance on all* TQBF*-oracle circuits of size* n^d*.*

Proof. For the negative result on Range Avoidance, we simply apply Theorem 17. For the positive result, we apply Theorem 10 (relativized to a TQBF oracle) and then use the same argument as in the proof of the previous corollary. □

5 Barriers to Improvements

In this section we present two barriers to improving our main results. First we show that the seed length $\log^{[O(1)]} n$ achieved in our PRG from Section 3 is in some sense optimal for hardness/randomness approaches which use the same overall structure as our argument: specifically, we show that our PRG construction corresponds to a construction of a special kind of seeded extractor with seed length $\log^{[O(1)]} n$, and prove unconditionally that this is the minimal achievable seed length in any such construction. Second we show that there is no relativizing argument that directly reduces the construction of $\mathsf{K}^{\mathrm{poly}(n)}$-random strings to the construction of hard truth tables.

5.1 Multi-Reconstructive Extractors for Block Sources

The classical approach to turning hardness into randomness [NW94, IW97] was famously shown by Trevisan [Tre01] to be essentially equivalent to the task of constructing an explicit seeded extractor. Roughly speaking, Trevisan showed that any method for producing a pseudorandom generator using a hard boolean function whose correctness is proven in a sufficiently *black box* fashion must actually produce a seeded extractor, which treats the hard function as a high min-entropy source and the seed of the PRG as the seed in the seeded extractor.

In Section 3 a PRG was constructed against *uniform* algorithms (or more generally, algorithms with extremely small advice), whose seed length was much smaller than $\log n$, using a different kind of hardness assumption applied across many different input lengths. In the section we cast such a hardness/randomness construction in terms of certain seeded *multi-source* extractors, where each source corresponds to a hardness assumption used at a different input length. Through this connection we are able to show that the iterated logarithmic ($\log^{[d]}(n)$, $d = O(1)$) seed length in our PRGs from Section 3 is necessary for any hardness/randomness tradeoff that uses the same general framework as ours, and in particular which uses "only $O(1)$ many" hardness assumptions.

We start by recalling the proof of correctness of the PRG from Section 3, first in the case where $O(\log \log n)$ seed length is achieved using a hardness assumption at two different input lengths. We first used a hard function whose truth table had size $\mathrm{poly}(n)$ in order to get a PRG with seed length $O(\log n)$; call this f_1. We then bounded the run time of this PRG by some $T(n) = \mathrm{poly}(n)$, and in the next step required a hard function f_2 of truth table size $\approx \log n$, which was hard for algorithms with running time $T(n)$; in particular f_2 needed to be hard even in the regime where the computation of f_1 is easy. For this reason, we may roughly interpret this as saying, f_2 is a function (of much smaller input length) which is hard *conditioned on* f_1. When we iterate the argument to reduce the seed length to $\log^{[d]}(n)$, we require a sequence of functions $f_1, \ldots, f_d$, of smaller and small input lengths, with f_j being hard for algorithms with enough running time to compute all of $f_1, \ldots, f_{j-1}$, i.e. which is hard conditioned on $f_1, \ldots, f_{j-1}$. This is naturally analogous to the following well-studied information-theoretic notion of a sequence of random variables, each with high min-entropy conditioned on all of the previous:

Definition 9 (Block Sources [CG85])**.** *A random variable $\bar{\mathcal{X}} = (\mathcal{X}_1, \ldots, \mathcal{X}_d)$ is said to be a block source with d blocks and entropy sequence $(k_1, \ldots, k_d)$, if for every $(x_1, \ldots, x_d) \in \mathrm{supp}(\bar{\mathcal{X}})$ and every $j \leq d$,*

$$H_\infty(\mathcal{X}_j \mid \mathcal{X}_1 = x_1, \ldots, \mathcal{X}_{j-1} = x_{j-1}) \geq k_j$$

In the case $j = 1$, this means $H_\infty(\mathcal{X}_1) \geq k_1$.

A (seeded) block source extractor is a function which can produce nearly uniform randomness from a block source and an independent uniform seed:

Definition 10 (Seeded Block Source Extractors)**.** *We say that a function* $E : \prod_{j \le d}\{0,1\}^{m_j} \times \{0,1\}^s \to \{0,1\}^n$ *is a block source extractor for entropy sequence* $(k_1, \ldots, k_d)$ *and error* ϵ *if, for every block source* $\bar{\mathcal{X}} = (\mathcal{X}_1, \ldots, \mathcal{X}_d)$ *supported on* $\prod_{j \le d}\{0,1\}^{m_j}$ *with entropy sequence* $(k_1, \ldots, k_d)$*, we have that* $E(\bar{\mathcal{X}}, \mathcal{U}_s)$ *is* ϵ*-close to* $\mathcal{U}_n$ *in TV distance, where* $\mathcal{U}_s$ *is the uniform distribution on* $\{0,1\}^s$ *(independent of the block source) and* $\mathcal{U}_n$ *is uniform on* $\{0,1\}^n$*. The parameter* s *is called the seed length of the extractor.*

We will now show that our PRG construction can be viewed as giving a specific kind of explicit *reconstructive* extractor for block sources (with a seed), in the same sense that the standard hardness randomness constructions can be viewed as reconstructive single-source extractors via Trevisan's connection. We define such reconstructive seeded block source extractors as follows:

Definition 11 (Multi-Reconstructive Extractors)**.** *Let* $E : \prod_{j \le d}\{0,1\}^{m_j} \times \{0,1\}^s \to \{0,1\}^n$*. We say that* E *is a multi-reconstructive extractor for entropy sequence* $(k_1, \ldots, k_d)$ *and error* ϵ *if there are* $\mathrm{poly}(m_1, \ldots, m_d)$*-time algorithms* $R_1, \ldots, R_d$*, each taking oracle access to some* $D : \{0,1\}^n \to \{0,1\}$*, with the following behavior*

1. *Each* R_j *takes strings* $x_1, \ldots, x_{j-1} \in \{0,1\}^{m_1}, \ldots, \{0,1\}^{m_{j-1}}$ *and an advice string* $a_j \in \{0,1\}^{k_j}$ *and outputs a string in* $\{0,1\}^{m_j}$ *(in the case* $j = 1$ *the only input to* R_1 *is* $a_1 \in \{0,1\}^{k_1}$*).*
2. *For every* $\bar{x} \in \prod_{j \le d}\{0,1\}^{m_j}$*, if*

$$\left| \Pr_{z \sim \{0,1\}^s}[D(E(\bar{x}, z)) = 1] - \Pr_{y \sim \{0,1\}^n}[D(y) = 1] \right| \ge \epsilon$$

then there exists $j \le d$ *and some* $a_j \in \{0,1\}^{k_j}$ *such that* $R_j^D(x_1, \ldots, x_{j-1}, a_j) = x_j$

Note that in the case $d = 1$, we obtain the standard definition of reconstructive single source extractors. As shown in the single-source case by Trevisan [Tre01], we may observe that any multi-reconstructive extractor is automatically a block source extractor with similar parameters:

Lemma 7. *If* $E : \prod_{j \le d}\{0,1\}^{m_j} \times \{0,1\}^s \to \{0,1\}^n$ *is a muti-reconstructive extractor for entropy sequence* $(k_1, \ldots, k_d)$ *and error* ϵ*, then it is a block source extractor for entropy sequence* $(2k_1, \ldots, 2k_d)$ *and error* $\epsilon' = \epsilon + \sum_{j \le d} 2^{-k_j}$*.*

Proof. Let $\bar{\mathcal{X}}$ be a block source with entropy sequence $(2k_1, \ldots, 2k_d)$. The deviation of $E(\bar{\mathcal{X}}, \mathcal{U}_s)$ from uniform is bounded by $\epsilon + \delta$, where δ is the maximum over all distinguishers D of the probability that any of the d reconstruction procedures associated with E succeed on a random sample $(x_1, \ldots, x_d) \sim \bar{\mathcal{X}}$ from the source. We bound the probability for each reconstruction procedure separately and take a union bound over $j \le d$. For a fixed D and any fixing of $\mathcal{X}_1 = x_1, \ldots, \mathcal{X}_{j-1} = x_{j-1}$, there are at most 2^{k_j} values in $\{0,1\}^{m_j}$ that $R_j^D(x_1, \ldots, x_{j-1}, a_j)$ can output as we range over a_j; if $H_\infty(\mathcal{X}_j \mid \mathcal{X}_1 = x_1, \ldots, \mathcal{X}_{j-1} = x_{j-1}) \ge 2k_j$ then the probability that $\mathcal{X}_j$ lies in this set is at most 2^{-k_j}. □

We can now recast the central step in our main construction in Theorem 5 in the language of reconstructive extractors:

Theorem 18 (Essentially in [NZ96])**.** *Let* $E^1, \ldots, E^d$ *be single source extractors* $E^j : \{0,1\}^{m_j} \times \{0,1\}^{s_j} \to \{0,1\}^{n_j}$*, so that* E^j *is a reconstructive single-source extractor for min entropy* k_j *and error* ϵ_j *and is computable in polynomial time. Say that for each* $j < d$*, we have* $s_j = n_{j+1}$*. Define* $\tilde{E}^j : \prod_{j' \le j}\{0,1\}^{m_{j'}} \times \{0,1\}^{s_j} \to \{0,1\}^{n_1}$ *by induction on* j*, with* $\tilde{E}^1 = E^1$*, and* $\tilde{E}^{j+1}(x_1, \ldots, x_{j+1}, z) = \tilde{E}^{j-1}(x_1, \ldots, x_{j-1}, E^j(x_j, z))$*. Then* $\tilde{E}^d$ *is a multi-reconstructive extractor for entropy sequence* $(k_1, \ldots, k_d)$ *and error* $\sum_{j \le d} \epsilon_d$*.*

In [NZ96], the same method of composing single source extractors to form a seeded block source extractor is analyzed in the information theoretic setting (rather than the computational/reconstructive setting). The proof of this lemma in the reconstructive framework is essentially identical to the core argument justifying the correctness of our pseudorandom generator construction in Theorem 5:

Proof. We prove by induction on j that $\tilde{E}^j$ is a multi-reconstructive extractor with error $\delta_j := \sum_{j' \leq j} \epsilon_j$. By definition it is true for $\tilde{E}^1 = E^1$. Now, say that $\tilde{E}^j$ is a reconstructive extractor for entropy sequence $(k_1, \ldots, k_j)$ and error δ_j. So there are reconstruction procedures $R_1, \ldots, R_j$ taking oracle access to a function $D : \{0,1\}^{n_1} \to \{0,1\}$ so that, given any $D \subseteq \{0,1\}^n$ distinguishing $\tilde{E}^j(\bar{x}, z)$ with error δ, there exists $j' \leq j$ so that $R^D_{j'}(x_, \ldots, x_{j'-1}, a)$ produces $x_{j'}$ for some $a \in \{0,1\}^{k_{j'}}$. We then consider $\tilde{E}^{j+1}(x_1, \ldots, x_{j+1}, z')$; our reconstruction procedures for $R_1, \ldots, R_j$ will be as they were for $\tilde{E}^{j+1}$, and for R_{j+1} we use the construction procedure for the single source extractor E^{j+1}. Let D be given. For any $x_1, \ldots, x_{j+1}$ such that the first $R_1, \ldots, R_j$ reconstruction procedures all fail to reconstruct a symbol of $x_1, \ldots, x_{j+1}$ from its prefix using D, we know that $\tilde{E}^j(x_1, \ldots, x_{j+1}, \mathcal{U}_{s_j})$ fools D with error δ_j. Now define the test $D' \subseteq \{0,1\}^{n_{j+1}}$ (recall $n_{j+1} = s_j$), with $D'(z) = D(\tilde{E}^j(x_1, \ldots, x_j, z)) = 1$. Then, for a uniformly random $z \in \{0,1\}^{s_j}$ with have that $D'(z) = 1$ with probability in the range $\Pr_{y \sim n_{j+1}}[D(y) = 1] \pm \delta_j$. Observe that D' is efficiently computable with oracle access to D and the values $x_1, \ldots, x_j$, since each of the $E^{j'}$ extractors are efficiently computable. Thus, using the reconstruction procedure for E^{j+1}, either R_{j+1} succeeds in reconstructing x_{j+1} using oracle access to D, the previous inputs $x_1, \ldots, x_j$, and k_{j+1} bits of advice, or else we must have that $E^{j+1}(x_{j+1}, \mathcal{U}_{s_{j+1}})$ fools D' with error ϵ_{j+1}, and hence the overall construction fools D with error $\delta_j + \epsilon_{j+1}$, completing the proof. □

Using the above in combination with explicit families of single-source reconstructive extractors (e.g. Trevisan's extractor [Tre01]) we observe:

Corollary 3. *For any n and fixed constants $d \in \mathbb{N}$, $\gamma > 0$, there is an explicit multi-reconstructive extractor $E : \prod_{j \leq d} \{0,1\}^{m_j} \times \{0,1\}^s \to \{0,1\}^n$ for entropy sequence $(k_1, \ldots, k_d)$ and error ϵ, where:*

$$\sum_{j \leq d} m_j \leq \text{poly}(n), \quad k_j = m_j^\gamma, \quad s \leq O(\log^{[d]} n), \quad \epsilon \leq (\log^{[d-1]} n)^{-1}$$

Moreover, $k_j \geq \log^{[j]} n$ for each j, and hence by Lemma 7 this construction is also a block source extractor with the same parameters stated above.

We now prove that the seed length $\log^{[d]} n$ achieved above is optimal as a function of the output length n, regardless of how we choose the source lengths $m_1, \ldots, m_d$:

Theorem 19. *Say $E : \prod_{j \leq d} \{0,1\}^{m_j} \times \{0,1\}^s \to \{0,1\}^n$ is a block source extractor for min entropy sequence $(k_1, \ldots, k_d)$, $k_j \leq \frac{1}{2} m_j$, and error $\frac{1}{2}$. Then $s \geq \Omega(\log^{[d]} n)$.*

To prove this we rely on the following lemma:

Lemma 8. *Let $A \subseteq \prod_{j \leq d} \{0,1\}^{m_j}$ with $\log |A| \geq (\sum_{j \leq d} m_j) - q$. Then there is a block source with entropy sequence $(k_1, \ldots, k_d)$ supported on A, with $k_j \geq m_j - q - d$.*

Proof of Lemma 8. We prove by induction on d; the case $d = 1$ is trivial. Let A be given; for each $x_1 \in \{0,1\}^{m_1}$ define $A_{x_1} = \{(x_2, \ldots x_d) \mid (x_1, x_2, \ldots, x_d) \in A\}$, $\deg(x_1) = |A_{x_1}|$. So $\sum_{x_1 \in \{0,1\}^{m_1}} \deg(x) = |A|$. Define the set $V \subseteq \{0,1\}^{m_1}$ by

$$V = \{x_1 \mid \deg(x_1) \leq \exp((\sum_{j>1} m_j) - q - 1)\} \subseteq \{0,1\}^{m_1}$$

Let $U = \{0,1\}^{m_1} \setminus V$; if $|U| < \exp(m_1 - q - 1)$ then we must have

$$|A| \leq |V| \cdot \max_{x_1 \in V} \deg(x_1) + |U| \cdot \max_{x_1 \in U} \deg(x_1) \tag{4}$$

$$< 2^{m_1} \cdot \exp((\sum_{j>1} m_j) - q - 1) + \exp(m_1 - q - 1) \cdot \exp(\sum_{j>1} m_j) \tag{5}$$

$$= \exp((\sum_{j \leq d} m_j) - q - 1) + \exp((\sum_{j \leq d} m_j) - q - 1) \leq \exp((\sum_{j \leq d} m_j) - q) \tag{6}$$

and we get a contradiction. Hence, we know that $|U| \geq \exp(m_1 - q - 1)$. For every $x_1 \in U$, we also know that $A_{x_1} \subseteq \prod_{j>1}\{0,1\}^{m_j}$ has size at least $\exp(\sum_j m_j - q - 1)$, so by induction there is a block source $\mathcal{X}_{x_1}$ with min entropies $(k_2, \ldots, k_d)$, $k_j \geq m_j - (q+1) - (d-1) \geq m_j - q - d$ supported on A_{x_1}. We then construct our overall distribution $\mathcal{X}$ as follows: sample $x_1 \sim U$ uniformly, then sample $(x_2, \ldots, x_d) \sim \mathcal{X}_{x_1}$ and output $(x_1, \ldots, x_d)$. We know that the first coordinate of this distribution has min entropy $k_1 := \log |U| \geq m_1 - q - 1$, and that the remaining entries are a block source with min entropy sequence $(k_2, \ldots, k_d)$ conditioned on any fixing of the first coordinate; so overall $\mathcal{X}$ is a block source with entropy sequence $(k_1, \ldots, k_d)$. □

The proof of Theorem 19 combines the above lemma with the argument of [NZ96] used to show an $\Omega(\log n)$ lower bound on the seed length of single source extractors:

Proof of Theorem 19. When $d = 0$, E is of the form $E : \{0,1\}^s \to \{0,1\}^n$, so that $E(\mathcal{U}_s)$ is $\frac{1}{2}$ close to uniform on $\{0,1\}^n$; in this case we clearly require $s \geq n - 1$. For $d > 0$, we will show how to choose some $j \leq d$ and construct from E a second extractor $E' : \prod_{j \neq j'}\{0,1\}^{m_j} \times \{0,1\}^{s'} \to \{0,1\}^n$ using only $d - 1$ sources, so that E' is a block source extractor for entropy sequence $(k_1, \ldots, k_{j'-1}, k_{j'+1}, \ldots, k_d)$, the same error $(\frac{1}{2})$, and at most exponentially larger seed length $s' \leq s + 2^{s+2} + 2d$. This will yield the theorem by induction.

We will set j' to be any index such that $m_{j'} \leq 2^{s+2} + 2d$; if we can find such an index, we construct E' by simply moving input j' of E into the seed, i.e.

$$E'(x_1, \ldots, x_{j'-1}, x_{j'+1}, x_d, (x_{j'}, z)) = E(x_1, \ldots, x_{j'-1}, x_{j'}, x_{j'+1}, \ldots, x_d, z)$$

Clearly for any choice of j', E' remains a block source extractor for the contracted entropy sequence $(k_1, \ldots, k_{j'-1}, k_{j'+1}, \ldots, k_d)$ and the same error as E. To argue that $m_j \leq 2^{s+2} + 2d$ for some j we rely on Lemma 8 above. Let $\mathcal{D} \subseteq \{0,1\}^{\{0,1\}^n}$ consist of all functions $D : \{0,1\}^n \to \{0,1\}$ which depend only on the first $s + 1$ bits. So $|\mathcal{D}| \leq 2^{2^{s+1}}$. For every $\bar{x} \in \prod_{j \leq d}\{0,1\}^{m_j}$, there is some $D_{\bar{x}} \in \mathcal{D}$ such that

$$|\Pr_{z \sim \mathcal{U}_s}[D(E(\bar{x}, z)) = 1] - \Pr_{y \sim \mathcal{U}_n}[D(y) = 1]| \geq \frac{1}{2}$$

hence there exists a fixed D, so that for the set $\bar{X}_D = \{\bar{x} \mid D_{\bar{x}} = D\}$, we have $\log |\bar{X}_D| \geq (\sum_{j \leq d} m_j) - 2^{s+1}$. We then apply Lemma 8 with $q = 2^{s+1}$ to conclude that there is a block source $(\mathcal{X}_1, \ldots, \mathcal{X}_d)$ supported on $\bar{X}_D$ with min entropy sequence $(k'_1, \ldots, k'_d)$, $k'_j := m_j - 2^{s+1} - d$. If $k'_j \geq k_j$ for all j this will contradict the correctness of the extractor E, since $E(\bar{\mathcal{X}}, \mathcal{U}_s)$ will be distinguished with error $\frac{1}{2}$ by the test D. Hence there exists j such that $k'_j < k_j$, i.e. $m_j - k_j < 2^{s+1} + d$. Recalling that $k_j \leq \frac{1}{2} m_j$, we get $m_j < 2^{s+2} + 2d$ which concludes the proof. □

Note that the proof yields more than just a lower bound on s:

Observation 4 (Follows from Proof of Theorem 19)**.** *In any construction achieving the optimal seed length $s = O(\log^{[d]} n)$, there must be some smallest source length m_{j_1} with $m_{j_1} = \Omega(\log^{[d-1]} n)$, then a second smallest source length of $\Omega(\log^{[d-2]} n)$, and so on with the longest source having length on the order $\Omega(n)$*

Hence our construction uses essentially the only possible sequence of source lengths $(m_1, \ldots, m_d)$ which can be made to achieve a an optimal seed length of $\log^{[d]} n$.

5.2 No Relativizing Reduction from K^{poly} Randomness to Hard Truth Tables

In [Kor21] it is shown that if we are permitted the use of an NP oracle in our explicit construction algorithms we have the following appealing equivalence:

Theorem 20. *The following are equivalent:*

1. *There is a polynomial time NP-oracle algorithm constructing strings $x \in \{0,1\}^n$ with $\mathsf{K}^{\text{poly}(n)}(x) \geq n - 1$*
2. *There is a polynomial time NP-oracle algorithm constructing truth tables $f : \{0,1\}^n \to \{0,1\}$ with hardness $2^{\Omega(n)}$ (i.e. $\text{poly}(2^n) = 2^{O(n)}$).*

If this kind of equivalence could be shown without the aid of an NP oracle, it would supersede all of the main results in this paper. The proof of Theorem 20 is relativizing, and more specifically it gives a black-box reduction from the problem of constructing high Kolmogorov-complexity strings to constructing hard truth tables (the nontrivial direction). We give some indication here that an NP oracle is necessary for such an argument to work.

Observe that if $f : \{0,1\}^n \to \{0,1\}$ has circuit complexity $o(2^n/n)$, then for $N = 2^n$ and interpreting f as an N-bit string, we have $\mathsf{K}^{N^2}(f) \leq N/2$; this is because evaluating a circuit on an input takes at most $2^n = N$ time, and hence reconstructing f from a circuit computing it takes at most quadratic time. Hence if there were an analogue to Theorem 20 in the polynomial time regime, it would give, in particular, a reduction from constructing strings of large $\mathsf{K}^{n^c}(\cdot)$ to strings of large $\mathsf{K}^{n^2}(\cdot)$ complexity.

We show here that there is no such reduction which is "black box" in the same sense as Theorem 20. We first need to define the suitable notion of a black box reduction:

Definition 12. *Let c, d be fixed constants and n large. A black box reduction from $\mathsf{K}^{n^c}(\cdot)$-construction to $\mathsf{K}^{n^2}(\cdot)$-construction is an algorithm, given access to some oracle $\mathcal{O} : \{0,1\}^* \to \{0,1\}$, which has the following behavior:*

1. *There is a procedure $A_n^{\mathcal{O}}$ which, given strings $y_1, \ldots, y_{\text{poly}(n)}$ with $y_m \in \{0,1\}^m$, makes $\text{poly}(m)$ additional queries to $\mathcal{O}$ and outputs a string $x \in \{0,1\}^n$*
2. *For any oracle $\mathcal{O}$, if $y_1, \ldots, y_{\text{poly}(n)}$ are strings such that $\mathsf{K}^{m^d,\mathcal{O}}(y_m) \geq \frac{m}{2}$ for each m, $A^{\mathcal{O}}(y_1, \ldots, y_{\text{poly}(n)})$ outputs a string $x \in \{0,1\}^n$ with $\mathsf{K}^{n^c,\mathcal{O}}(x) \geq n^{\Omega(1)}$.*

Theorem 21. *For $c > d$ fixed constants, $c - d > 1$, there is no black box reduction from d-Avoid to c-Avoid.*

Proof. Consider the Range Avoidance instance $C^{\mathcal{O}} : \{0,1\}^{n^\epsilon} \to \{0,1\}^n$, defined by $C^{\mathcal{O}}(z) = (\mathcal{O}((z, 1^{n^{c-1}}, 0)), \ldots, \mathcal{O}((z, 1^{n^{c-1}}, 0)))$ where $(\cdot,\cdot,\cdot)$ is some standard pairing function. Clearly for any oracle $\mathcal{O}$ and any x such that $x \in \text{range}(C^{\mathcal{O}})$ we have $\mathsf{K}^{n^c,\mathcal{O}}(x) \leq n^\epsilon$. Hence if we can find an oracle $\mathcal{O}$ and supply strings $y_1, \ldots, y_{\text{poly}(n)}$ so that $\mathsf{K}^{m^d,\mathcal{O}}(y_m) \geq \frac{m}{2}$ for each m, but $A^{\mathcal{O}}(y_1, \ldots, y_{\text{poly}(n)})$

outputs a string in range($C^{\mathcal{O}}$) then we are done. Initially we fix the value of the oracle $\mathcal{O}$ to be zero on all strings of length at most n^{c-1}. We then conclude that for any $m \leq 4n$, we may determine a string y_m so that $\mathsf{K}^{m^d,\mathcal{O}}(y_m) \geq \frac{m}{2}$ is already forced by the current information about the oracle; this holds since a machine running in time $m^d \leq O(n^d) < n^{c-1}$ cannot access the any unfixed part of the oracle. We will fix these strings $y_1, \ldots, y_{4n}$ for the remainder of the proof.

Now, consider the algorithm $\mathcal{A}$ with the first $4n$ arguments fixed, as a function of the remaining arguments $y_{4n+1}, \ldots, y_{\text{poly}(n)}$. By the correctness of $\mathcal{A}$, for any extension of the partially defined oracle $\mathcal{O}$ and any valid solutions for these remaining arguments to $\mathcal{A}$, $\mathcal{A}$ will find a solution to the Range Avoidance instance $C^{\mathcal{O}}$. On the other hand, observe that if $y_{4n+1}, \ldots, y_{\text{poly}(n)}$ are chosen uniformly at random, then for *any oracle* $\mathcal{O}$ the probability that any y_m fails to have $\mathsf{K}^{m^d,\mathcal{O}}(y_m) \geq \frac{m}{2}$ is bounded by $\text{poly}(n)2^{-\frac{m}{2}} \leq 2^{-2n+o(n)}$ since for each such m we have $m \geq 4n$. Hence we obtain a randomized query procedure, making $\text{poly}(n)$ queries to the oracle $\mathcal{O}$, which outputs a solution to Avoid on $C^{\mathcal{O}}$ with failure probability $\leq 2^{-2n+o(n)}$. Since we have not fixed the behavior of $\mathcal{O}$ above input length n^{c-1}, $C^{\mathcal{O}}$ can take on any value $\{0,1\}^{n^\epsilon} \to \{0,1\}^n$, and is thus an arbitrary oracle-presented Range Avoidance instance. It then remains only to show that a randomized query algorithm making $\text{poly}(n)$ queries to an Avoid instance with stretch $n^\epsilon \mapsto n$ cannot succeed with probability as high as $1 - 2^{-2n+o(n)}$.

To obtain this randomized query lower bound, we appeal to Yao's lemma, and consider the best success probability of a deterministic $\text{poly}(n)$-query algorithm $\mathcal{Q}$ on a uniformly random instance $C : \{0,1\}^{n^\epsilon} \to \{0,1\}^n$. For each possible sequence of $\text{poly}(n)$ queries we argue the probability the supplied answer is incorrect conditioned on this query sequence ocurring is is at least 2^{-n}. This holds trivially, since $\text{poly}(n) < 2^{n^\epsilon}$, and hence a random C extending a fixed sequence of $\text{poly}(n)$ mappings $C(x_1) = y_1, \ldots,$ has probability at least 2^{-n} of hitting any fixed string in $\{0,1\}^n$. □

Acknowledgment

The authors would like to thank Hanlin Ren for many useful discussions.

References

[AASY15] Benny Applebaum, Sergei Artemenko, Ronen Shaltiel, and Guang Yang. Incompressible functions, relative-error extractors, and the power of nondeterministic reductions (extended abstract). In *Proceedings of the 30th Conference on Computational Complexity*, CCC '15, page 582–600, Dagstuhl, DEU, 2015. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[ABK24] Scott Aaronson, Harry Buhrman, and William Kretschmer. A qubit, a coin, and an advice string walk into a relational problem. In Venkatesan Guruswami, editor, *15th Innovations in Theoretical Computer Science Conference, ITCS 2024, January 30 to February 2, 2024, Berkeley, CA, USA*, volume 287 of *LIPIcs*, pages 1:1–1:24. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2024. URL: `https://doi.org/10.4230/LIPIcs.ITCS.2024.1`, `doi:10.4230/LIPICS.ITCS.2024.1`.

[All17] Eric Allender. The complexity of complexity. In Adam R. Day, Michael R. Fellows, Noam Greenberg, Bakhadyr Khoussainov, Alexander G. Melnikov, and Frances A. Rosamond, editors, *Computability and Complexity - Essays Dedicated to Rodney G. Downey on the Occasion of His 60th Birthday*, volume 10010 of *Lecture Notes in Computer Science*, pages 79–94. Springer, 2017. `doi:10.1007/978-3-319-50062-1\_6`.

[BS06] Joshua Buresh-Oppenheim and Rahul Santhanam. Making hard problems harder. In *21st Annual IEEE Conference on Computational Complexity (CCC 2006), 16-20 July 2006, Prague, Czech Republic*, pages 73–87. IEEE Computer Society, 2006. doi:10.1109/CCC.2006.26.

[CG85] Benny Chor and Oded Goldreich. Unbiased bits from sources of weak randomness and probabilistic communication complexity. In *26th Annual Symposium on Foundations of Computer Science (sfcs 1985)*, pages 429–442, 1985. doi:10.1109/SFCS.1985.62.

[CGL+23] Eldon Chung, Alexander Golovnev, Zeyong Li, Maciej Obremski, Sidhant Saraogi, and Noah Stephens-Davidowitz. On the randomized complexity of range avoidance, with applications to cryptography and metacomplexity. *Electron. Colloquium Comput. Complex.*, TR23-193, 2023. URL: https://eccc.weizmann.ac.il/report/2023/193, arXiv:TR23-193.

[CHLR23] Yeyuan Chen, Yizhi Huang, Jiatu Li, and Hanlin Ren. Range avoidance, remote point, and hard partial truth table via satisfying-pairs algorithms. In Barna Saha and Rocco A. Servedio, editors, *Proceedings of the 55th Annual ACM Symposium on Theory of Computing, STOC 2023, Orlando, FL, USA, June 20-23, 2023*, pages 1058–1066. ACM, 2023. doi:10.1145/3564246.3585147.

[CHR24] Lijie Chen, Shuichi Hirahara, and Hanlin Ren. Symmetric exponential time requires near-maximum circuit size. In Bojan Mohar, Igor Shinkar, and Ryan O'Donnell, editors, *Proceedings of the 56th Annual ACM Symposium on Theory of Computing, STOC 2024, Vancouver, BC, Canada, June 24-28, 2024*, pages 1990–1999. ACM, 2024. doi:10.1145/3618260.3649624.

[CL24] Yilei Chen and Jiatu Li. Hardness of range avoidance and remote point for restricted circuits via cryptography. In Bojan Mohar, Igor Shinkar, and Ryan O'Donnell, editors, *Proceedings of the 56th Annual ACM Symposium on Theory of Computing, STOC 2024, Vancouver, BC, Canada, June 24-28, 2024*, pages 620–629. ACM, 2024. doi:10.1145/3618260.3649602.

[CT21] Lijie Chen and Roei Tell. Simple and fast derandomization from very hard functions: eliminating randomness at almost no cost. In *STOC*, pages 283–291. ACM, 2021. doi:10.1145/3406325.3451059.

[GGNS23] Karthik Gajulapalli, Alexander Golovnev, Satyajeet Nagargoje, and Sidhant Saraogi. Range avoidance for constant depth circuits: Hardness and algorithms. In Nicole Megow and Adam D. Smith, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2023, September 11-13, 2023, Atlanta, Georgia, USA*, volume 275 of *LIPIcs*, pages 65:1–65:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023. URL: https://doi.org/10.4230/LIPIcs.APPROX/RANDOM.2023.65, doi:10.4230/LIPICS.APPROX/RANDOM.2023.65.

[GLW22] Venkatesan Guruswami, Xin Lyu, and Xiuhan Wang. Range avoidance for low-depth circuits and connections to pseudorandomness. In Amit Chakrabarti and Chaitanya Swamy, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2022, September 19-21, 2022, University of Illinois, Urbana-Champaign, USA (Virtual Conference)*, volume 245 of *LIPIcs*, pages 20:1–20:21. Schloss Dagstuhl - Leibniz-Zentrum für In-

formatik, 2022. URL: https://doi.org/10.4230/LIPIcs.APPROX/RANDOM.2022.20, doi:10.4230/LIPICS.APPROX/RANDOM.2022.20.

[Hir22] Shuichi Hirahara. Meta-computational average-case complexity: A new paradigm toward excluding heuristica. *Bull. EATCS*, 136, 2022. URL: http://bulletin.eatcs.org/index.php/beatcs/article/view/688.

[ILW23] Rahul Ilango, Jiatu Li, and R. Ryan Williams. Indistinguishability obfuscation, range avoidance, and bounded arithmetic. In Barna Saha and Rocco A. Servedio, editors, *Proceedings of the 55th Annual ACM Symposium on Theory of Computing, STOC 2023, Orlando, FL, USA, June 20-23, 2023*, pages 1076–1089. ACM, 2023. doi:10.1145/3564246.3585187.

[IW97] Russell Impagliazzo and Avi Wigderson. P = bpp if e requires exponential circuits: derandomizing the xor lemma. In *Proceedings of the Twenty-Ninth Annual ACM Symposium on Theory of Computing*, STOC '97, page 220–229, New York, NY, USA, 1997. Association for Computing Machinery. doi:10.1145/258533.258590.

[IW98] Russell Impagliazzo and Avi Wigderson. Randomness vs. time: De-randomization under a uniform assumption. In *39th Annual Symposium on Foundations of Computer Science, FOCS '98, November 8-11, 1998, Palo Alto, California, USA*, pages 734–743. IEEE Computer Society, 1998. doi:10.1109/SFCS.1998.743524.

[JLS21] Aayush Jain, Huijia Lin, and Amit Sahai. Indistinguishability obfuscation from well-founded assumptions. In Samir Khuller and Virginia Vassilevska Williams, editors, *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing, Virtual Event, Italy, June 21-25, 2021*, pages 60–73. ACM, 2021. doi:10.1145/3406325.3451093.

[KKMP21] Robert Kleinberg, Oliver Korten, Daniel Mitropolsky, and Christos H. Papadimitriou. Total functions in the polynomial hierarchy. In James R. Lee, editor, *12th Innovations in Theoretical Computer Science Conference, ITCS 2021, January 6-8, 2021, Virtual Conference*, volume 185 of *LIPIcs*, pages 44:1–44:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021. URL: https://doi.org/10.4230/LIPIcs.ITCS.2021.44, doi:10.4230/LIPICS.ITCS.2021.44.

[Kor21] Oliver Korten. The hardest explicit construction. In *62nd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2021, Denver, CO, USA, February 7-10, 2022*, pages 433–444. IEEE, 2021. doi:10.1109/FOCS52979.2021.00051.

[KP24] Oliver Korten and Toniann Pitassi. Strong vs. weak range avoidance and the linear ordering principle. In *65th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2024, Chicago, IL, USA, October 27-30, 2024*, pages 1388–1407. IEEE, 2024. doi:10.1109/FOCS61266.2024.00089.

[KvM99] Adam R. Klivans and Dieter van Melkebeek. Graph nonisomorphism has subexponential size proofs unless the polynomial-time hierarchy collapses. In *Proceedings of the Thirty-First Annual ACM Symposium on Theory of Computing*, STOC '99, page 659–667, New York, NY, USA, 1999. Association for Computing Machinery. doi:10.1145/301250.301428.

[Li24] Zeyong Li. Symmetric exponential time requires near-maximum circuit size: Simplified, truly uniform. In Bojan Mohar, Igor Shinkar, and Ryan O'Donnell, editors, *Proceedings of the 56th Annual ACM Symposium on Theory of Computing, STOC 2024, Vancouver, BC, Canada, June 24-28, 2024*, pages 2000–2007. ACM, 2024. doi:10.1145/3618260.3649615.

[LO22] Zhenjian Lu and Igor C. Oliveira. Theory and applications of probabilistic kolmogorov complexity. *Bull. EATCS*, 137, 2022. URL: http://bulletin.eatcs.org/index.php/beatcs/article/view/700.

[LV19] Ming Li and Paul M. B. Vitányi. *An Introduction to Kolmogorov Complexity and Its Applications, 4th Edition*. Texts in Computer Science. Springer, 2019. doi:10.1007/978-3-030-11298-1.

[MSN94] Rajeev Motwani, Joseph (Seffi) Naor, and Moni Naor. The probabilistic method yields deterministic parallel algorithms. *Journal of Computer and System Sciences*, 49(3):478–516, 1994. 30th IEEE Conference on Foundations of Computer Science. URL: https://www.sciencedirect.com/science/article/pii/S0022000005800698, doi:10.1016/S0022-0000(05)80069-8.

[NW94] Noam Nisan and Avi Wigderson. Hardness vs randomness. *Journal of Computer and System Sciences*, 49(2):149–167, 1994. URL: https://www.sciencedirect.com/science/article/pii/S0022000005800431, doi:10.1016/S0022-0000(05)80043-1.

[NZ96] Noam Nisan and David Zuckerman. Randomness is linear in space. *J. Comput. Syst. Sci.*, 52(1):43–52, February 1996. doi:10.1006/jcss.1996.0004.

[RSW22] Hanlin Ren, Rahul Santhanam, and Zhikun Wang. On the range avoidance problem for circuits. In *63rd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2022, Denver, CO, USA, October 31 - November 3, 2022*, pages 640–650. IEEE, 2022. doi:10.1109/FOCS54457.2022.00067.

[SU05] Ronen Shaltiel and Christopher Umans. Simple extractors for all min-entropies and a new pseudorandom generator. *J. ACM*, 52(2):172–216, March 2005. doi:10.1145/1059513.1059516.

[Tre01] Luca Trevisan. Extractors and pseudorandom generators. *J. ACM*, 48(4):860–879, July 2001. doi:10.1145/502090.502099.

[Uma02] Christopher Umans. Pseudo-random generators for all hardnesses. In *Proceedings of the Thiry-Fourth Annual ACM Symposium on Theory of Computing*, STOC '02, page 627–634, New York, NY, USA, 2002. Association for Computing Machinery. doi:10.1145/509907.509997.

A Proof of Lemma 2

We prove here Lemma 2, which we restate below for convenience:

Lemma (Lemma 2, Restated)**.** *Assume Hypothesis* $\mathsf{WH}(\mathcal{O}, d+1, v)$*,* $d \geq 0$*,* $v \geq 2$*. Then for every time constructible* $\exp^{[d]}(n) \leq T(n) \leq \mathrm{poly}((\exp^{[d]}(n))$ *there is a PRG* $(G_n : \{0,1\}^{s(n)} \to \{0,1\}^n)_{n\in\mathbb{N}}$ *computable uniformly in time* $O^{\langle d+v\rangle}(T(n))$ *which fools* $\mathsf{TIME}^{\mathcal{O}}[T(n)]/3n$ *and has seed length* $s(n) \leq O^{\langle v-1\rangle}(\log n)$*.*

For this we require a strengthening of Theorem 4 due to Umans [Uma02] (following a similar result of Shaltiel-Umans [SU05]) given as follows:

Theorem 22 ([Uma02])**.** *There is a fixed universal constant $\gamma > 0$ so that the following holds. Let $h : \mathbb{N} \to \mathbb{N}$ be any time-constructible "hardness" parameter, $h(n) \leq 2^n$. On input n and given oracle access to a function $f : \{0,1\}^n \to \{0,1\}$, UM^f computes a function $\mathsf{UM}^f : \{0,1\}^{s(n)} \to \{0,1\}^{m(n)}$ in $2^{O(n)}$ time for some time constructible $m(n) = \Theta(h(n)^\gamma)$, $s(n) \leq O(n)$, such that for any $D : \{0,1\}^{m(n)} \to \{0,1\}$ with*

$$|\Pr_{x \sim \{0,1\}^{m(n)}}[D(x) = 1] - \Pr_{z \sim \{0,1\}^{s(n)}}[D(\mathsf{UM}^f(z)) = 1]| \geq \frac{1}{m}$$

there exists a circuit C with D oracle gates computing f whose total size is at most $h(n)$.

Proof of Lemma 2. Assuming Hypothesis $\mathsf{WH}(\mathcal{O}, d+1, v)$, we conclude that for some $\epsilon > 0$, there is a language in $\mathsf{TIME}[\exp^{[d+1]}(n)]$ which is not in $\mathsf{TIME}^{\mathcal{O}}[\Phi_{v,\epsilon}(\exp^{[d+1]}(n))]/\Phi_{v,\epsilon}(2^n)$ even infinitely often. Let $h(n) = \Phi_{v,\epsilon^2}(2^n)$, and define $\tilde{G}_n : \{0,1\}^{s(n)} \to \{0,1\}^{m(n)}$ by $G_n = \mathsf{UM}^{L_n}$ where $s(n) \leq O(n), m(n) = \Theta(h(n)^\gamma)$ are the time constructible bounds guaranteed in Theorem 22. By Theorem 22, if $\tilde{G}_n$ is distinguished by some $D : \{0,1\}^{m(n)} \to \{0,1\}$ then there is an oracle circuit of total size $\leq h(n)$ computing L_n; hence if a distinguisher for $\tilde{G}_n$ exists which is computable in time $t(n)$ with an $\mathcal{O}$ oracle and $O(m(n))$ bits of advice for infinitely many n, then L is decidable with an $\mathcal{O}$ oracle in time $t(n) \cdot h(n)$ with $O(m(n)) + h(n) = O(h(n))$ bits of advice for infinitely many n.

Let $T(n) = (\exp^{[d]}(n))^k$ for some constant k. We then define our final generator G in terms of $\tilde{G}$ by a simple reparameterization of input lengths; for $n \in \mathbb{N}$, we determine the least $\tilde{n}$ such that $m(\tilde{n}) \geq n$ and $T(n) \cdot h(\tilde{n}) < \Phi_{v,\epsilon}(\exp^{[d+1]}(\tilde{n}))$ and set $G_n = \tilde{G}_{\tilde{n}}$ (truncating the output length of $\tilde{G}_{\tilde{n}}$ as necessary). We want to show that G_n fools time $T(n)$ algorithms using an $\mathcal{O}$ oracle and $O(n)$ bits of advice, and runs in time $O^{\langle d+v \rangle}(T(n))$ with seed length $O^{\langle v-1 \rangle}(\log n)$. For the first point, observe that for any $D : \{0,1\}^n \to \{0,1\}$ running in time $T(n)$ with $O(n)$ bits of advice, we obtain a computation of $L_{\tilde{n}}$ running in time $T(n) \cdot h(\tilde{n})$ time with $O(h(\tilde{n}))$ bits of advice, which contradicts our hardness assumption provided $T(n) \cdot h(\tilde{n}) < \Phi_{v,\epsilon}(\exp^{[d+1]}(\tilde{n}))$ and $O(h(\tilde{n})) < \Phi_{v,\epsilon}(2^{\tilde{n}})$ which both hold by construction.

On the other hand the runtime and seedlength of G_n are bounded by $2^{O(\tilde{n})} \cdot \exp^{[d+1]}(\tilde{n})$ and $O(\tilde{n})$ respectively, so it remains to bound the growth rate of $\tilde{n}$. Recall that we chose $\tilde{n}$ to be the least integer such that $m(\tilde{n}) \geq n$ and $T(n) \cdot h(\tilde{n}) < \Phi_{v,\epsilon}(\exp^{[d+1]}(\tilde{n}))$, where $m(\tilde{n}) = \Theta(h(\tilde{n}))^\gamma$. For the first point, we have $h(x) \leq 2^x \leq T(x)$ hence we can bound $\tilde{n}$ by the least integer satisfying $T(n)^2 < \Phi_{v,\epsilon}(\exp^{[d+1]}(\tilde{n}))$, i.e. $(\exp^{[d]}(n))^{2k} < \Phi_{v,\epsilon}(\exp^{[d+1]}(\tilde{n}))$, so it suffices here to take $\tilde{n} = O^{\langle v-1 \rangle}(\log n)$. On the other hand, to have $m(\tilde{n}) \geq n$ it suffices to have $h(\tilde{n}) \geq n$, i.e. $\Phi_{v,\epsilon^2}(2^{\tilde{n}}) \geq n$, so setting $\tilde{n} \leq O^{\langle v-1 \rangle}(\log n)$ suffices. Hence in the end we obtain a runtime of $\exp^{[d+1]}(O^{\langle v-1 \rangle}(\log n)) \leq O^{\langle d+v \rangle}(T(n))$ and seed length $O^{\langle v-1 \rangle}(\log n)$.

□

We will use a slight generalization of Lemma 2 that follows directly by padding (the difference compared to the previous lemma is merely that we allow a slightly more general upper bound on $T(n)$):

Corollary 4. *Assume Hypothesis $\mathsf{WH}(\mathcal{O}, d+1, v)$, $d > 0$, $v \geq 2$ and let $\ell \geq d$ be a fixed constant. Then for every time constructible $\exp^{[d]}(n) \leq T(n) \leq O^{\langle d+v \rangle}(\exp^{[d]}(n))$ there is a PRG $(G_n : \{0,1\}^{s(n)} \to \{0,1\}^n)_{n \in \mathbb{N}}$ computable uniformly in time $O^{\langle d+v \rangle}(T(n))$ which fools $\mathsf{TIME}^{\mathcal{O}}[T(n)]/3n$ and has seed length $s(n) \leq O^{\langle v-1 \rangle}(\log n)$.*

Proof. Using Lemma 2, from our hardness assumption we get a generator fooling $\mathsf{TIME}^{\mathcal{O}}[\exp^{[d]}(n)]/3n$ with runtime $O^{\langle d+v\rangle}(T(n))$ and seed length $s(n) \leq O^{\langle v-1\rangle}(\log n)$. On input length n, we choose some $n' = n'(n)$ such that $\exp^{[d]}(n') \geq T(n)$ and apply our generator on input length n'; clearly we may use this generator for input length n as well, by considering a language $L \subseteq \{0,1\}^n$ as $L' \subseteq \{0,1\}^{n'}$ depending on only the first $n \leq n'$ bits. It suffices to take $n' = O^{\langle v\rangle}(n)$, in which case the seed length as a function of n is $O^{\langle v-1\rangle}(\log O^{\langle v\rangle}(n)) \leq O^{\langle v-1\rangle}(\log n)$ and the runtime is $O^{\langle d+v\rangle}(\exp^{[d]}(O^{\langle v\rangle}(n))) \leq O^{\langle d+v\rangle}(\exp^{[d]}(n)) = O^{\langle d+v\rangle}(T(n))$. □

We are now ready to prove Theorem 7:

Theorem (Theorem 7, Restated)**.** *Assume Hypothesis* $\mathsf{WH}(\mathcal{O}, d+1, v)$ *with* $d \geq 0$, $v \geq 2$, *fixed constants, and* $k \in \mathbb{N}$ *a fixed constant. Then there is a pseudorandom generator with seed length* $O^{\langle v-1\rangle}(\log^{[d+1]}(n))$ *and runtime* $O^{\langle d+v\rangle}(n)$ *that fools* $\mathsf{TIME}^{\mathcal{O}}[n^k]/\log^{[d]}(n)$ *with error* $(2\log^{[d]}(n))^{-1}$ *whenever* $\mathsf{K}^{n^k}(n) \leq \log^{[d]}(n)$.

Proof. Let $k \geq 1$, $d \geq 0$, $v \geq 2$ be fixed constants and let $T(n) = n^k$. We prove by induction on d that there is a pseudorandom generator $(\mathcal{G}_n^d : \{0,1\}^{s_d(n)} \to \{0,1\}^n)_{n\in\mathbb{N}}$ which runs in time $O^{\langle d+v\rangle}(n)$ and fools $\mathsf{TIME}^{\mathcal{O}}[n^k]/\log^{[d]}(n)$ with error $\leq \sum_{j\leq d}(\log^{[j]}(n))^{-1}$ and has seed length $s_d(n) \leq O^{\langle v-1\rangle}(\log^{[d+1]}(n))$, provided the input length n satisfies $\mathsf{K}^{n^k}(n) \leq \log^{[d]}(n)$.

In the case $d = 0$ we may apply Lemma 2 directly. Now, say that the generator $(\mathcal{G}_n^{d-1} : \{0,1\}^{s_{d-1}(n)} \to \{0,1\}^n)_{n\in\mathbb{N}}$ is given, computable in time $T_{d-1}(n) := O^{\langle d-1+v\rangle}(n)$ with $s_{d-1}(n) \leq O^{\langle v-1\rangle}(\log^{[d]} n)$ and fooling $\mathsf{TIME}^{\mathcal{O}}[n^k]/\log^{[d-1]}(n)$ with error $\sum_{j\leq d-1}(\log^{[j]}(n))^{-1}$. Let t_d be a time constructible function such that $T_{d-1}(n) \leq 3t_d(s_{d-1}(n))$; we may set $t_d = O^{\langle d-1+v\rangle}(\exp^{[d]}(n))$. Now, let $(G_n : \{0,1\}^{s'(n)} \to \{0,1\}^n)_{n\in\mathbb{N}}$ be the generator guaranteed by Corollary 4 for time bound $t_d(\cdot)$; so G_n fools $\mathsf{TIME}^{\mathcal{O}}[t_d(n)]/3n$, runs in time $t'_d(n) = O^{\langle d+v\rangle}(t_d(n))$, and has error n^{-1} and seed length $s'(n) = O^{\langle v-1\rangle}(\log n)$. We then set $\mathcal{G}_n^d$ to be the generator with seed length $s_d(n) = s'(s_{d-1}(n))$ given by $\mathcal{G}_n^d(z') = \mathcal{G}_n^{d-1}(G_{s_{d-1}(n)}(z'))$. The run time of $\mathcal{G}_n^d$ is bounded by some constant times the sum of the run times of the two constituent generators, which overall is bounded by $O^{\langle d-1+v\rangle}(n) + O(t'_d(s_{d-1}(n)))$. Let $L \subseteq \{0,1\}^n$ be decided by a $\mathsf{TIME}^{\mathcal{O}}[n^k]/\log^{[d]}(n)$ machine; recall that $n = \exp^{[d-1]}(\ell)$ for some ℓ. Define $L' \subseteq \{0,1\}^{s_{d-1}(n)}$ given by $L' = \{z \mid \mathcal{G}_n^{d-1}(z) \in L\}$; since $\mathcal{G}_n^{d-1}$ fools L with error $\epsilon := \sum_{j\leq d-1}\log^{[j]}(n)$, we have that $\Pr[y \in L] \in \Pr[z \in L'] \pm \epsilon$. On the other hand L' is decidable in time $\leq 3t_d(s_d(n))$ with $2\log^{[d]}(n) + O(1)$ bits of advice provided n is of the form $\mathsf{K}^{n^k}(n) \leq \log^{[d]}(n)$: we use the advice for deciding L, together with $\log^{[d]}(n)$ bits of advice describing the number n, and $O(1)$ advice to specify the code for $\mathcal{G}^{d-1}$ and the procedure explained herein. Hence $G_{s_{d-1}(n)}$ fools L' to within error $\delta := (s_{d-1}(n))^{-1} \leq (\log^{[d]}(n))^{-1}$, so overall we must have that $\mathcal{G}_n^d = \mathcal{G}_n^{d-1} \circ G_{s_{d-1}}$ fools L to within error $\epsilon + \delta \leq \sum_{j\leq d}(\log^{[j]}(n))^{-1}$. It remains to bound the seed length and runtime of $\mathcal{G}_n^d$. The seed length is $O^{\langle v-1\rangle}(\log s_{d-1}(n))$, and $s_{d-1} = O^{\langle v-1\rangle}(\log^{[d]}(n))$, so overall the seed length is bounded by $O^{\langle v-1\rangle}(\log^{[d+1]}(n))$. On the other hand the runtime is dominated by $t'_d(s_{d-1}(n)) = O^{\langle d+v\rangle}(\exp^{[d]}(s_{d-1}(n))) = O^{\langle d+v\rangle}(\exp^{[d]}(O^{\langle v-1\rangle}(\log^{[d]}(n)))) \leq O^{\langle d+v\rangle}(n)$.

□

ECCC ISSN 1433-8092
https://eccc.weizmann.ac.il