Electronic Colloquium on Computational Complexity, Report No. 211 (2023)

Characterizing the Power of (Persistent) Randomness in Log-space

Rafael Pass* Oren Renard†

December 23, 2023

Abstract

We study the problem of whether *persistent* randomness is helpful for polynomial-time algorithms that only use *logarithmic* space. In more detail, we consider the class **SearchBP*L**, of *search*-problems that are solvable by a polynomial-time Probabilistic Logspace TMs with *2-way* access (i.e., with multiple, as opposed to one-time, access) to a random tape—denoted P*LM—and demonstrate two complexity-theoretic hardness assumptions that are *equivalent* to **SearchBP*L** = **SearchL**. Namely, the following are equivalent:

1. **SearchBP*L** = **SearchL**,
2. The gap problem, of deciding whether the conditional Kolmogorov-Space complexity of strings KS is "large" (i.e., $\mathsf{KS}(x \mid y) \geq n-1$) or "small" (i.e., $\mathsf{KS}(x \mid y) \leq O(\log n)$), is worst-case hard for P*LM with sufficiently bounded time and space, given *any* sufficiently long auxiliary input y.
3. The existence of a logspace computable function $f\colon \{0,1\}^n \to \{0,1\}^n$, that is *almost-all-input hard* for sufficiently space and time bounded P*LM, even in the presence of a bounded-length leakage computable by a (sufficiently time/space-bounded) P*LM that is given access to the solution.

Our results leverage and extend earlier characterizations of **BPP** = **P** of Liu and Pass (CCC'22, CCC'23), and relies on the recent space-efficient reconstruction technique of Doron and Tell (CCC'23).

Taken together with the earlier results of Liu and Pass, our results enable simple comparison between the hardness assumptions required to derandomize **SearchBPP** (or equivalently **BPP**) vs. **SearchBP*L**.

*Tel Aviv University and Cornell Tech. Supported in part by NSF Award CNS 2149305, AFOSR Award FA9550-23-1-0387, the Algorand Centres of Excellence programme managed by Algorand Foundation, and DARPA under Agreement No. HR00110C0086. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the United States Government, United States Airforce, DARPA or the Algorand Foundation. email: rafaelp@tau.ac.il

†School of Computer Science, Tel Aviv University. Supported by the European Research Council (ERC) under the European Unions Horizon 2020 research and innovation programme (Grant agreement No. 835152, DARPA under Agreement No. HR00110C0086 and AFOSR Award FA9550-18-1-0267). email: orenrenard@mail.tau.ac.il

ISSN 1433-8092

Contents

1 Introduction

Randomness is an ubiquitous tool in algorithm design. A central problem in complexity theory concerns the question of whether all randomized algorithms can be derandomized; that is, can every randomized polynomial-time algorithm be simulated by a deterministic polynomial-time? In this work (in accordance with the recent literature), we consider this question with respect to promise problems; by slightly abusing notations, we refer to **BPP** as the class of *promise problems* (as opposed to languages) that can be solved in probabilistic polynomial time (with 2-sided error), and **P** to the class of promise problems than can be solved in deterministic polynomial time; **SearchBPP**, **SearchP** denotes the search version of the promise problems (i.e., the tasks of finding a witness for some problem).

The most general question of whether "all algorithms can be derandomized" amounts to the question of whether **SearchBPP** = **SearchP**, but by a result by Goldreich [Gol11], this question is equivalent to the "simpler" **BPP** = **P** question, which has been the focus of a long line of research.

The Hardness v.s. Randomness Paradigm Starting in the 1980s, a sequence of works originating with the works of Blum and Micali [BM84], Yao [Yao82], Nisan [Nis91], Nisan and Wigderson [NW94], Babai et al. [BFNW91], and Impagliazzo and Wigderson [IW97] have presented beautiful connections between the **BPP** v.s. **P** and the problem of proving computational-complexity lower bounds—the so-called *hardness v.s. randomness* paradigm.

For instance, the results of [NW94; IW97] show that **BPP** = **P** under the assumption that **E** = **DTIME**($2^{O(n)}$) contains a language that requires Boolean circuits of size $2^{\Omega(n)}$ for almost all input lengths (i.e., **E** is not contained in $\mathsf{ioSIZE}(2^{\Omega(n)})$). Additionally, results by Impagliazzo, Kabanets, and Wigderson [IKW02] show a *partial* converse: if **BPP** = **P**, then some non-trivial circuit lower bound must also hold. While the above works give very clean complexity theoretic hardness assumptions that imply derandomization, and those implied by it, it is not clear whether these are necessary. More recently, Chen and Tell [CT22] introduced a new class of "almost-all-input" hardness assumption, and Liu and Pass [LP22; LP23] presented uniform complexity-theoretic hardness assumption of the almost-all-input hardness type (w.r.t., the Time-Bounded Kolmogorov complexity problem, and Leakage-Resilient Hard functions) that *characterize* **BPP** = **P** (and thus also **SearchBPP** = **SearchP**).

More broadly, several papers in the recent years [CT22; LP22; LP23; Kor22] presented different assumptions that are sufficient or equivalent to derandomize **BPP**, yet the connections between these results is a priori unclear. Chen, Tell, and Williams [CTW23] managed to bridge the gap by presenting a new framework dubbed "derandomization vs. refutation", proving that each of the aforementioned derandomizations' assumptions can be spelled out in terms of a "refutation". Specifically, a refuter for a lower bound $f \notin \mathcal{C}$ (for some function f and complexity class $\mathcal{C}$) is an algorithm that given a description of $C \in \mathcal{C}$, outputs efficiently a proof that C fails to compute f—namely an x such that $f(x) \neq C(x)$. In essence, "refutation" is the constructive version of lower bound, and in these notations, [CTW23] managed to compare and improve these papers by weakening the complexity class $\mathcal{C}$ and strengthening the function f, thereby presenting weaker hardness assumption that is equivalent to **BPP** = **P**. This framework, however, fails to capture assumptions characterizing partial derandomization, which some of the aforementioned papers do.

Derandomization of Logspace Algorithms If the original randomized algorithm only employs a small—*logarithmic*—amount of space, we ideally would also like the derandomized algorithm to use a logarithmic amount of space. This setting is the focus of the current paper. As explained by Nisan [Nis93], there are two quite different reasonable models of logspace randomized computation: (a) the *multi-access* (or *two-way access*) model, and (b) the *single-access* (or *one-way access* model).

- In the **one-time/one-way access** model, the algorithm can only access each bit on its random tape *once* (i.e., the algorithm is specified using a TM that only has *one-directional* access to a long

random tape). This corresponds to a setting where randomness is generated on the fly (e.g., using a coin-toss), and earlier random coins may not necessarily be remembered, since the algorithm only has logarithmic space. We refer to **BPL** (resp. **SearchBPL**) as the promise (reps. promise-search) problems solvable by logarithmic space algorithms with such one-way access to randomness. It is well known that in this one-way access model, any log space algorithm must also run in polynomial time (see eg, [AB09; Vad12]).

- In the **multi/two-way access** model, on the other hand, we consider a log space algorithm that may access this random tape multiple times (i.e., using a TM that has two-way access to the random tape). This corresponds to algorithms that on their own only have a small a amount of storage, but they have access to some potentially much larger amount of randomness (e.g., by observing the sky/sunspots etc.) and this randomness is *persistent*. As observed by Nisan [Nis93], such log-space randomized algorithms may not necessarily run in polynomial time (as the number of configurations the machine can be in no longer is polynomially bounded). Consequentially, we need to explicitly also limit the running time to be polynomial. We refer to $\mathbf{BP^*L}$ (resp. $\mathbf{SearchBP^*L}$) as the promise (reps. promise-search) problems solvable by logarithmic space algorithms with such two-way access to randomness.

The one-way access model is more well-studied, and while the general problem of whether $\mathbf{BPL} = \mathbf{L}$ is still wide open, there have been promising progress toward resolving it unconditionally [Sav70; SZ99; Rei08; Hoz21]. There are also *conditional* derandomization results, but as far as we know, they all also apply to the two-way accessible model (and will be discussed shortly).

The main focus of the current paper is on the more powerful (and less studied) two-way accessible model to randomness, and on the question of whether such randomized algorithms can be derandomized. We note that in the setting of log space algorithms, the search-to-decision reduction of Goldreich [Gol11] no longer applies, so to understand whether randomness is useful for log space algorithm, we directly study the question of whether $\mathbf{SearchBP^*L} = \mathbf{SearchL}$ (which implies that $\mathbf{BP^*L} = \mathbf{L}$), with the goal of providing hardness assumptions that characterize it. In particular, by doing so, we will attempt to gain insights into the different types of hardness assumptions required to derandomize **SearchBPP** (or equivalently **BPP**) v.s. $\mathbf{SearchBP^*L}$.

As far as we know, prior to our work, only few works have investigated conditional derandomization of space bounded machines. Klivans and Melkebeek [KM02] implemented the pseudorandom generator (PRG) construction of Nisan and Wigderson [NW94]/Impagliazzo and Wigderson [IW97] in logarithmic space[1], given that they start off with a function that can be computed in linear space. As a consequence, they show that under a variation of the assumption of [NW94; IW97]—namely, the existence of a linear-space (as opposed to exponential-time) computable function $f\colon \{0,1\}^n \to \{0,1\}$ that cannot be implemented by circuits of size $2^{\Omega(n)}$—it holds that $\mathbf{BPL} = \mathbf{L}$. In fact, their proof extends also to show that $\mathbf{SearchBP^*L} = \mathbf{SearchL}$ under the same assumption. Recently, [DT23] managed to weaken the assumption of [KM02], showing how to rely on hard functions for $\mathbf{TC}^0$ with oracle access to ROBPs; they also present *uniform* assumptions, similar in spirit to the leakage-resilient hardness assumption of [LP23] under which $\mathbf{BPL} = \mathbf{L}$ (plus an additional uniform cryptographic assumptions).[2]

1.1 Our Results

Our main results shows that appropriate analogous of the characterizations of $\mathbf{BPP} = \mathbf{P}$ in terms of the almost-all-instance hardness [CT22] of (a) some resource bounded Kolmogorov complexity problem

[1] Specifically, they showed that the so-called "designs" in the NW PRG [NW94] can be implemented using pair-wise independence distributions, thereby reducing its space complexity; see [KM02] for full details.

[2] We note that while the results of [DT23] are all stated using this additional cryptographic assumption, this assumption only seems to be used to get a tighter simulation (in terms of the space overhead), and it would seem that one can dispense of it if only aiming to show that $\mathbf{BPL} = \mathbf{L}$.

[LP22], and (b) leakage-resilient hardness [LP23] can be extended also to characterize **SearchBP*L** = **SearchL**.

Towards explaining these results, let us briefly introduce the notions that we will consider:

Kolmogorov-Space Complexity The conditional *Kolmogorov-Space complexity* [Sol64; Cha69; Kol68; Har83; Ko86; Tra84; ABK+06; AKRR11], $\mathsf{KS}(x \mid y)$, of a string x conditioned on a string y, is defined as the minimal "cost" (w.r.t. description size and space complexity) of a TM that prints the string x, when given (for free) access to the auxiliary input y; here the cost is defined as the sum of the program's description length and its space complexity:

$$\mathsf{KS}(x \mid y) \stackrel{def}{=} \min_{\Pi \in \{0,1\}^*} \{|\Pi| + \text{space}(\Pi(y)) \mid \Pi(y) = x\}.$$

This Kolmogorov-problem, as well as other resource bounded variants of it, were introduced and studied in [ABK+06; AKRR11], and can be view of a "linear-charge" version of the notion of *Levin-Kolmogorov complexity* [Lev84; Lev85; Tra84] (where instead of adding the space used by the algorithm, one charges *logarithmically* for running time); this Levin-Kolmogorov complexity was used in the characterization of **BPP** = **P** of [LP22]. Though introduced nearly two decades ago, limited new information has emerged about that problem since the publication of the original paper: Allender et al. [ABK+06] considered the *un*-conditional $\mathsf{KS}(\cdot)$ complexity, and presented several basic facts about it; moreover, they showed that the characteristic language of strings with high $\mathsf{KS}(\cdot)$ complexity[3] is complete for **PSPACE** under **ZPP** reductions.

Following [LP22], we will consider the gap-problem version of $\mathsf{KS}(\cdot \mid \cdot)$, denoted by $\mathsf{GapMcKSP}[a(n), b(n)]$; here the problem is, given an auxiliary string y, and an input x that has either a "low" complexity–that is, $\mathsf{KS}(x \mid y) \leq a(|x|)$—or "high" complexity—that is $\mathsf{KS}(x \mid y) \geq b(|x|)$, to decide which of the two that holds. We emphasize that the **PSPACE**-completeness of $\mathsf{KS}(\cdot)$ due to [ABK+06] is not known to hold for the promise problem that we consider.

We say that a machine M fails to decide $(x, y) \in \mathsf{GapMcKSP}[a(n), b(n)]$ if it err on its decision with high probability, that is:

$$\Pr_r [M[x, y, r] = 1(\mathsf{KS}(x \mid y) \leq a(|x|))] < \frac{2}{3},$$

where $1(\cdot)$ is the indicator function. Following [LP22], we will consider an "almost all-input" [CT22] notion of hardness for this problem: We will consider worst-case of this problem, w.r.t. *all* sufficiently large auxiliary inputs.

Leakage-resilient Space-Hard Functions Following [LP23], we will also consider a notion of *leakage-resilient hard function* $f\colon \{0,1\}^n \to \{0,1\}^n$, parameterized by $(\mathsf{Adv}, \mathsf{Leak}, \ell)$. Here $\mathsf{Adv}, \mathsf{Leak}$ stands for collections of TMs of Adversaries and Leakage algorithms that tries to compute f on a given input x. Specifically, an adversary $A \in \mathsf{Adv}$ tries to compute $f(x)$ given $\ell(|x|)$ bits of leakage $L(x, f(x))$ on $x, f(x)$, produced by a leakage algorithm $L \in \mathsf{Leak}$. We say that A *fails to compute* $f(x)$ if:

$$\Pr_{r,r'} \left[A(x, L(x, f(x), r), r') = f(x)\right] \leq \frac{1}{3},$$

where the probability is over the randomness of A, L.

Following [CT22; LP23], we say that f is an $(\mathsf{Adv}, \mathsf{Leak}, \ell)$-*almost-all-input leakage resilient hard*, if for every pair of algorithms $(A \in \mathsf{Adv}, L \in \mathsf{Leak})$ such that the output of $|L(x', \cdot)| \leq \ell(|x'|)$ for all x', there exist at most finitely many x such that $A(x, L(x, f(x))$ succeeds (i.e., does not fail) to compute $f(x)$. See [LP23] for further discussion of leakage-resilience hardness, and connection to results in Cryptography.

While [LP23] considered leakage-resilient hardness w.r.t. time-bounded attackers, we will here consider hardness against log-space (and polynomial-time) attackers. We refer to $\text{R}^*\text{TISP}(T, S)$ as the class of randomized algorithms with *two-way* access to randomness that run in time $T(\cdot)$ and space $S(\cdot)$.

[3] Namely, the collection of strings $x \in \{0,1\}^*$ s.t. $\mathsf{KS}(x) > |x|^{1/c}$, for some fixed constant $c \in \mathbb{N}$.

Main Theorem Our main result provides two characterizations for derandomizing **SearchBP*L**. Let R*TISP(T, S) denote the collection of all the Probabilistic Turing Machines, that on input length n, runs in time $T(n)$ and space $S(n)$, given 2-way access to their randomness tape.

Theorem 1.1 (Informally, see Theorems 4.1 and 5.1)**.** *The following are equivalent:*

1. **SearchBP*L = SearchL**.
2. *For every large enough constant a, there exists $b, c = O(1)$ so the following holds.*

   For every R*TISP$(n^c, \frac{a}{b}\log n)$ *machine M, for all sufficiently long $y \in \{0,1\}^n$, there exists some $x \in \{0,1\}^n$ such that M fails to decide whether $(x, y) \in \mathsf{GapMcKSP}[a \log n, n-1]$*
3. *There exists constants a, c such that for every constant $b > a$ the following holds. There exists an $(\mathsf{Adv}, \mathsf{Leak}, n^{2/3})$-almost-all-inputs leakage resilient hard function $f\colon \{0,1\}^n \to \{0,1\}^n$, computable in space $b \log n$, where $\mathsf{Leak} \stackrel{def}{=} \mathsf{Adv} \stackrel{def}{=}$ R*TISP$(n^c, a \log n)$.*

In fact, in Theorem 5.1 we show that its sufficient and necessary to consider *deterministic* adversaries for the leakage resilient function, opposed to the probabilistic ones stated in bullet (3); as a consequence, we get that in this context, it is without loss of generality to restrict attention to deterministic attacks (but probabilistic leakage).[4]

We defer the comparison of Theorem 1.1 with previous works to Section 1.2.

We view our contributions as mostly conceptual in nature, identifying the right problems that can be used to characterize **SearchBP*L = SearchL**: on a high level, our proof will follow the proof approaches in [LP22; LP23] (which in turn rely on [Gol11; NW94; IW97; CT22]), but to perform the reductions in log space, we will need to be more careful. Indeed, while as mentioned above [KM02] managed to provide a log space implementation of the NW/IW-PRG [NW94; IW97], we need to ensure also that the *security reduction* is log space (and this was not shown by them). Fortunately, very recent works of Doron and Tell [DT23], and independently that of Pyne, Raz, and Zhan [PRZ23], provide exactly the right tools to enable this.

Characterizing partial derandomization We additionally show that the "leakage-resilient" characterization extends also to capture *partial* derandomization—that is, characterizing when **SearchBP*L** $\subseteq$ **SearchL**$^{1+\gamma}$ for some $\gamma \geq 1$ (as opposed to just the case when $\gamma = 1$ as in Theorem 1.1).

Theorem 1.2. *Let $\gamma \geq 0$. The following are equivalent:*

1. **SearchBP*L** $\subseteq$ **SearchL**$^{1+\gamma}$.
2. *There exists constants a, c such that for every constant $b > a$ the following holds. There exists an $(\mathsf{Adv}, \mathsf{Leak}, n^{2/3})$ leakage resilient hard function $f\colon \{0,1\}^n \to \{0,1\}^n$ for almost-all-inputs, computable in space $b \log^{1+\gamma} n$, where $\mathsf{Leak} \stackrel{def}{=} \mathsf{Adv} \stackrel{def}{=}$ R*TISP$(n^c, a \log n)$.*

The "Kolmogorv" characterization falls short of capturing partial derandomization as one of its ingredients (the invocation of [BF99]) relies on some "nice" property of the underlying classes, which do not hold for **SearchL**$^{1+\gamma}$.

[4] We remark that this statement is not trivial since the length of the leakage is small so the leakage function cannot simply sample the randomness for the attacker.

1.2 Comparison with previous works

Comparison with [LP22; LP23] Let us compare our characterizations with those of $\mathbf{BPP} = \mathbf{P}$ [LP22; LP23]. Our bullet (2) is very similar to the assumption characterizing $\mathbf{BPP} = \mathbf{P}$ from [LP22]. There are two differences: (a) [LP22] considers Levin-Kolmogorov complexity Kt [Lev84] (as opposed to KS) which charges logarithmically for running time (as opposed to charging linearly for space), and (b) [LP22] considered hardness w.r.t. polynomial-time bounded machines, as opposed to log-space bounded and polynomial-time algorithms.

Our bullet (3) is very similar to the leakage-resilient hardness assumption of [LP23] but again with two differences: (a) whereas [LP23] requires the function to be computable in polynomial time, we require it to be logspace computable, and (b) [LP23] considers hardness w.r.t. probabilistic time-bounded leakage and adversaries, as opposed to probabilistic log-space bounded (and polynomial-time).

In essence, in both cases the hardness condition is *weaker* (i.e., we consider hardness only w.r.t. polynomial-time *and* log-space bounded, as opposed to simply polynomial-time bounded, algorithms). But in both case, we consider problems that are at least as *easy* (and potentially easier): in bullet (2), the $\mathsf{GapMcKSP}[O(\log n), n-1]$ problem can trivially be solved by a solver for the $\mathsf{GapMcKtP}[O(\log n), n-1]$ problem considered in [LP22][5], and in bullet (3) any log-space computable function trivially gives a polynomial-time computable function.

Comparison with [DT23] Recently Doron and Tell [DT23] found several new sufficient assumptions to derandomize $\mathbf{BPL}$, and continued the hardness vs. randomness framework of [NW94; IW97; KM02] while incorporating more modern techniques (see references therein). While [DT23] indeed relies on the read-once property of randomness in $\mathbf{BPL}$, they seemingly do so only to achieve a small space overhead. Thus, it seems that minor variants of their assumptions also lead to derandomization of $\mathbf{BP^*L}$ (i.e., that $\mathbf{BP^*L} = \mathbf{L}$).

In particular, the assumption most related to our results is that of an almost-all-input "compression-hard functions" against probabilistic log-space attackers with 1-way access to randomness. [DT23, Theorem 3] shows that this assumption (together with a cryptographic hardness assumption) suffices for showing that $\mathbf{BPL} = \mathbf{L}$ (with small space complexity overhead), but as mentioned it would seem that the cryptographic assumption is not needed to simply get that $\mathbf{BPL} = \mathbf{L}$. As we show in Appendix C, the "compression hard function" assumption, albeit when considered against log-space attackers with 2-way access to randomness, is *syntactically* equivalent to our concept of leakage-resilient hardness. As such, according to our results, their assumption is not only sufficient but also necessary for derandomization (at least when one wants to derandomize also $\mathbf{SearchBP^*L}$, and not just $\mathbf{BPL}$).

Comparison with [Hir20] Hirahara [Hir20] have shown characterization of (black-box) Hitting Sets Generators (HSGs), as opposed to our study of derandomization, by studying a close problem to $\mathsf{GapMcKSP}$. More specifically, he consider the same promise problem, but without any conditioning/auxiliary inputs, and shows that hardness with respect to circuits (as opposed to uniform probabilistic log-space poly-time algorithms as we do) implies HSGs that are hard with respect to the same class of circuits.

On a high level, Hirahara uses the PRG construction of [NW94; IW97; KM02], albeit extending its seed to use additional $O(\log n)$ random bits to select a program, whose output is essentially considered as the truthtable of the supposedly hard function in [KM02]. The construction is claimed to be HSG, and the proof is by a reduction from any attacker to the HSG to a distinguisher that decides whether $\mathsf{KS}(x)$ is small (i.e. $\leq O(\log n)$) or large (i.e. $\geq n-1$).

Our targeted-HSG construction relies on similar principles. Similarly to [Hir20], we use the first $O(\log n)$ bits of the seed to select a program Π, but we apply it on the target string $\sigma \in \{0,1\}^n$, and get

[5] To see this, first observe that $\mathsf{KS}(x \mid y) \leq O(\log n)$ implies x is producible by a machine Π, with description size at most $O(\log n)$, such that $\Pi(y)$ is computable in at most $O(\log n)$ space, and hence runs in time at most $\mathrm{poly}(n)$. This implies $\mathsf{Kt}(x \mid y) \leq O(\log n)$. Hence, any Kt decider for complexity $\leq O(\log n)$ also decides correctly whether $\mathsf{KS}(x \mid y) \leq O(\log n)$.

a string $\Pi(\sigma) \in \{0,1\}^n$. In contrast, [Hir20] applies it on $i = 1, \ldots, n$, and consider the considers the string $\Pi(1)_1, \ldots, \Pi(n)_1$, namely the first output bit of each invocation. Each of these strings is considered as the hard function in the respective construction. In our case, as we shall see, we can show that any distinguisher for the targeted HSG that works given the target σ can be used, together with the [DT23] reconstruction procedure, to distinguish for any $x \in \{0,1\}^{|z|}$ whether $\mathsf{KS}(x \mid z)$ is small or large. Hirahara [Hir20] employs similar techniques, but as mentioned, he considers a different promise problem, and derives a (black-box) HSG.

Comparison with [MPV15] Mandal, Pavan, and Vinodchandran [MPV15] investigated space-bounded machines with *bounded* number of times access to the random tape. Specifically, they considered generalization of the read-once model, where multiple read-once passes are allowed. Such a model is strictly stronger than the standard read-once model, yet weaker than our model—which essentially allows polynomial number of read-once accesses to the random tape. Then they showed that if the set of languages decidable by log-space machines, that uses $O(\log^{1+\omega(1)} n)$ random bits, and allowed to have $\omega(1)$ number of read-once accesses to it, is contained in $\mathbf{P}$, then there exists a non-trivial derandomization of *time* machines, namely $\mathbf{BPTIME}(n) \subseteq \mathbf{DTIME}(2^{o(n)})$.

In our terminology, those languages are strictly contained in $\mathbf{BP^*L}$. In particular, derandomization of the sort $\mathbf{SearchBP^*L} = \mathbf{SearchL}$ implies $\mathbf{BP^*L} = \mathbf{L} \subseteq \mathbf{P}$, and by their work, it further implies non-trivial derandomization of time machines.

2 Proof overview

While our results are inspired and follows ideas presented in [LP22; LP23], they overcome several challenges that arise in the space-bounded settings—most notably, we leverage the very recent work of [DT23].

2.1 The class SearchBP*L

Let us start by defining the class of search problems, $\mathbf{SearchBP^*L}$. As any search class, it consists of relations $R \subseteq \{0,1\}^* \times \{0,1\}^*$ such that (1) we can "efficiently" verify containment in R—through a Verification algorithm—and (2) we can "efficiently" find—through a so-called Finder algorithm—a witness y for any "true" statement x (for which there exist some y s.t. $(x,y) \in R$). By "efficient", we mean here by a P*LM algorithm, where P*LM refers to the class of logarithmic space and polynomial-time algorithms with two-way access to randomness.

2.2 From Derandomization to Hardness

We first show that derandomization of $\mathbf{SearchBP^*L}$ implies the two hardness conditions. Towards this, let us first recall the notion of Pseudo-Random Generators (PRGs) [BM84; NW94] and *targeted-PRGs* [Gol11], that will be important for our results. An ε-PRG $G\colon \{0,1\}^s \to \{0,1\}^n$ is an "efficiently" computable function (where in our context, efficiency will mean log-space computable) whose output given a random input is indistinguishable from random w.r.t. some class of algorithm; namely, if M is a machine in that collection, then

$$\left| \mathop{\mathbb{E}}_{r \in_R \{0,1\}^n} [M[r]] - \mathop{\mathbb{E}}_{x \in_R \{0,1\}^s} [M[G(x)]] \right| \leq \varepsilon.$$

Targeted-PRGs, introduced by Goldreich [Gol11], are just like standards PRG G, with the modification that they gets an additional "target" σ as input and the pseudorandomness property of G also holds with respect to machines M that also get the target σ as input, namely:

$$\left| \mathop{\mathbb{E}}_{r \in_R \{0,1\}^n} [M[\sigma, r]] - \mathop{\mathbb{E}}_{x \in_R \{0,1\}^s} [M[\sigma, G(\sigma, x)]] \right| \leq \varepsilon.$$

We say that G is an ε targeted-PRG, if the above holds *for every* target σ.

Standard probabilistic arguments can be used to show that a random function is either a targeted-PRG (as was shown by [Gol11; LP22]), or a leakage-resilient hard function (as was shown by [LP23]). Additionally, as shown in [Gol11; LP22; LP23] for PRGs/leakage-resilient functions against the class of time-bounded attackers, the search problem of finding such an object belongs to **SearchBPP**. We start by noting that the analysis can also be extended to the space bounded setting, to show containment in **SearchBP*L**, but this heavily leverages 2-way access to the instance (to be verified) and the randomness. As a consequence, under the assumption that **SearchBP*L** = **SearchL**, these objects can be computed in deterministic logspace.

While this suffices for the characterization in terms of leakage-resilient hardness, it is still left to reduce targeted-PRGs to worst-case hardness of KS. Here we observe that, similarly to the time-bounded regime, strings in the range of a targetted-PRG have significantly smaller KS complexity than random string (with high probability):

1. The logspace generator has constant description-length given the target, so all the generated strings have *conditional* KS complexity $\leq O(\log n)$
2. Whereas, by a standard counting argument, most strings have KS complexity $\geq n - O(1)$.

Thus, a decider for KS (given auxiliary input σ) must also distinguish the targeted-PRG (given target σ) which concludes the proof.

2.3 From Hardness to Derandomization

The more interesting direction of the characterization is showing derandomization of **SearchBP*L** from the hardness assumptions. Here, both of our characterizations are based on variants of the Nisan-Wigderson (NW) PRG [NW94]. To deal with the fact that we only assume *worst-case* hardness, we will rely on the variant of Impagliazzo and Wigderson [IW97], which as observed by Klivans and Melkebeek [KM02] is also constructible in logspace (specifically, they showed a logspace implementation for the "designs" of [NW94], using pair-wise independence). Additionally, we require a logspace *reconstruction* procedure, achieved only recently by Doron and Tell [DT23] by further modifying the construction. Essentially, they presented a space-efficient variant of [STV99], another PRG variant from the same line of works.

This utilization of [DT23] will be a crucial component of our construction. Henceforth, we denote their PRG by DT^f (given a "hard" function f). We highlight that the use of this PRG is different from the approach in [LP22; LP23], where log space reconstructibility was not a concern.

2.3.1 A Blackbox Logspace-Reconstructible PRG

Our results rely on stronger objects than simply PRGs based on hard functions. We will require a PRG where (a) the PRG construction is computable in (deterministic) linear (in the seed length) space while having black-box access to the "hard function" f, and (b) the security reduction (i.e., the "reconstruction" algorithm) is blackbox in both the distinguisher and the function f, where the reduction is computable in randomized log space (with two-way access to randomness). We refer to such a PRG as a *black-box logspace reconstructible PRG*, and note that this is a strengthening of the notion of a black-box reconstructible PRG of [LP23] (by adding the extra space-efficiency requirements), which in turn is a strengthening of the notion of a black-box PRG from [Vad12] (which only requires the reconstruction algorithm to be black-box in the distinguisher, but not the function).

In more details, we require a function $G^f \colon \{0,1\}^s \to \{0,1\}^n$ that is linear space computable, and log space machines $\mathcal{A}, \mathcal{R}$, that given any distinguisher D s.t.

$$\mathop{\mathbb{E}}_{x \in_R \{0,1\}^s}\left[D[G^f(x)]\right] - \mathop{\mathbb{E}}_{r \in_R \{0,1\}^n}[D[r]] \geq 0.1,$$

it holds that

$$\Pr_r\left[\forall y, \quad \mathcal{R}^D[1^n, \mathcal{A}^{D,f}[1^n, r], y] = f(y)\right] \geq 2/3.$$

The "black-box reconstruction" terminology stems from the fact that the reconstruction algorithm has oracle access to both D, f.

We note that such black-box reconstructive PRG was constructed in [DT23], by presenting a space-efficient variant of [NW94; STV99].

2.3.2 Derandomization from Kolmogorov-hardness

Given the above notion of a blackbox logspace reconstructible PRG, we are now ready to derandomize **SearchBP*L** assuming the hardness of GapMcKSP.

Derandomizing R*L We first show how to derandomize **R*L**—the class of all languages decidable, with 1-sided error, by probabilistic poly-time and log-space machines with multiple access to randomness. That is, $L \in \mathbf{R^*L}$ if and only if there exists such a resource bounded TM M such that

$$\begin{aligned} \forall x \in L \quad &\implies \quad \Pr_{r \in_R \{0,1\}^*}[M[x,r] = 1] \geq 2/3 \\ \forall x \notin L \quad &\implies \quad \Pr_{r \in_R \{0,1\}^*}[M[x,r] = 0] = 1. \end{aligned}$$

Towards this, we will rely on the notion of a Hitting-Set Generator (HSG) and more precisely that of a *targeted-HSG*, both of which are natural variants of PRGs and targeted-PRGs (see Section 2.2). An ε-HSG $H\colon \{0,1\}^s \to \{0,1\}^n$ satisfies the property that it "hits" some collection of TMs. Namely, if M is a machine in that collection, then

$$\mathop{\mathbb{E}}_{r \in_R \{0,1\}^n}[M[r]] \geq \varepsilon \quad \implies \quad \mathop{\mathbb{E}}_{x \in_R \{0,1\}^s}[M[G(x)]] > 0.$$

Targeted-HSG, are just like a standard HSG H, with the modification that they gets an additional target σ as input; the "hitting" property of H then holds with respect to uniform algorithms that also get the target σ as input, namely:

$$\mathop{\mathbb{E}}_{r \in_R \{0,1\}^n}[M[\sigma, r]] \geq \varepsilon \quad \implies \quad \mathop{\mathbb{E}}_{x \in_R \{0,1\}^s}[M[\sigma, H(\sigma, x)]] > 0.$$

We say that H is an ε-target HSG, if the above holds *for every* target σ.

Assuming that the gap variant of KS is worst-case hard for suitably restricted machines, we construct a new targeted-HSG H against *deterministic* logspace algorithms (defer for the moment the question why we claim H is targeted-HSG and not a targeted-PRG):

$$H(\sigma, \Pi, x) \stackrel{def}{=} \mathrm{DT}^{\Pi(\sigma)}(x),$$

where, we set $a = O(1)$ as large enough constant and,

- $\sigma \in \{0,1\}^n$ stands for an input target,
- $\Pi \in \{0,1\}^{a \log n}$ is a randomly selected deterministic logspace machine with short description,
- $\Pi(\sigma) \in \{0,1\}^{\mathrm{poly}(n)}$ is the output of Π when simulated on σ, which is associated with the truth-table of supposedly hard function $\{0,1\}^{O(\log n)} \to \{0,1\}$,
- and $x \in \{0,1\}^{O(\log n)}$ is a seed for $\mathrm{DT}^{\Pi(\sigma)}$.

Overall, the construction has $O(\log n)$ seed length and is computable in logspace (because so does $\mathrm{DT}(\cdot), \Pi(\cdot)$).

The correctness is by reduction from a distinguisher D for H, on instance σ, to a suitably efficient solver for $\mathsf{KS}(z \mid \sigma)$. A deterministic distinguisher D for the HSG implies that

$$\mathop{\mathbb{E}}_{r}[D[\sigma, r]] \geq \varepsilon,$$
$$\mathop{\mathbb{E}}_{(\Pi,x)}[D[\sigma, H_\sigma(\Pi, x)]] = 0.$$

Now let us explain how to decide KS. Here we make use of the blackbox Reconstruction and Advice procedures $\mathcal{R}, \mathcal{A}$ of DT that were discussed in Section 2.3.1. The idea is that a, which was defined as $\max|\Pi| \leq a \log n$, and controls the description length of the hard function, could be arbitrary large and *independent* of the space complexity of $\mathcal{A}, \mathcal{R}$. Thus,

1. If $\mathsf{KS}(z \mid \sigma) \leq O(\log n)$, then simulating $\mathcal{A}^{D[\sigma,\cdot],z}$ on the distinguisher $D[\sigma, \cdot]$ and the string $z \in \{0,1\}^{\mathrm{poly}(n)}$ that represents a hard function, would produce (w.h.p.) a good and short advice that helps $\mathcal{R}$ to fully reconstruct z; the catch is that computing everything here is possible in logspace, but "constantly less" than $a \log n$. In that case we decide that z belongs to the yes case.

2. Otherwise, if $\mathsf{KS}(z \mid \sigma) \geq n-1$, then by definition, it is not possible that both the advice is short (say $\leq n/2$) and the reconstruction would succeed to fully reconstruct z while doing so in log space—because then $\mathsf{KS}(z \mid \sigma) \leq 0.75n$. Thus, either the advice is long, or the computation was not in logspace, and hence its safe to always decide z belongs to the no case.

This concludes the HSG correctness, and further shows that $\mathbf{R}^*\mathbf{L} = \mathbf{L}$: the implied derandomization is by simple enumeration of seeds and taking the majority.

Let us highlight why we can only claim that H is a HSG (rather than a PRG): our KS decider needs to solve the problem in *worst-case*, namely it can get an adversarially chosen strings z as input. The reduction, however, only needs to work for "hard functions" z chosen according to the distribution used in the construction. We here rely on the observation that every YES-instance z is in the support of the constructed HSG, and as a consequence, the distinguisher needs to work on those instances (since we only aim to show the hitting property).

It finally follows using standard techniques that such a log space efficient targeted PRG implies derandomization of $\mathbf{R}^*\mathbf{L}$

As a final remark, let us explain why we assumed that $\mathsf{GapMcKSP}$ is hard given *all* sufficiently long auxiliary inputs (as stated in Theorem 1.1). The auxiliary inputs are considered, during the reduction, as the target-instances of the machines. To conclude "worst-case" derandomization, we need by definition, to decide correctly every instance.

Derandomizing $\mathbf{BP}^*\mathbf{L}$ We next show how to derandomize $\mathbf{BP}^*\mathbf{L}$ using the above targeted HSG. Towards this, we will build on the result of Buhrman and Fortnow [BF99]: They considered the class $\mathbf{RP}$, of all the problems decidable by polynomial time algorithms with 1-sided error and showed that $\mathbf{RP} = \mathbf{P} \implies \mathbf{BPP} = \mathbf{P}$. In other words, to derandomize $\mathbf{BPP}$, we can without loss of generalize focus on derandomizing $\mathbf{RP}$. We observe that their proof can be extended (with only minor modifications) to apply in the logspace regime given 2-way access to randomness, and conclude that $\mathbf{R}^*\mathbf{L} = \mathbf{L}$ implies that $\mathbf{BP}^*\mathbf{L} = \mathbf{L}$. This implies that our constructed targeted-HSG further implies $\mathbf{BP}^*\mathbf{L} = \mathbf{L}$.

Derandomizing $\mathbf{SearchBP}^*\mathbf{L}$ We finally proceed to showing how to derandomize also $\mathbf{SearchBP}^*\mathbf{L}$. Let F, V be a solution Finder and Verifier for some relation in $\mathbf{SearchBP}^*\mathbf{L}$ (see Section 2.1 for definition). Since the Verifier have 2-way access to solution-candidates, they can be considered as standard *decisional* algorithms for languages in $\mathbf{BP}^*\mathbf{L}$, which we already claimed can be decided deterministically. It only

remains to derandomize the solution Finder F. But since they can reproduce their output (given free access to their random tape), we can consider them, together with the deterministic Verifier, as algorithms that *decide* whether there exists a solution or none; namely solving some problem in $\mathbf{R}^*\mathbf{L}$. The HSG guarantees to hit good random strings in these cases, which here can be converted to good solutions when fed to the Finders. Altogether, $\mathbf{SearchBP}^*\mathbf{L} \subseteq \mathbf{SearchL}$.

2.4 Derandomization from Leakage-resilience

Here we assume the existence of a logspace computable function $f\colon \{0,1\}^n \to \{0,1\}^n$ that is hard to compute, in the presence of bounded length-leakage. Let us defer the parameters, and infer them from the proof.

We use similar construction to the previous paragraph, when using the function f on the instance, rather than a randomly selected TM. Specifically,

$$P(\sigma, x) \stackrel{def}{=} \mathrm{DT}^{f(\sigma)}(x)$$

where σ, x are as before, and $f(\sigma) \in \{0,1\}^n$ is associated with the truth-table of supposedly hard function $\{0,1\}^{\log n} \to \{0,1\}$.

First, its important to observe that the logspace computability of f implies that P is also logspace computable. Now we claim that P is targeted-*PRG* rather than HSG, which also shortcuts the proof.

The correctness is again by reduction. Given any distinguisher D, on instance σ, we construct the following leakage L and adversary A algorithms on input σ. In fact, the black-box construction we make use of give rise to very simple algorithms, that are associated with the Advice and Reconstruction procedures discussed earlier:

$$L(\sigma, f(\sigma)) \stackrel{def}{=} \mathcal{A}^{D[\sigma,\cdot],f(\sigma)}[1^n],$$
$$A(\sigma, adv, k) \stackrel{def}{=} \mathcal{R}^{D[\sigma,\cdot]}[1^n, adv, k],$$

namely the leakage algorithm $L(\sigma, f(\sigma))$ is simply defined as the advice (which is guaranteed to be short), and the adversary $A(\sigma, adv, k)$ simply simulates the reconstruction procedure on the generated advice.

Now observe that since $D[\sigma,\cdot]$ is a distinguisher for $P(\sigma,\cdot)$, then by definition we can invoke Advice and Reconstruction to compute $f(\sigma)$, which is the hard function used by DT. Thus A succeeds to compute (reconstruct) $f(\sigma)_k$ whenever the leakage algorithm produced a good output (advice); these holds by the definitions of $\mathcal{A}, \mathcal{R}$, which contradicts the hardness assumption.

All that's left is to set the parameters. The Advice and Reconstruction algorithms $\mathcal{A}, \mathcal{R}$ are logspace computable, independent of the space takes to compute f. Hence we require f to be computable in a slightly more space (still log space, but constantly larger), which concludes a contradiction.

As a final remark, let us explain why we assumed that the function is leakage-resilient, for almost-all-inputs; this follows from the same argument as in the "Kolmogorov" direction. The inputs to the function are considered, during the reduction, as the target-instances of the machines. To conclude "worst-case" derandomization, the function needs by definition, to be hard on every input.

2.5 Additional notes

Why Blackbox? In both directions we make use of the instance σ to generate "hard functions". The correctness is by reduction from distinguishers to a short and efficient program, or efficient leakage and adversary procedures, that computes the hard function. However, we crucially rely on giving access to σ to the advice procedure, as σ is too long to be hard-wired to the programs. To avoid this, recall that the KS decider is given σ as the *auxiliary* input, namely as "oracle" access; the leakage and adversary algorithms are given σ as inputs, and hence could simulate the Advice and Reconstructions procedures correctly, without "paying" for σ. In turn, we just need to make sure the Advice and Reconstruction produces could use σ in black-box manner.

Leakage resilient vs Kolmogorov hardness Observe that our KS decider simulated both the Advice and Reconstruction procedures $\mathcal{A}, \mathcal{R}$, in contrast to our leakage and adversary algorithms that are associated with either $\mathcal{A}$ or $\mathcal{R}$. Since $\mathcal{R}$ is deterministic, it allows us to consider weaker adversaries for the leakage-resilient function (namely, deterministic).

3 Preliminaries

3.1 Complexity classes, space bounded computation

Turing machines We distinguish between TM that access their randomness tape in 1- or 2-way manner. The notation $M[x; y]$ denotes that a TM M have 2-way access to its input x but 1-way access to y.

Complexity class For clarity, we start by recalling the definition of **prBPL**.

Definition 3.1 (**prBPL**). *The class* **prBPL** *consists of all the promise problems* $(\textsc{Yes}, \textsc{No})$*, such that* $\textsc{Yes}, \textsc{No} \subseteq \{0,1\}^*$*, for which there exists a probabilistic logspace machine M with 1-way access to randomness that always halts, such that:*

1. *for every* $x \in \textsc{Yes}$, $\Pr_r[M[x; r] = 1] \geq \frac{2}{3}$,
2. *for every* $x \in \textsc{No}$, $\Pr_r[M[x; r] = 0] \geq \frac{2}{3}$.

The requirement of the machines to always halts implies that **prBPL** $\subseteq$ **prP** (see, eg, [Vad12; AB09]). The class **prBP*****L** is defined similarly, but allows the machines 2-way access to their random tape. Although a natural variant, it seems to significantly increase the computational power; in particular, it is not clear whether their running time is polynomially bounded (see [Nis93]). To overcome this annoyance, we modify the definition to also impose a polynomial running time bound on the algorithm; more generally, we upper bounds the running time exponentially in the space complexity.

Definition 3.2 (**prBP*****SPACE**, **prR*****SPACE**). *Let $s\colon \mathbb{N} \to \mathbb{N}$ be any function. The class* **prBP*****SPACE**(s) *consists of all the promise problems* $(\textsc{Yes}, \textsc{No})$*, such that* $\textsc{Yes}, \textsc{No} \subseteq \{0,1\}^*$*, for which there exists a Probabilistic TM M, that on n-bit input x, it uses $s(n)$ memory bits, runs at most $2^{O(s(n))}$ time, has 2-way access to its randomness, such that:*

1. *for every* $x \in \textsc{Yes}$, $\Pr_r[M[x, r] = 1] \geq \frac{2}{3}$,
2. *for every* $x \in \textsc{No}$, $\Pr_r[M[x, r] = 0] \geq \frac{2}{3}$.

The class **prR*****SPACE**(s) *is defined similarly, but the machines have only 1-sided error: namely, the machine M (defined as before), satisfies:*

1. *for every* $x \in \textsc{Yes}$, $\Pr_r[M[x, r] = 1] \geq \frac{2}{3}$,
2. *for every* $x \in \textsc{No}$, $\Pr_r[M[x, r] = 0] = 1$.

Finally,

$$\textbf{prBP}^*\textbf{L} \stackrel{def}{=} \bigcup_{a \in \mathbb{N}} \textbf{prBP}^*\textbf{SPACE}(a \log n),$$
$$\textbf{prR}^*\textbf{L} \stackrel{def}{=} \bigcup_{a \in \mathbb{N}} \textbf{prR}^*\textbf{SPACE}(a \log n).$$

More generally, we can restrict both the time and space complexity:

Definition 3.3 (**prBP*****TISP**, **prR*****TISP**)**.** *Let $t, s\colon \mathbb{N} \to \mathbb{N}$ be any two functions. The class* **prBP*****TISP**(t, s) *consists of all the promise problems* $(\textsc{Yes}, \textsc{No})$*, such that* $\textsc{Yes}, \textsc{No} \subseteq \{0,1\}^*$*, for which there exists a probabilistic TM M, that on n-bit input x, it uses $s(n)$ memory bits, runs at most $t(n)$ time, has 2-way access to its randomness, such that:*

1. *for every $x \in \textsc{Yes}$, $\Pr_r[M[x,r] = 1] \geq \frac{2}{3}$,*
2. *for every $x \in \textsc{No}$, $\Pr_r[M[x,r] = 0] \geq \frac{2}{3}$.*

The class **prR*****TISP**(t, s) *is defined similarly, but the machines have only 1-sided error.*

From hereon, we identify complexity classes with their promise version, e.g. **BP*****L** refers to **prBP*****L**.

Resource-Bounded TMs We make use of the following resource-restricted classes of TMs:

1. Deterministic Logspace Machines, denoted DLM, which consists all the TMs M such that for every input $x \in \{0,1\}^*$, $M[x]$ is computable in space $O(\log n)$.
2. Probabilistic* Logspace Machines, denoted P*LM, which consists all the Probabilistic TMs M such that for every input $x \in \{0,1\}^*$ and every coins tosses $r \in \{0,1\}^*$, $M[x,r]$ is computable in space $O(\log n)$ and time $\mathrm{poly}(n)$.
3. Time and Space Machines, denoted $\mathrm{TISP}(T, S)$, which consists all the TMs M such that for every input $x \in \{0,1\}^*$, $M[x]$ runs in deterministic time $T(n)$ and space $S(n)$,
4. Probabilistic* Time and Space Machines $\mathrm{R}^*\mathrm{TISP}(T, S)$, which consists all the TMs M such that for every input $x \in \{0,1\}^*$ and every coins tosses $r \in \{0,1\}^*$, $M[x,r]$ runs in time $T(n)$ and space $S(n)$.

We abuse notations and associate the TMs collections with its resource restrictions; for example, we say that a TM is a DLM if it belongs to that collection.

Lemma 3.1 (Composition of space-bounded machines)**.** *Let $f_1, f_2\colon \{0,1\}^* \to \{0,1\}^*$ be two functions that on input length n, are computable in space $s_1(n), s_2(n) = \Omega(\log n)$, respectively. Then, for every x, $f_2(f_1(x))$ is computable in space $O(s_1(|x|) + s_2(|f_1(x)|))$.*

For any TM M and input x, we denote by $s[M[x]]$ the **space consumption** of $M[x]$.

3.2 Pseudorandom objects

We consider a generalized notion of a *targeted PRG* [Gol11] which, following [LP22], allows also probabilistic classes of distinguishers.

Definition 3.4. *Let $s, m\colon \mathbb{N} \to \mathbb{N}$ and $\varepsilon\colon \mathbb{N} \to [0,1]$.*

An $\varepsilon(n)$ targeted-PRG $G\colon \{0,1\}^n \times \{0,1\}^{s(n)} \to \{0,1\}^{m(n)}$ *against a collection of TMs $\mathcal{C}$, is defined such that, for every sufficiently large n, and for every machine $M \in \mathcal{C}$ and 2-way accessible input $\sigma \in \{0,1\}^n$ (dubbed target henceforth),*

$$\left| \mathop{\mathbb{E}}_{r \in_R \{0,1\}^{m(n)}} [M[\sigma, r]] - \mathop{\mathbb{E}}_{x \in_R \{0,1\}^{s(n)}} [M[\sigma, G(\sigma, x)]] \right| \leq \varepsilon(n).$$

Similarly,

Definition 3.5. *Let $s, m\colon \mathbb{N} \to \mathbb{N}$ and $\varepsilon\colon \mathbb{N} \to [0,1]$.*

An $\varepsilon(n)$ targeted-HSG *$H\colon \{0,1\}^n \times \{0,1\}^{s(n)} \to \{0,1\}^{m(n)}$ against a collection of TMs $\mathcal{C}$ satisfies, that for every sufficiently large n, and for every machine $M \in \mathcal{C}$ and 2-way accessible target $\sigma \in \{0,1\}^n$:*

$$\mathop{\mathbb{E}}_{r \in_R \{0,1\}^{m(n)}} [M[\sigma, r]] \geq \varepsilon(n) \quad \Longrightarrow \quad \mathop{\mathbb{E}}_{x \in_R \{0,1\}^{s(n)}} [M[\sigma, H(\sigma, x)]] > 0.$$

We emphasize that if $\mathcal{C}$ is consists of probabilistic machines, then all the above probabilities are also over additional random string which is provided to M (which access it according to the restrictions of $\mathcal{C}$).

3.3 Kolmogorov Space complexity

Our proofs make use of a Universal Turing machine U, that can simulates any TM Π on input length n in time $O(t \log t)$ and space $O(s)$, where $t = t(n), s = s(n)$ are the time and space complexity of Π on input length n. For a reference, see for example [AB09].

We make use of the space-bounded Kolmogorov problem as was introduced by [ABK+06]. See the paper for several basic facts about it.

Definition 3.6. *The* Kolmogorov Space *bounded complexity of a string $x \in \{0,1\}^*$, given an auxiliary string $y \in \{0,1\}^*$ is defined as*

$$\mathsf{KS}(x \,|\, y) \stackrel{def}{=} \min_{\Pi \in \{0,1\}^*, s \in \mathbb{N}} \{|\Pi| + s[\Pi(y)] \mid U(\Pi(y)) = x\}.$$

Definition 3.7. *Let $a, b\colon \mathbb{N} \to \mathbb{N}$. The promise problem $\mathsf{GapMcKSP}[a,b]$ is compromised with two kinds of strings (x,y), both of length n:*

1. *$(x,y) \in$ YES if $\mathsf{KS}(x \,|\, y) \leq a(n)$,*
2. *$(x,y) \in$ NO if $\mathsf{KS}(x \,|\, y) \geq b(n)$.*

Say that a probabilistic machine M **fails to decide** $(x,y) \in \mathsf{GapMcKSP}[a,b]$, if the pair (x,y) is yes/no instance but

$$\Pr_r [M[x,r] = 1((x,y) \in \text{YES})] \leq \frac{2}{3},$$

where $1(\cdot)$ is the indicator function.

We say that $\mathsf{GapMcKSP}[a,b]$ is worst-case hard for probabilistic complexity class $\mathcal{C}$, given all sufficiently long auxiliary inputs, if for every $M \in \mathcal{C}$, every sufficiently long string y, there exists an x of the same length, so M fails to decide $(x,y) \in \mathsf{GapMcKSP}[a,b]$.

3.4 Searching in Logspace

Let $R \subseteq \{0,1\}^* \times \{0,1\}^*$ be a relation, and denote $R(x) \stackrel{def}{=} \{y \mid (x,y) \in R\}$. The following definitions are of *promise* search problems.

Definition 3.8. *Say $R \in$ **SearchL** if there are* DLM *F, V, such that for every x, y:*

1. *Deterministic finding of good solutions: $F[x] \in R(x)$,*
2. *Deterministic verification of solutions: $V[x,y] = 1(y \in R(x))$.*

Definition 3.9. *Say $R \in$ **SearchBP*L** if there are* P*LM *F, V such that for every x, y:*

1. *Probabilistic finding of good solutions: $\Pr_r [F[x,r] \in R(x)] \geq 2/3$,*
2. *Probabilistic verification of solutions: $\Pr_r [V[x,y,r] = 1(y \in R(x))] \geq 2/3$.*

Definition 3.10. *Say $R \in$ **SearchR*L** if there is* P*LM *F and* DLM *V such that for every x, y:*

1. *Probabilistic finding of good solutions: $\Pr_r [F[x,r] \in R(x)] \geq \frac{1}{\text{poly}(n)}$,*
2. *Deterministic verification of solutions: $V[x,y] = 1(y \in R(x))$.*

3.5 Blackbox PRGs from hard functions

We make use of Blackbox reconstructive PRGs, as was defined in [LP23] (which in turn is based on [Vad12]); by focusing on space efficient derandomization, we adapt the definition to be space-efficient.

Definition 3.11. *Let* $\mathsf{Adv}, \mathsf{Rec}$ *be collections of TMs. Let* $k\colon \mathbb{N}^2 \to \mathbb{N}$ *be a function. Let* $G\colon \{0,1\}^d \to \{0,1\}^n$ *be a deterministic algorithm.*

We say that G *is a* $(\mathsf{Adv}, \mathsf{Rec}, k)$*-*logspace-reconstructive PRG construction*, if there exists machines* $\mathcal{A} \in \mathsf{Adv}$ *and* $\mathcal{R} \in \mathsf{Rec}$*, such that for every function* $f\colon \{0,1\}^{\log m} \to \{0,1\}$*, and arbitrary distinguisher* $D\colon \{0,1\}^n \to \{0,1\}$ *such that*

$$\mathop{\mathbb{E}}_{x \in_R \{0,1\}^d}\left[D(G^f(x))\right] - \mathop{\mathbb{E}}_{r \in_R \{0,1\}^n}[D(r)] \geq \frac{1}{\mathrm{poly}(n)},$$

then $\mathcal{A}^{f,D}[1^{n+m}]$ *will output* $\leq k(n,m)$ *bits so when given as input to* $\mathcal{R}$*, it holds*

$$\Pr_{x \in_R \{0,1\}^{\log m}}\left[\mathcal{R}^D[1^{n+m}, \mathcal{A}^{f,D}[1^{n+m}], x] = f(x)\right] = 1.$$

If Adv *is a collection of probabilistic machines, the above should hold with probability* $\geq 2/3$ *over the randomness of* $\mathcal{A}$*. If* Rec *is a collection of probabilistic machines, then* $\mathcal{R}$ *should be able to compute* $f(x)$ *with probability* $\geq 2/3$ *over its own randomness.*

We observe that there exists a logspace-reconstructive PRG. The construction is due to [DT23], whom only recently showed logspace computable reconstruction variant of [NW94; STV99].

Theorem 3.2 (Implicit in [DT23, Theorem 6.1])**.** *There exists a* $(\mathrm{PLM}, \mathrm{DLM}, k)$*-logspace-reconstructive PRG construction, denoted* $\mathrm{DT}^f\colon \{0,1\}^{O(\log n)} \to \{0,1\}^n$ *such that, for sufficiently large* $m \stackrel{def}{=} n^{O(1)}$ *and* $f\colon \{0,1\}^{\log m} \to \{0,1\}$*, the following holds:*

1. *Explicitness:* G^f *is computable in deterministic space* $O(\log m) = O(\log n)$*.*
2. *Short advice:* $k(n,m) \stackrel{def}{=} m^{2/3}$*.*

The original phrasing differs as follows:

1. Their reconstruction algorithm output a $\mathbf{TC}^0$ circuit that makes non-adaptive queries to the distinguisher D. As was further shown in [DT23], a DLM can evaluate such a circuit when given as input (this follows because: (1) for non-oracle circuits, we have $\mathbf{TC}^0 \subseteq \mathbf{NC}^1 \subseteq \mathbf{L}$, and (2), the inclusion still holds for circuits with oracle to DLMs with non-adaptive queries).
2. They do not distinguish between the Advice and the Reconstruction algorithms; instead, they argue that their reconstruction algorithm produces a relatively small $\mathbf{TC}^0$ circuit. In our consideration, the circuit serves as the Advice's output; the Reconstruction procedure, given access to that circuit, simply evaluates it on a given input (thus, it is computable in log-space, as described in the former bullet).

4 SearchBP*L vs GapMcKSP

The main theorem of this section is:

Theorem 4.1. *For every large enough constant* a*, there exists* $b, c = O(1)$ *so the following holds.* $\mathbf{SearchBP^*L} = \mathbf{SearchL}$ *if and only if* $\mathsf{GapMcKSP}[a \log n, n-1]$ *is worst-case hard for* $\mathrm{R^*TISP}(n^c, \frac{a}{b}\log n)$ *given all sufficiently long auxiliary inputs.*

4.1 SearchBP*L = SearchL implies GapMcKSP is hard

We first observe that a random function is a targeted-PRG against R*TISP$(n, \log n)$; Goldreich [Gol11] proved this results against any deterministic (bounded) polynomial-time algorithms, and [LP22] noted that his proof extends also to deal with probabilistic polynomial-time algorithms. We here note that the same proof also directly extends to log space bounded algorithms—in fact, the class of algorithms does not matter for the proof as long as it is a uniform class. We defer the proof to Appendix B.

Claim 4.2. *Let $\varepsilon > 0$ be an arbitrary constant, and $\sigma \in \{0,1\}^n$. Let $a, b = O(1)$ be arbitrary.*

Then, a random function $f \colon \{0,1\}^{10 \log n} \to \{0,1\}^n$ is an ε targeted-PRG against all $M \in \text{R}^*\text{TISP}(n^a, b \log n) \cap \{0,1\}^{\log\log\log n}$ that decides the instance σ, w.h.p:

$$\Pr_f \left[\forall M : \left| \mathop{\mathbb{E}}_{\substack{s \in_R \{0,1\}^{10\log n}, \\ y \in_R \{0,1\}^{n^a}}} [M[\sigma, f(s), y]] - \mathop{\mathbb{E}}_{\substack{r \in_R \{0,1\}^{n}, \\ y \in_R \{0,1\}^{n^a}}} [M[\sigma, r, y]] \right| \geq \varepsilon \right] \leq 2^{-n}.$$

Let us define the search-problem of finding targeted-PRGs.

Definition 4.1. *Let $\mathcal{C}$ be a collection of TMs. The relation* Find-PRG$_{\mathcal{C}}[\alpha, \beta]$ *is defined as follows. Let $\sigma \in \{0,1\}^n$, and let $G_\sigma \colon \{0,1\}^{O(\log n)} \to \{0,1\}^n$ be an arbitrary function.*

1. $(\sigma, G_\sigma) \in$ Yes *if for all $M \in \{0,1\}^{\log\log\log n} \cap \mathcal{C}$,*

$$\left| \mathop{\mathbb{E}}_{x \in_R \{0,1\}^{O(\log n)}} [M[\sigma, G_\sigma(x)]] - \mathop{\mathbb{E}}_{r \in_R \{0,1\}^{n}} [M[\sigma, r]] \right| \leq \alpha,$$

2. $(\sigma, G_\sigma) \in$ No *if there exists an $M \in \{0,1\}^{\log\log\log n} \cap \mathcal{C}$ such that,*

$$\left| \mathop{\mathbb{E}}_{x \in_R \{0,1\}^{O(\log n)}} [M[\sigma, G_\sigma(x)]] - \mathop{\mathbb{E}}_{r \in_R \{0,1\}^{n}} [M[\sigma, r]] \right| \geq \beta.$$

If $\mathcal{C}$ consists of probabilistic machines, then all the probabilities are also over the random coins of M.

Observe that Claim 4.2 essentially implies that to find a good targeted-PRG, one can simply print a random function. The next claim shows that also *verifying* whether a given function, is a good one, is possible in logspace given 2-way access to the random and the candidate function. The proof follows using the same approach as [Gol11], and verifies the log-space efficiency of solver.

Claim 4.3. *For every constants $a, b = O(1)$,*

$$\text{Find-PRG}_{\text{R}^*\text{TISP}(n^a, b\log n)}[.1, .2] \in \textbf{SearchBP}^*\textbf{L}.$$

Proof. We describe suitable P*LM solution Finder and Verifier F, V. The finder simply prints a random input, whereas the verifier simply verifies (using its 2-way access to solution), if it is good one.

Algorithm 1: Low-space algorithms for $\textsc{Find-PRG}_{\mathrm{R^*TISP}(n^a, b\log n)}$

Input: $\sigma \in \{0,1\}^n$

Randomness (multiple read): $G_\sigma \colon \{0,1\}^{10\log n} \to \{0,1\}^n$ and $r' \in \{0,1\}^{\mathrm{poly}(n)}$

Function $F[\sigma, G_\sigma]$**:**

  print G_σ

Function $V[\sigma, G_\sigma, r']$**:**

  for $M \in \{0,1\}^{\log\log\log n} \cap \mathrm{R^*TISP}(n^a, b\log n)$ **do**

    Let p_M be an estimation, using $\mathrm{poly}(n)$ many $y \in_R \{0,1\}^n$ and $r \in_R \{0,1\}^{n^a}$, of

$$\mathop{\mathbb{E}}_{\substack{y\in_R\{0,1\}^n,\\ r\in_R\{0,1\}^{n^a}}} [M[\sigma, y, r]]$$

    Let q_M be an estimation, using $\mathrm{poly}(n)$ many $r \in \{0,1\}^{n^a}$, of

$$\mathop{\mathbb{E}}_{\substack{s\in_R\{0,1\}^{3a\log n},\\ r\in\{0,1\}^{n^a}}} [M[\sigma, G_\sigma(s), r]]$$

    if $|p_M - q_M| \geq 0.2 - \frac{1}{2n}$ **then return** 0

  return 1

To enumerate over $\mathrm{R^*TISP}(n^a, b\log n)$ machines, we verify during the simulation of $M[\sigma, y, r]$ that M uses at most $\leq b\log n$ space and runs for $\leq n^a$ steps; otherwise, we skip to the next $M \in \{0,1\}^{\log\log\log n}$.

We now claim that both F, V are P*LM:

1. F simply prints its randomness tape, so it is clearly a P*LM.
2. Consider V. First observe it uses $|r'| = \mathrm{poly}(n)$ random bits, as each iteration requires $\mathrm{poly}(|r|\cdot|y|) \leq \mathrm{poly}(n)$ many random bits to estimate p_M, q_M, and there are $\leq 2^{\log\log\log n} \leq \log\log n$ iterations.

   Since V has has two-way access its randomness and candidate solution G_σ, it can be implemented to consume at most $O(\log n)$ space, by recalling G_σ and random strings from the input tape for proper simulations of $M[\sigma, G_\sigma(s), r]$ or $M[\sigma, y, r]$.

   Each simulation requires only logspace and polynomial time overhead, which in turn concludes that V is indeed a P*LM.

As for the correctness, for every fixed σ, by Claim 4.2, $F[\sigma, \cdot]$ finds good PRG w.h.p:

$$\Pr_{G_\sigma}[F[\sigma, G_\sigma] \in \textsc{Yes}] = \Pr_{G_\sigma}[G_\sigma \text{ is 0.1 PRG}] \geq 1 - 2^{-n}.$$

Now consider $G_\sigma \in \textsc{Yes}$. Observe that V verifies G_σ correctly at least when all the estimations p_M, q_M were accurate up to error $1/n$; by using polynomially large enough number of sampling of r, y, the Hoeffding bound implies that each of which is accurate with probability $\geq 1 - 2^{-n}$. Thus by the union bound, V verifies correctly with overwhelming probability:

$$\Pr_{G_\sigma, r'}\left[V[\sigma, G_\sigma, r'] = 1 \,\middle|\, G_\sigma \text{ is 0.1 PRG}\right] \geq 1 - O(\log\log n)\cdot 2^{-n} \geq 1 - 2^{-0.99n}.$$

On the other hand, if $G_\sigma \in \textsc{No}$, namely it is not an 0.02 targeted-PRG, then V verifies badly only if some p_M or q_M were approximated badly; this happens with negligible probability:

$$\Pr_{G_\sigma, r'}\left[V[\sigma, G_\sigma, r'] = 1 \,\middle|\, G_\sigma \in \textsc{No}\right] \leq 2^{-n}.$$

□

Finally, assuming that **SearchBP*L** can be fully derandomized, we can argue that GapMcKSP is suitably hard for (sufficiently bounded) P*LM.

Claim 4.4. *For every large enough constant a, there exists $b = O(1)$ so the following holds.*

If **SearchBP*L** = **SearchL**, *then* $\mathsf{GapMcKSP}[a\log n, n-1]$ *is worst-case hard for* $\mathrm{R^*TISP}(n, \frac{a}{b}\log n)$, *given any sufficiently long auxiliary input.*

Proof. Let b be a constant TBD.

By Claim 4.3, the search problem $\textsc{Find-PRG}_{\mathrm{R^*TISP}(n,\frac{a}{b}\log n)}[.1,.2]$ belongs to **SearchBP*L**, and so the hypothesis implies it further belongs to **SearchL**.

By Claim 4.2, for every σ and constants a, b, there exists some G_σ such that $(\sigma, G_\sigma) \in \textsc{Yes}$, namely the relation is non-empty.

Combined together with the fact the relation in **SearchL**, implies that there exists a DLM, that on input $\sigma \in \{0,1\}^n$, prints the truthtable of a function

$$G\colon \{0,1\}^n \times \{0,1\}^{O(\log n)} \to \{0,1\}^n,$$

which is of length $\mathrm{poly}(n)$, such that for all $M \in \mathrm{R^*TISP}(n, \frac{a}{b}\log n) \cap \{0,1\}^{\log\log\log n}$, it holds

$$\left| \mathop{\mathbb{E}}_{\substack{x\in_R\{0,1\}^{O(\log n)}\\ y\in_R\{0,1\}^n}} [M[\sigma, G_\sigma(x), y]] - \mathop{\mathbb{E}}_{\substack{r\in_R\{0,1\}^n\\ y\in_R\{0,1\}^n}} [M[\sigma, r, y]] \right| \le 0.1,$$

Furthermore, the algorithms prints it in space $O(\frac{a}{b}\log n)$. Put differently, this implies a targeted-PRG computable in that space.

Let $\sigma \in \{0,1\}^n$ be sufficiently long target, and abbreviate $G_\sigma(x) \stackrel{def}{=} G(\sigma, x)$. Observe that given σ, G_σ has constant length description; thus by the PRG definition, for any seed x,

$$\begin{aligned}\mathsf{KS}(G_\sigma(x) \,|\, \sigma) &\le |G_\sigma| + |x| + s[G_\sigma(x)] \\ &= O\Big(\frac{a}{b}\log n\Big).\end{aligned}$$

Set $b = O(1)$ such that the above is bounded by $\le a\log n$; thus

$$\max_x \mathsf{KS}(G_\sigma(x) \,|\, \sigma) \le a\log n.$$

Assume, by way of contradiction, that there is a $\mathrm{R^*TISP}(n, \frac{a}{b}\log n)$ distinguisher A for GapMcKSP, given the auxiliary input σ:

$$\begin{aligned}\forall (z \mid \sigma) \in \textsc{Yes}, \quad \mathop{\mathbb{E}}_r [A[z,\sigma,r]] \ge 0.9,\\ \forall (z \mid \sigma) \in \textsc{No}, \quad \mathop{\mathbb{E}}_r [A[z,\sigma,r]] \le 0.1.\end{aligned}$$

We now show A succeeds to distinguish G_σ, namely it distinguishes pseudorandom inputs from random ones, with high probability.

Let $y = G_\sigma(x)$ be a pseudorandom input (for any seed x). By b's definition, $(\sigma, G_\sigma) \in \textsc{Yes}$, so,

$$\mathop{\mathbb{E}}_{x,r} [A[G_\sigma(x), \sigma, r]] \ge 0.9.$$

Consider random inputs $y \in_R \{0,1\}^n$. As every deterministic machine Π produces one string, the number of strings y with KS complexity $\leq n-2$ is at most

$$\begin{aligned}|\{y \in \{0,1\}^n \mid \mathsf{KS}(y \mid \sigma) \leq n-2\}| &\leq |\{\Pi \in \{0,1\}^n \mid |\Pi| + s[\Pi(\sigma)] \leq n-2\}| \\ &\leq |\{\Pi \in \{0,1\}^n \mid |\Pi| \leq n-2\}| \\ &\leq 2^{n-1},\end{aligned}$$

the fraction of random y's such that $(y,\sigma) \notin \text{No}$, namely having KS complexity $\leq n-2$ is at most,

$$\begin{aligned}\Pr_{y \in_R \{0,1\}^n}[(y,\sigma) \notin \text{No}] &= \Pr_{y \in_R \{0,1\}^n}[\mathsf{KS}(y \mid \sigma) \leq n-2] \\ &\leq \frac{2^{n-1}}{2^n} \\ &= 0.5,\end{aligned}$$

so A would reject random strings with high probability:

$$\begin{aligned}\mathop{\mathbb{E}}_{y \in_R \{0,1\}^n, r}[A[y,\sigma,r]] &\leq \Pr_{y \in_R \{0,1\}^n}[\mathsf{KS}(y \mid \sigma) \leq n-2] \cdot 1 + 1 \cdot \mathop{\mathbb{E}}_{y \in_R \{0,1\}^n, r}[A[y,\sigma,r] \mid \mathsf{KS}(y \mid \sigma) > n-2] \\ &\leq 0.5 + \mathop{\mathbb{E}}_{y \in_R \{0,1\}^n, r}[A[y,\sigma,r] \mid (y,\sigma) \in \text{No}] \\ &\leq 0.6.\end{aligned}$$

It follows that, for every sufficiently long target σ, A distinguishes G_σ with probability ≥ 0.3, a contradiction. □

4.2 Hardness of GapMcKSP implies SearchBP*L = SearchL

The main claim of this section is:

Claim 4.5. *For every large enough constant a, there exists $b, c = O(1)$ so the following holds.*

If $\mathsf{GapMcKSP}[a \log n, n-1]$ is worst-case hard for $\mathrm{R}^*\mathrm{TISP}(n^c, \frac{a}{b}\log n)$ given all sufficiently long auxiliary inputs, then **SearchBP*L = SearchL**.

Proof. Combine Claims 4.6 and 4.7 and Corollary 4.12 (see below). □

In high level, we first construct a targeted-HSG against $\mathrm{TISP}(n, \log n)$, based on hardness of $\mathsf{GapMcKSP}$ against $\mathrm{R}^*\mathrm{TISP}(n^c, \frac{a}{b}\log n)$; we then show how to conclude **SearchBP*L = SearchL**.

4.2.1 Hardness to target HSG

Claim 4.6. *Let $\varepsilon > 1/n^{O(1)}$. For every large enough constant a, there exists $b, c = O(1)$ so the following holds.*

If $\mathsf{GapMcKSP}[a \log n, n-1]$ is worst-case hard for $\mathrm{R}^*\mathrm{TISP}(n^c, \frac{a}{b}\log n)$ given all sufficiently long auxiliary inputs, then there exists an ε target HSG $H\colon \{0,1\}^{\mathrm{poly}(n)} \times \{0,1\}^{O(\log n)} \to \{0,1\}^n$ against $\mathrm{TISP}(n^\gamma, \log n)$ for all sufficiently long target, computable in space $O(\log n)$, for some constant $\gamma > 0$.

Proof. Let $\mathrm{DT}^f\colon \{0,1\}^{O(\log n)} \to \{0,1\}^n$ be the construction from Theorem 3.2 using parameter $m \stackrel{def}{=} n^{\Theta(1)}$. Let $\gamma > 0$ be the constant satisfying $n = m^\gamma$.

Given a target $\sigma \in \{0,1\}^m$, denote $H_\sigma(\cdot) \stackrel{def}{=} H(\sigma, \cdot)$; we define the HSG

$$H_\sigma\colon \left(\bigcup_{\ell \in [a \log m]} \{0,1\}^\ell\right) \times \{0,1\}^{O(\log m)} \to \{0,1\}^n$$

as

$$H_\sigma(\Pi, x) \stackrel{def}{=} \mathrm{DT}^{\Pi(\sigma)}(x),$$

where $\Pi(\sigma) \in \{0,1\}^m$ is the output of the TM represented by Π on input σ that uses at most $a \log m$ space; if the output is shorter we pad it by 0's, and if its longer, we truncate it to the first m bits.

Space Complexity $H_\sigma(\Pi, x)$ is computable in logspace because so does DT (Theorem 3.2) and $\Pi(\sigma)$ (the space complexity overhead of the simulation is linear).

Reduction Suppose H is not a target HSG for all sufficiently long target; thus there is some $\mathrm{TISP}(m^\gamma, \log m)$ distinguisher A, that on input length m distinguishes H, when simulated on infinitely many instances $\sigma \in \{0,1\}^m$:

$$\begin{aligned} \mathop{\mathbb{E}}_{r}[A[\sigma, r]] &\geq \varepsilon, \\ \mathop{\mathbb{E}}_{(\Pi,x)}[A[\sigma, H_\sigma(\Pi, x)]] &= 0. \end{aligned} \tag{1}$$

Observe that $m^\gamma = n$ and hence $A[\sigma, \cdot]$ runs in time n, and H_σ indeed outputs n bits.

Let $\mathcal{A}, \mathcal{R}$ be the Advice and Reconstruction P*LM from Theorem 3.2.

Consider the following distinguisher $B(z \mid \sigma)$ for GapMcKSP on input length $m = |z| = |\sigma|$:

1. Compute the advice $adv \stackrel{def}{=} \mathcal{A}^{z, A[\sigma,\cdot]}[1^m]$, by associating z with a function $\{0,1\}^{\log m} \to \{0,1\}$.

   By $A[\sigma, \cdot]$ we mean the adversary program that relies on the fixed instance σ.

2. Return 1 iff

$$\forall k \in \{0,1\}^{\log m}, \quad \mathcal{R}^{A[\sigma,\cdot]}[1^m, adv, k] = z_k \tag{2}$$

   and $|adv| \leq |z|^{2/3}$ and everything was computed in $O(\log m)$ space.

   We emphasize that for every k, B compute on the fly the advice adv to evaluate $\mathcal{R}$. This only increase the space complexity by additional $O(\log m)$ factor.

Complexity Since $\mathcal{A}[1^m], \mathcal{R}[1^m, \cdot]$ are simulatable by P*LM (at worst; proven in Theorem 3.2), and $A[\sigma, \cdot]$ is a DLM, by composition of space bounded machines (Lemma 3.1), it follows that B is also P*LM.

Moreover, observe that the input-length of $\mathcal{A}, \mathcal{R}$ is m, and hence their space complexity $O(\log m)$ is *independent* of the constant a which controlled the seed length of H (namely the length of the chosen TM, whose output was considered as the hard function in DT). Thus, if a is large enough, there exists $b, c = O(1)$ such that B is simulatable by

$$\mathrm{R^*TISP}\left(m^c, \frac{a}{b} \log m\right).$$

We emphasize that B computes everything on the fly; in particular, the advice adv is recomputed over and over and thus does not blows up B's space complexity.

Correctness Let $\sigma \in \{0,1\}^m$ be a distinguishable instance. We claim that B decides GapMcKSP, given the auxiliary input σ, for every z of length $|z| = m$.

Let $(z \mid \sigma) \in \textsc{No}$. We claim that the algorithm B would never accept such inputs. This follows because, by definition, every program Π that produces z, is either $> 0.5(m-1)$ bit long, or evaluable in $> 0.5(m-1)$ bits of work-space. Thus B would always reject:

$$\mathop{\mathbb{E}}_{r}[B[z, \sigma, r]] = 0.$$

Let $(z \mid \sigma) \in \textsc{Yes}$. Thus, there is some Π that produces $\Pi(\sigma) \mapsto z$ and

$$|\Pi| + s[\Pi(\sigma)] \leq a \log m,$$

and so by definition of H_σ, that includes all the machines of description length $\leq a \log m$ that are evaluable in space $\leq a \log m$, for every x it holds

$$\mathrm{DT}^z(x) \in \mathrm{Im}(H_\sigma(\cdot, x)).$$

In particular, by Equation (1), A would distinguish $\mathrm{DT}^z(\cdot)$ from random string.

Thus, by Theorem 3.2, the Advice procedure $\mathcal{A}^{z,A[\sigma,\cdot]}$ would output with probability $\geq 2/3$ a "good advice", such that $\mathcal{R}^{A[\sigma,\cdot]}$ would reconstruct z with probability 1, namely Equation (2) would hold. By Theorem 3.2, the advice is relatively short:

$$|adv| \leq |z|^{2/3},$$

so since indeed $\mathcal{A}, \mathcal{R}$ are computable in logspace, B would output 1 w.h.p.:

$$\mathop{\mathbb{E}}_{r}[B[z, \sigma, r]] \geq \Pr_{r}\left[\mathcal{A}^{z,A[\sigma,\cdot]}[1^m, r] \text{ succeeds}\right] \geq \frac{2}{3}.$$

This contradicts the hardness of $\mathsf{GapMcKSP}$ given all sufficiently long auxiliary input σ. □

Observe that for every constant $\gamma > 0$, a targeted-HSG against $\mathrm{TISP}(n^\gamma, \log n)$ can be extended, using padding argument, to be against DLM, namely $\mathrm{TISP}(\mathrm{poly}(n), O(\log n))$ for arbitrary large poly-time and logarithmic space machines.

Claim 4.7. *Let $m = m(n), s = s(n), d = d(n), \varepsilon(n) = \varepsilon$ be functions of n. Let $\gamma > 0$ be a constant.*

If there exists an ε target HSG $H\colon \{0,1\}^m \times \{0,1\}^s \to \{0,1\}^n$ against $\mathrm{TISP}(n^\gamma, \log n)$, computable in space d, then there exists an $\varepsilon(\mathrm{poly}(n))$ target HSG $H'\colon \{0,1\}^m \times \{0,1\}^{s(\mathrm{poly}(n))} \to \{0,1\}^n$ against DLM that is computable in space $d(\mathrm{poly}(n))$.

In particular, if $s(n), d(n) = O(\log n)$, then H' is logspace computable and has logarithmic seed length.

Proof. Fix an instance $\sigma \in \{0,1\}^n$, and let D be an arbitrary DLM such that

$$\mathop{\mathbb{E}}_{r}[D[\sigma, r]] \geq \varepsilon,$$

where it uses $\ell \stackrel{def}{=} |r| = \mathrm{poly}(n)$ random bits (the bound is due to D's polynomial running time bound).

Let $m \stackrel{def}{=} \ell^{1/\gamma} = \mathrm{poly}(n)$. Define the machine D' as follows:

$$D'[\tau', y] \stackrel{def}{=} \begin{cases} D[\tau, y] & |\tau'| = n + m \wedge \tau' = \tau \circ 1^m, \\ 0 & \text{otherwise.} \end{cases}$$

Observe that D' decides the "padded" language that D decides: namely, for every string y, it accepts targets of the form $\tau \circ 1^m$ iff D accepts τ. Moreover, since D' can verify in $O(\log(n+m)) = O(\log n)$ space whether its input τ' is of the form $\tau \circ 1^m$, it follows that D' is a DLM.

Now observe that $D'[\sigma \circ 1^m]$, which has input length $n + m > m = \ell^{1/\gamma}$, consumes only $\ell \leq (n+m)^\gamma$ random bits, and

$$\mathop{\mathbb{E}}_{r \in_R \{0,1\}^\ell}\left[D'[\sigma \circ 1^m, r]\right] = \mathop{\mathbb{E}}_{r \in_R \{0,1\}^\ell}[D[\sigma, r]] \geq \varepsilon,$$

in which case, by definition of D' and H,

$$\mathop{\mathbb{E}}_{x \in_R \{0,1\}^s}[D[\sigma, H(\sigma \circ 1^m, x)]] = \mathop{\mathbb{E}}_{x \in_R \{0,1\}^s}\left[D'[\sigma \circ 1^m, H(\sigma \circ 1^m, x)]\right] > 0.$$

Now define the HSG $H'(\sigma,\cdot) \stackrel{def}{=} H(\sigma \circ 1^m,\cdot)$, namely generating strings according to the "padded" instance. Hence, by the above equation, H' is guaranteed to hit $D[\sigma,\cdot] = D'[\sigma \circ 1^m,\cdot]$. Observe that H' computable in space $s(n+m) = s(\text{poly}(n))$, has seed length $d(n+m) = d(\text{poly}(n))$, and the HSG's new threshold error becomes $\varepsilon(\text{poly}(n))$. □

Such an targeted-HSG, as in the hypothesis of Claim 4.7, implies $\mathbf{R^*L} \subseteq \mathbf{L}$ by enumerating the HSG seeds and taking the majority vote. Furthermore, using the space-variant reduction of [BF99] from Theorem A.1, that shows $\mathbf{R^*L} = \mathbf{L} \implies \mathbf{BP^*L} = \mathbf{L}$, this even implies that a good targeted-HSG implies 2-sided error derandomization:

Corollary 4.8. *For every large enough constant a, there exists $b, c = O(1)$ so the following holds.*

If $\mathsf{GapMcKSP}[a\log n, n-1]$ *is worst-case hard for* $\text{R}^*\text{TISP}(n^c, \frac{a}{b}\log n)$ *given all sufficiently long auxiliary inputs, then* $\mathbf{BP^*L} \subseteq \mathbf{L}$.

4.2.2 Target HSG against DLM implies derandomization of SearchBP*L

This section is devoted to prove Corollary 4.12, which exhibits derandomization of search problems $\mathbf{SearchBP^*L} = \mathbf{SearchL}$, given an optimal target HSG against DLM.

Claim 4.9. *If* $\mathbf{BP^*L} = \mathbf{L}$, *then* $\mathbf{SearchBP^*L} \subseteq \mathbf{SearchR^*L}$.

Proof. Given a relation $R \in \mathbf{SearchBP^*L}$, let F, V be P*LM which are R's solution Finder and Verifier TMs, respectively; by definition,

$$\forall x,y, \quad \Pr_r[V[x,y,r] = 1(y \in R(x))] \geq \frac{2}{3}.$$

Thus V decides the language $R = \{(x,y)\} \in \mathbf{BP^*L}$. By the hypothesis, $R \in \mathbf{L}$, and so there exists a DLM verifier V' for R:

$$\forall x,y: \quad V'[x,y] = 1(y \in R(x)),$$

which implies $R \in \mathbf{SearchR^*L}$ (see Definitions 3.9 and 3.10). □

Claim 4.10. *If there exists a logspace targeted-HSG against* DLM *for all sufficiently long targets, then* $\mathbf{SearchR^*L} \subseteq \mathbf{SearchL}$.

Proof. Let $R \in \mathbf{SearchR^*L}$, and let F, V be its P*LM finder and DLM verifier, respectively.

Define the language $R' \subseteq \{0,1\}^*$ as

$$R' \stackrel{def}{=} \{x \in \{0,1\}^* \mid R(x) \neq \emptyset\}.$$

Claim 4.11. $R' \in \mathbf{R^*L}$.

Proof. Consider the machine $M[x,r]$ that acts as follows:

$$M[x,r] \stackrel{def}{=} V[x, F[x,r]].$$

Observe that M is P*LM, as it leverages the 2-way access to randomness and composition of space bounded machines (Lemma 3.1).

As for the correctness, by definition of F, V:

$$\begin{aligned} x \in R' &\implies R(x) \neq \emptyset \implies \Pr_r[F[x,r] \in R(x)] \geq \frac{2}{3} \implies \Pr_r[V[x,F[x,r]] = 1] \geq \frac{2}{3}, \\ x \notin R' &\implies R(x) = \emptyset \implies \Pr_r[F[x,r] \in R(x)] = 0 \implies \Pr_r[V[x,F[x,r]] = 1] = 0. \end{aligned}$$

□

Let n be sufficiently large such that there exists a target HSG for that n.
Let $x \in \{0,1\}^n$, and consider the following search algorithm $F'[x]$ for R:

1. Let $H_x \colon \{0,1\}^{O(\log n)} \to \{0,1\}^{\mathrm{poly}(n)}$ be a targeted HSG against DLM for target x.
2. Loop all $s \in \{0,1\}^{O(\log n)}$:
   (a) If $M[x, H_x(s)] = 1$: Print $F[x, H_x(s)]$.

Then, by definition of the HSG,

$$\begin{aligned} R' \in \mathbf{R}^*\mathbf{L} &\iff \Pr_r[M[x,r]=1] \geq \frac{2}{3} \\ &\iff \exists s \in \{0,1\}^{O(\log n)}, \quad M[x, H_x(s)] = 1 \\ &\iff \exists s \in \{0,1\}^{O(\log n)}, \quad V[x, F[x, H_x(s)]] = 1 \\ &\iff \exists s \in \{0,1\}^{O(\log n)}, \quad F[x, H_x(s)] \in R(x), \end{aligned}$$

so $F'[x]$ would always print a good solution in deterministic logspace, and hence $R \in \mathbf{SearchL}$. □

Corollary 4.12. *If there exists a* 0.1 *targeted-HSG* $H \colon \{0,1\}^n \times \{0,1\}^{O(\log n)} \to \{0,1\}^n$ *against* DLM *for all sufficiently long targets, that is computable in space* $O(\log n)$*, then* $\mathbf{SearchBP}^*\mathbf{L} = \mathbf{SearchL}$.

Proof. The existence of logspace targeted-HSG implies that $\mathbf{R}^*\mathbf{L} = \mathbf{L}$ by enumeration of the HSG. By Theorem A.1, we further conclude $\mathbf{BP}^*\mathbf{L} = \mathbf{L}$; by Claim 4.9, $\mathbf{SearchBP}^*\mathbf{L} \subseteq \mathbf{SearchR}^*\mathbf{L}$; then, by Claim 4.10, we can use the HSG to derandomize $\mathbf{SearchR}^*\mathbf{L} \subseteq \mathbf{SearchL}$. □

5 SearchBP*L vs Leakage-resilient

5.1 Definitions

Let $\ell \colon \mathbb{N} \to \mathbb{N}$. A probabilistic TM L is said to have a **bounded output length** ℓ if, on every sufficiently long input x, it outputs at most $\ell(|x|)$ bits, regardless of its randomness r:

$$\max_r |L[x,r]| \leq \ell(|x|).$$

Definition 5.1. *Let* $\mathsf{Adv}, \mathsf{Leak}$ *be collections of TMs; let* $\ell \colon \mathbb{N} \to \mathbb{N}$.

A function $f \colon \{0,1\}^n \to \{0,1\}^m$ *is an* $(\mathsf{Adv}, \mathsf{Leak}, \ell)$-almost-all-inputs-leakage resilience hard*, if for all algorithms* $(A, L) \in \mathsf{Adv} \times \mathsf{Leak}$ *such that* L *has have bounded output length* ℓ*, the following holds.*

For every sufficiently long $x \in \{0,1\}^n$*,* A *fails to compute* $f(x)$ *with constant probability, even when given 2-way access to the leakage of* $L[x, f(x)]$*; that is,*

$$\Pr_{r,r'}\left[A[x, L[x, f(x), r'], r] = f(x)\right] \leq \frac{2}{3}.$$

Similarly, we define f *to be* $(\mathsf{Adv}, \mathsf{Leak}, \ell)$-almost-all-inputs-leakage resilience locally hard *similarly as leakage resilience hard function with the following modifications: first,* A *gets an additional input* $i \in [n]$ *and is asked to compute only* $f(x)_i \in \{0,1\}$*; second, the hardness assumption now is that for every sufficiently long* x *there exists* i *so* $A[x, i, \cdot]$ *fails to compute* $f(x)_i$ *w.h.p:*

$$\exists i \in [n], \quad \Pr_{r,r'}\left[A[x, i, L[x, f(x), r'], r] = f(x)_i\right] \leq \frac{2}{3}.$$

The main theorem of this section is:

Theorem 5.1. *There exists constants $a, c = O(1)$ so for every constant $b > a$ the following holds.*
Define the collection of TMs $\mathsf{Adv}, \mathsf{Leak}$ *as*

$$\mathsf{Leak} \stackrel{def}{=} \mathrm{R}^*\mathrm{TISP}(n^c, a\log n)$$
$$\mathsf{Adv} \stackrel{def}{=} \mathrm{TISP}(n^c, a\log n).$$

Then $\mathbf{SearchBP^*L} = \mathbf{SearchL}$ *if and only if there exists an* $(\mathsf{Adv}, \mathsf{Leak}, n^{2/3})$*-almost-all-inputs leakage resilient locally hard function* $f\colon \{0,1\}^n \to \{0,1\}^n$*, computable in space* $b\log n$*.*

Proof. By combining Claim 5.3 and Theorem 5.7 with the observation that every $(\mathcal{A}, \mathcal{L}, \ell)$ leakage resilient hard function is also a leakage resilient *locally* hard one (for almost all inputs). □

5.2 Leakage resilience implies SearchBP*L = SearchL

We first show, based on leakage resilience function, how to derive targeted-PRG against DLM; using that, we derandomize $\mathbf{SearchBP^*L}$.

Theorem 5.2. *There exists constants $a, c = O(1)$ so for every constant $b > a$ the following holds.*
Define the collections of TMs $\mathsf{Adv}, \mathsf{Leak}$ *as*

$$\mathsf{Leak} \stackrel{def}{=} \mathrm{R}^*\mathrm{TISP}(n^c, a\log n)$$
$$\mathsf{Adv} \stackrel{def}{=} \mathrm{TISP}(n^c, a\log n).$$

Let $\varepsilon > 1/n^{O(1)}$. If there exists an $(\mathsf{Adv}, \mathsf{Leak}, n^{2/3})$*-almost-all-inputs leakage resilient locally hard function* $f\colon \{0,1\}^n \to \{0,1\}^n$*, computable in space* $b\log n$*, then there exists an* ε *targeted-PRG* $G\colon \{0,1\}^{\mathrm{poly}(n)} \times \{0,1\}^{O(\log n)} \to \{0,1\}^n$ *against* DLM*, computable in space* $O(\log n)$*, for all sufficiently long targets.*

For simplicity we focus on $\mathrm{TISP}(n^\gamma, \log n)$ adversaries, for some constant $\gamma > 0$, so we need to generate n^γ pseudo-random bits. The general case follows by padding argument on the target.

Proof. Let $\mathrm{DT}^f\colon \{0,1\}^{O(\log n)} \to \{0,1\}^n$ be the construction from Theorem 3.2 using parameter $m \stackrel{def}{=} n^{\Theta(1)}$. Let $\gamma > 0$ be the constant satisfying $n = m^\gamma$. From hereon we use the leakage function on input length m.

Let us describe the PRG with parameters

$$G\colon \{0,1\}^m \times \{0,1\}^{O(\log n)} \to \{0,1\}^n.$$

Given a target $\sigma \in \{0,1\}^m$, we abbreviate $G_\sigma(x) \stackrel{def}{=} G(\sigma, x)$. Define the targeted PRG as

$$G_\sigma(x) \stackrel{def}{=} \mathrm{DT}^{f(\sigma)}(x),$$

where the string $f(\sigma) \in \{0,1\}^m$ is viewed as a function $\{0,1\}^{\log m} \to \{0,1\}$.

Space complexity. Using composition of space bounded-machines, $G_\sigma(x)$ is computable in $O(\log n)$ space because both $f(\sigma)$ and $\mathrm{DT}^{f(\sigma)}$ are computable in space $O(\log m) = O(\log n)$ (see Theorem 3.2).

Correctness Suppose, by way of contradiction, that G is not an ε target PRG for all sufficiently long targets; so there is a $\mathrm{TISP}(m^\gamma, \log m)$ distinguisher D, that on input length m, and infinitely many targets σ such that

$$\mathop{\mathbb{E}}_{x}[D[\sigma, G_\sigma(x)]] - \mathop{\mathbb{E}}_{r}[D[\sigma, r]] \geq \varepsilon.$$

Observe that $m^\gamma = n$ and hence $D[\sigma, \cdot]$ runs in time n, and G_σ indeed outputs n bits.

Let $\mathcal{A}, \mathcal{R}$ be the Advice and Reconstruction P*LM from Theorem 3.2. We now construct leakage and adversary algorithms that contradicts the hardness of f.

The leakage $L[\sigma, f(\sigma)]$ returns the advice defined as

$$L[\sigma, f(\sigma)] \stackrel{def}{=} \mathcal{A}^{f(\sigma), D[\sigma,\cdot]}[1^m].$$

By $D[\sigma, \cdot]$ we mean the distinguisher program that relies on the fixed instance σ.

By Theorem 3.2, the advice has bounded length $\leq m^{2/3}$, so indeed L produce short enough leak:

$$|L[\sigma, f(\sigma)]| = \left|\mathcal{A}^{f(\sigma), D[\sigma,\cdot]}[1^m]\right| \leq m^{2/3}.$$

By composing space-bounded machines, since $D, \mathcal{A}[1^m]$ are computable by P*LM (at worst), so does L; thus for some constants $a, c = O(1)$,

$$L \in \mathrm{R}^*\mathrm{TISP}(m^c, a \log m) = \mathsf{Leak}.$$

The deterministic adversary $A[\sigma, i, adv]$**,** on input σ, index $i \in [m]$ and leakage adv, returns the evaluation

$$A[\sigma, i, adv] \stackrel{def}{=} \mathcal{R}^{D[\sigma,\cdot]}(1^m, adv, i).$$

Observe that A access adv in 2-way manner, and it does not count as part of its space-tape. By composing space-bounded machines, since $\mathcal{R}[1^m, adv, \cdot], D$ are DLM, so does A; so for large enough constants $a, c = O(1)$,

$$A \in \mathrm{TISP}(m^c, a \log m) = \mathsf{Adv}.$$

Correctness for sufficiently long σ, by Theorem 3.2, with probability $\geq 2/3$, the leakage algorithm L would produce a good leakage adv, such that $\mathcal{R}^{D[\sigma,\cdot]}[1^m, adv, \cdot]$ would fully reconstruct the string $f(\sigma) \in \{0,1\}^m$ (associated with the function $\{0,1\}^{\log m} \to \{0,1\}$); in that case, the adversary A would succeed with probability 1:

$$\forall i \in [m], \quad \Pr_r[A[\sigma, i, L[\sigma, f(\sigma), r]] = f(\sigma)_i] = \Pr_r\left[\mathcal{A}^{f(\sigma), D[\sigma,\cdot]}[1^m, r] \text{ succeeds}\right] \geq \frac{2}{3},$$

which contradicts the leakage-resilience local hardness of f. □

Building on our targeted-PRG against DLM, we can now derandomize **SearchBP*L**:

Claim 5.3. *There exists constants $a, c = O(1)$ so for every constant $b > a$ the following holds.*

Define the collections of TMs $\mathsf{Adv}, \mathsf{Leak}$ *as*

$$\mathsf{Leak} \stackrel{def}{=} \mathrm{R}^*\mathrm{TISP}(n^c, a \log n)$$
$$\mathsf{Adv} \stackrel{def}{=} \mathrm{TISP}(n^c, a \log n).$$

If there exists an $(\mathsf{Adv}, \mathsf{Leak}, n^{2/3})$*-almost-all-inputs leakage resilient locally hard function* $f\colon \{0,1\}^n \to \{0,1\}^n$*, computable in space* $b \log n$*, then* **SearchBP*L = SearchL**.

Proof. Observe that Theorem 5.2 implies, by the hypothesis, a targeted PRG: there is 1-1 correspondence between targets of the PRG, and hard inputs of the function; since its almost-all-input hard, the generator is against all sufficiently long target.

The proof follows by combining it with Corollary 4.12, with the observation that a targeted PRG is by definition also a targeted-HSG. □

Finally, we observe that the space complexity of the given leakage-resilient function corresponds to the space complexity of our constructed targeted-PRG. This helps us to characterize *partial* derandomization of **SearchBP*L**.

Corollary 5.4. *Let $\gamma > 0$. There exists constants $a, c = O(1)$ so for every constant $b > a$ the following holds.*

Define the collections of TMs $\mathsf{Adv}, \mathsf{Leak}$ *as*

$$\mathsf{Leak} \stackrel{def}{=} \mathrm{R}^*\mathrm{TISP}(n^c, a\log n)$$
$$\mathsf{Adv} \stackrel{def}{=} \mathrm{TISP}(n^c, a\log n).$$

If there exists an $(\mathsf{Adv}, \mathsf{Leak}, n^{2/3})$-almost-all-inputs leakage resilient locally hard function $f\colon \{0,1\}^n \to \{0,1\}^n$, computable in space $b\log^{1+\gamma} n$, then **SearchBP*L** $=$ **SearchL**$^{1+\gamma}$.

Proof. Essentially as Theorem 5.2, except that now the construction G is computable in space $b\log^{1+\gamma}$, which in turn blows-up the derandomization complexity. □

5.3 SearchBP*L = SearchL implies leakage-resilience hard function

This section adapts [LP23] proof to the logspace regime. In high level, it shows that a random function is leakage-resilience, and that it possible to verify it by a P*LM; this in turn implies that the search problem of finding leakage-resilient hard function belongs to **SearchBP*L**. Then, we leverage the hypothesis to compute it deterministically.

Definition 5.2. *Let* $\mathsf{Adv}, \mathsf{Leak}$ *be collections of TMs, and $\ell\colon \mathbb{N} \to \mathbb{N}$; let $\alpha, \beta > 0$.*

The relation $\textsc{Find Leakage Resilient Function}_{\mathsf{Adv},\mathsf{Leak},\ell}[\alpha,\beta]$ *consists of pairs $\{(x,y) \mid x,y \in \{0,1\}^n\}$ where, for every x, y of length n,*

1. *$(x,y) \in$ Yes if for all $(A,L) \in \mathsf{Adv} \times \mathsf{Leak}$ that have short description $A, L \in \{0,1\}^{\log n}$ it holds*

$$\Pr_{r,r'}\left[|L[x,y,r]| \le \ell(|x|) \wedge A[x, L[x,y,r], r'] = y\right] \le \alpha.$$

2. *$(x,y) \in$ No if there exists $(A,L) \in \mathsf{Adv} \times \mathsf{Leak}$ that have short description $A, L \in \{0,1\}^{\log n}$ and*

$$\Pr_{r,r'}\left[|L[x,y,r]| \le \ell(|x|) \wedge A[x, L[x,y,r], r'] = y\right] \ge \beta.$$

Fix large enough constants $a, c = O(1)$. From hereon we use the collection of TMs and leakage length

$$\begin{aligned} \mathsf{Leak} &\stackrel{def}{=} \mathrm{R}^*\mathrm{TISP}(n^c, a\log n), \\ \mathsf{Adv} &\stackrel{def}{=} \mathrm{TISP}(n^c, a\log n), \\ \ell(n) &\stackrel{def}{=} n^{2/3}. \end{aligned} \tag{3}$$

As observed by [LP23], a random function is an $(\mathsf{Adv}, \mathsf{Leak}, \ell)$ leakage resilient with high probability.

Claim 5.5 ([LP23]). *For any fixed TM A, and for all $x \in \{0,1\}^n$,*

$$\Pr_{y \in_R \{0,1\}^n}\left[\exists t \in \{0,1\}^\ell : \Pr_r\left[A[x,t,r] = y\right] \geq \varepsilon\right] \leq \frac{2^{\ell+1}/\varepsilon}{2^n}.$$

We provide the proof for completeness:

Proof. For every fixed t, the algorithm $A[x,t,\cdot]$ can produce at most $\leq 1/\varepsilon$ outputs with probability $\geq \varepsilon$. By the union bound over all "leakages" $t \in \{0,1\}^\ell$, there are at most $2^{\ell+1}/\varepsilon$ such "frequent" outputs of $A[x,t,\cdot]$; the string y is bad if its one them. This occur with probability that is proportional to their density in $\{0,1\}^n$. □

Next, we observe that the search-problem of finding leakage-resilient hard function belongs to **SearchBP*L**, as was shown by [LP23]; the proof verifies that everything is computable in low space, given 2-way access to the random tape. The proof is essentially similar to Claim 4.3.

Claim 5.6. Find Leakage Resilient Function$_{\mathsf{AdvLeak},\ell}[0.1, 0.2] \in$ **SearchBP*L**.

Proof. Consider the following algorithms F, V that finds and verify solutions for the search problem.

Algorithm 2: Low-space algorithms for Find Leakage Resilient Function$_{\mathsf{Adv},\mathsf{Leak},\ell}$

Input: $x \in \{0,1\}^n$
Randomness (multiple read): $y \in \{0,1\}^n$ and $r'' \in \{0,1\}^{\mathrm{poly}(n)}$
Function $F[x,y]$**:**
  print y
Function $V[x,y,r'']$**:**
  for $(A,L) \in \mathsf{Adv} \times \mathsf{Leak}$ *such that* $A, L \in \{0,1\}^{\log n}$ **do**
    Let $p_{A,L}$ be an estimation, using $\mathrm{poly}(n)$ many r, r', of

$$\Pr_{r,r'}\left[|L[x,y,r]| \leq \ell \wedge A[x, L[x,y,r], r'] = y\right].$$

    if $p_{A,L} \geq 0.2 - \frac{1}{n}$ **then return** 0
  return 1

To enumerate over $\mathsf{Adv}, \mathsf{Leak}$ machines, we verify during the simulation of $L[x,y,r], A[x, L[x,y,r], r']$ that they obey the time and space restrictions of $\mathsf{Adv}, \mathsf{Leak}$; if not, we continue to the next pair.

We now claim that F, V are P*LM:

1. F simply prints its random tape, so it is clearly a P*LM.

2. Consider V. First observe that for every fixed A, L, each estimation $p_{A,L}$ requires $\mathrm{poly}(|r| \cdot |r'|) \leq \mathrm{poly}(n)$ many random bits, as A, L consumes $\leq \mathrm{poly}(n)$ random bits (see the definition of $\mathcal{A}, \mathcal{L}$ from Equation (3)). So since there are $\leq 2^{2\log n} \leq n^2$ iterations, V consumes $|r''| = \mathrm{poly}(n)$ many random bits.

   Second, as V can access its randomness and candidate solution y in 2-way, he can be implemented to consume at most $O(\log n)$ space, by recalling y and random strings from the random tape, for proper simulations of $L[x,y,r]$ and $A[x, L[x,y,r], r']$.

   Each simulation requires only logspace and polynomial time overhead, which in turn concludes that V is a P*LM.

As for the correctness, observe that for every x, by Claim 5.5, $F[x,\cdot]$ finds good values y w.h.p:

$$\begin{aligned}\Pr_{y\in_R\{0,1\}^n}[F[x,y]\notin \textsc{Yes}] &\leq \Pr_{y\in_R\{0,1\}^n}\left[\exists t\in\{0,1\}^\ell\colon \Pr_r[A[x,t,r]=y]\geq 0.1\right]\\ &\leq \frac{10\cdot 2^{\ell+1}}{2^n}\\ &\leq 2^{-n/2},\end{aligned}$$

where we set $\ell = n^{2/3}$.

Now consider $(x,y)\in \textsc{Yes}$. Observe that V verifies y correctly at least when all the estimations $p_{A,L}$ were accurate up to $1/n$ error; assuming it uses polynomially large enough number of sampling of r, r', by the Hoeffding bound, each of which is accurate with probability $\geq 1-2^{-n}$. Thus by the union bound over all pairs, V verifies correctly with overwhelming probability:

$$\Pr_{y,r}[V[x,y,r]=1 \mid (x,y)\in \textsc{Yes}]\geq 1-n^2\cdot 2^{-n}\geq 1-2^{-0.99n}.$$

On the other hand, if $(x,y)\in \textsc{No}$, namely y is "easy" to compute given some efficiently computable leakage, then V verifies badly only if some $p_{A,L}$ was approximated badly; this happens with negligible probability:

$$\Pr_{y,r}[V[x,y,r]=1 \mid (x,y)\in \textsc{No}]\leq 2^{-n}.$$

□

Thus we conclude,

Theorem 5.7. *For every constant $a, c = O(1)$, there exists a constant $b > a$, so the following holds.*

Define the collection of TMs $\mathsf{Adv}, \mathsf{Leak}$ *as*

$$\begin{aligned}\mathsf{Leak} &\stackrel{def}{=} \mathrm{R}^*\mathrm{TISP}(n^c, a\log n)\\ \mathsf{Adv} &\stackrel{def}{=} \mathrm{TISP}(n^c, a\log n).\end{aligned}$$

If $\mathbf{SearchBP^*L} = \mathbf{SearchL}$*, then there exists an explicit* $(\mathsf{Adv},\mathsf{Leak},n^{2/3})$*-almost-all-inputs leakage resilient function $f\colon\{0,1\}^n\to\{0,1\}^n$, computable in space $b\log n$.*

Proof. By Claim 5.6, the search problem $\textsc{Find Leakage Resilient Function}_{\mathsf{Adv},\mathsf{Leak},n^{2/3}}[0.1,0.2]$ belongs to $\mathbf{SearchBP^*L}$, and thus by the assumption it also belongs to $\mathbf{SearchL}$.

Let $x\in\{0,1\}^n$ be arbitrary. Due to Claim 5.5, there exists some y in the relation with x, namely the relation is non-empty:

$$(x,y)\in \textsc{Find Leakage Resilient Function}_{\mathsf{Adv},\mathsf{Leak},n^{2/3}}[0.1,0.2].$$

Thus, combined with the fact the relation in $\mathbf{SearchL}$, implies there exists a DLM, that on input x, prints y, in space $b\log n > ac\log n$, where $b = O(1)$. This algorithm, by definition, evaluates the function f, which is leakage resilient for all all sufficiently long input. □

Finally, we observe that the above theorem generalizes in the natural way, when beginning instead with partial derandomization assumption; this in turn, affects the space complexity of the hard function.

Corollary 5.8. *Let $\gamma > 0$. For every constants $a, c = O(1)$, there exists a constant $b > a$, so the following holds.*

Define the collection of TMs $\mathsf{Adv}, \mathsf{Leak}$ *as*

$$\mathsf{Leak} \stackrel{def}{=} \mathrm{R}^*\mathrm{TISP}(n^c, a\log n)$$
$$\mathsf{Adv} \stackrel{def}{=} \mathrm{TISP}(n^c, a\log n).$$

If $\mathbf{SearchBP^*L} \subseteq \mathbf{SearchL}^{1+\gamma}$*, then there exists an explicit* $(\mathsf{Adv}, \mathsf{Leak}, n^{2/3})$*-almost-all-inputs leakage resilient function* $f\colon \{0,1\}^n \to \{0,1\}^n$*, computable in space* $b\log^{1+\gamma} n$*.*

6 Discussion

Our focus was on characterizing the power of randomness, given a multiple access to it, while restricting the computation to logspace. Liu and Pass [LP22; LP23] considered also characterization of derandomizing **MA** and "effective dereandomization" of **BPP**; it seems that our techniques are extendable to capture the analogue results in our settings, but we omit the details in this manuscript.

Acknowledgment We thank Muli Safra and Itamar Rot for enlightening discussions.

References

[AB09] Sanjeev Arora and Boaz Barak. *Computational Complexity - A Modern Approach*. Cambridge University Press, 2009, pp. I–XXIV, 1–579. ISBN: 978-0-521-42426-4.

[ABK+06] Eric Allender, Harry Buhrman, Michal Kouckỳ, Dieter Van Melkebeek, and Detlef Ronneburger. "Power from random strings". In: *SIAM Journal on Computing* 35.6 (2006), pp. 1467–1493.

[AKRR11] Eric Allender, Michal Kouckỳ, Detlef Ronneburger, and Sambuddha Roy. "The pervasive reach of resource-bounded Kolmogorov complexity in computational complexity theory". In: *Journal of Computer and System Sciences* 77.1 (2011), pp. 14–40.

[BF99] Harry Buhrman and Lance Fortnow. "One-sided versus two-sided error in probabilistic computation". In: *STACS 99: 16th Annual Symposium on Theoretical Aspects of Computer Science Trier, Germany, March 4–6, 1999 Proceedings 16*. Springer. 1999, pp. 100–109.

[BFNW91] László Babai, Lance Fortnow, Noam Nisan, and Avi Wigderson. "BPP has subexponential time simulations unless EXPTIME has publishable proofs". In: *1991 Proceedings of the Sixth Annual Structure in Complexity Theory Conference*. IEEE Computer Society. 1991, pp. 213–214.

[BM84] Manuel Blum and Silvio Micali. "How to Generate Cryptographically Strong Sequences of Pseudorandom Bits". In: *SIAM Journal on Computing* 13.4 (1984), pp. 850–864.

[CT22] Lijie Chen and Roei Tell. "Hardness vs randomness, revised: Uniform, non-black-box, and instance-wise". In: *2021 IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE. 2022, pp. 125–136.

[CTW23] Lijie Chen, Roei Tell, and Ryan Williams. "Derandomization vs Refutation: A Unified Framework for Characterizing Derandomization". In: *Electron. Colloquium Comput. Complex.* TR23-105 (2023). ECCC: TR23-105. URL: https://eccc.weizmann.ac.il/report/2023/105.

[Cha69] Gregory J. Chaitin. "On the Simplicity and Speed of Programs for Computing Infinite Sets of Natural Numbers". In: *J. ACM* 16.3 (1969), pp. 407–422.

[DT23] Dean Doron and Roei Tell. "Derandomization with minimal memory footprint". In: *38th Computational Complexity Conference (CCC 2023)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik. 2023.

[Gol11] Oded Goldreich. "In a world of P=BPP". In: *Studies in Complexity and Cryptography. Miscellanea on the Interplay between Randomness and Computation*. Springer, 2011, pp. 191–232.

[Har83] J. Hartmanis. "Generalized Kolmogorov complexity and the structure of feasible computations". In: *24th Annual Symposium on Foundations of Computer Science (sfcs 1983)*. 1983, pp. 439–445. DOI: 10.1109/SFCS.1983.21.

[Hir20] Shuichi Hirahara. "Non-disjoint promise problems from meta-computational view of pseudorandom generator constructions". In: *35th Computational Complexity Conference (CCC 2020)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik. 2020.

[Hoz21] William Hoza. "Better Pseudodistributions and Derandomization for Space-Bounded Computation." In: *Electron. Colloquium Comput. Complex.* Vol. 28. 2021, p. 48.

[IKW02] Russell Impagliazzo, Valentine Kabanets, and Avi Wigderson. "In search of an easy witness: Exponential time vs. probabilistic polynomial time". In: *Journal of Computer and System Sciences* 65.4 (2002), pp. 672–694.

[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*. 1997, pp. 220–229.

[KM02] Adam R. Klivans and Dieter van Melkebeek. "Graph nonisomorphism has subexponential size proofs unless the polynomial-time hierarchy collapses". In: *SIAM Journal on Computing* 31.5 (2002), pp. 1501–1526.

[Ko86] Ker-I Ko. "On the Notion of Infinite Pseudorandom Sequences". In: *Theor. Comput. Sci.* 48.3 (1986), pp. 9–33. DOI: 10.1016/0304-3975(86)90081-2. URL: https://doi.org/10.1016/0304-3975(86)90081-2.

[Kol68] A. N. Kolmogorov. "Three approaches to the quantitative definition of information". In: *International Journal of Computer Mathematics* 2.1-4 (1968), pp. 157–168.

[Kor22] Oliver Korten. "Derandomization from time-space tradeoffs". In: *37th Computational Complexity Conference (CCC 2022)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik. 2022.

[LP22] Yanyi Liu and Rafael Pass. "Characterizing Derandomization Through Hardness of Levin-Kolmogorov Complexity". In: *37th Computational Complexity Conference, CCC 2022, July 20-23, 2022, Philadelphia, PA, USA*. Ed. by Shachar Lovett. Vol. 234. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022, 35:1–35:17. DOI: 10.4230/LIPIcs.CCC.2022.35. URL: https://doi.org/10.4230/LIPIcs.CCC.2022.35.

[LP23] Yanyi Liu and Rafael Pass. "Leakage-resilient hardness vs randomness". In: *38th Computational Complexity Conference (CCC 2023)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik. 2023.

[Lev84] Leonid A Levin. *Universal search problems*. 1984.

[Lev85] Leonid A Levin. "One-way functions and pseudorandom generators". In: *Proceedings of the seventeenth annual ACM symposium on Theory of computing*. 1985, pp. 363–365.

[MPV15] Debasis Mandal, Aduri Pavan, and NV Vinodchandran. "On probabilistic space-bounded machines with multiple access to random tape". In: *International Symposium on Mathematical Foundations of Computer Science*. Springer. 2015, pp. 459–471.

[NW94] Noam Nisan and Avi Wigderson. "Hardness vs. randomness". In: *Journal of Computer and System Sciences* 49.2 (1994), pp. 149–167.

[Nis91] Noam Nisan. "Pseudorandom bits for constant depth circuits". In: *Combinatorica* 11 (1991), pp. 63–70.

[Nis93] Noam Nisan. "On read once vs. multiple access to randomness in logspace". In: *Theoretical Computer Science* 107.1 (1993), pp. 135–144.

[PRZ23] Edward Pyne, Ran Raz, and Wei Zhan. "Certified Hardness vs. Randomness for Log-Space". In: *Electron. Colloquium Comput. Complex.* TR23-040 (2023). ECCC: TR23-040. URL: https://eccc.weizmann.ac.il/report/2023/040/.

[Rei08] Omer Reingold. "Undirected connectivity in log-space". In: *Journal of the ACM (JACM)* 55.4 (2008), pp. 1–24.

[STV99] Madhu Sudan, Luca Trevisan, and Salil Vadhan. "Pseudorandom generators without the XOR lemma". In: *Proceedings of the thirty-first annual ACM symposium on Theory of computing*. 1999, pp. 537–546.

[SZ99] Michael E. Saks and Shiyu Zhou. "$\mathsf{BP_HSPACE}(S) \subseteq \mathsf{DSPACE}(S^{2/3})$". In: *Journal of Computer and System Sceinces* 58.2 (1999), pp. 376–403.

[Sav70] Walter J. Savitch. "Relationships Between Nondeterministic and Deterministic Tape Complexities". In: *Journal of Computer and System Sciences* 4.2 (Apr. 1970), pp. 177–192. ISSN: 0022-0000.

[Sol64] R.J. Solomonoff. "A formal theory of inductive inference. Part I". In: *Information and Control* 7.1 (1964), pp. 1 –22. ISSN: 0019-9958. DOI: https://doi.org/10.1016/S0019-9958(64)90223-2.

[Tra84] Boris A Trakhtenbrot. "A survey of Russian approaches to perebor (brute-force searches) algorithms". In: *Annals of the History of Computing* 6.4 (1984), pp. 384–400.

[Vad12] Salil Vadhan. "Pseudorandomness". In: *Foundations and Trends® in Theoretical Computer Science* 7.1–3 (2012), pp. 1–336.

[Yao82] Andrew C Yao. "Theory and application of trapdoor functions". In: *Foundations of Computer Science, 1982. SFCS'08. 23rd Annual Symposium on*. IEEE. 1982, pp. 80–91.

A Derandomizing $\mathbf{R}^*\mathbf{L}$ implies derandomization of $\mathbf{BP}^*\mathbf{L}$

We show that the proof of $\mathbf{RP} = \mathbf{P}$ implies $\mathbf{BPP} = \mathbf{P}$ due to Buhrman and Fortnow [BF99] works just the same for bounded-space complexity classes with 2-way access to randomness.

Theorem A.1. *If* $\mathbf{R}^*\mathbf{L} \subseteq \mathbf{L}$*, then* $\mathbf{BP}^*\mathbf{L} \subseteq \mathbf{L}$.

For simplicity we prove that $\mathbf{BP}^*\mathbf{TISP}(n, \log n) \subseteq \mathbf{L}$.

Proof. Let $L \in \mathbf{BP}^*\mathbf{TISP}(n, \log n)$ be any language decidable by some machine with error $\leq 1/3$; by naive amplification, let M be the P*LM that decides L with error $\leq 2^{-n}$, whose running time is $t = \text{poly}(n)$; thus it uses $\leq t$ random coins.

Define the promise language $A = (\textsc{Yes}, \textsc{No})$ as follows; for every $(x, r_1, \ldots, r_t) \in \{0,1\}^{n+t^2}$:

1. $(x, \vec{r}) \in \textsc{Yes}$ if
$$\Pr_{w \in_R \{0,1\}^t} [\exists i \in [t], \quad M[x, w \oplus r_i] = 1] = 1.$$
2. $(x, \vec{r}) \in \textsc{No}$ if
$$\Pr_{w \in_R \{0,1\}^t} [\exists i \in [t], \quad M[x, w \oplus r_i] = 1] \leq 0.5.$$

Claim A.2. $A \in \mathbf{R}^*\mathbf{L}$.

Proof. Consider the machine $M'[x, \vec{r}]$ that samples t many w_j, and returns

$$M'[x, \vec{r}, \vec{w}] \stackrel{def}{=} \bigwedge_{j \in [t]} 1\big(\exists i_j \in [t], \quad M[x, w_j \oplus r_{i_j}] = 1\big).$$

Correctness If $(x, \vec{r}) \in \textsc{Yes}$, then M' would always accept.

If $(x, \vec{r}) \in \textsc{No}$, by the Chernoff bound M' would reject with probability $\geq 1 - 2^{-t}$.

Complexity M' runs in space $O(\log t + s[M]) = O(\log n)$ by referring to the w_j's from the random tape; its running time is $\mathrm{poly}(t) = \mathrm{poly}(n)$. □

Thus, by the hypothesis A can be derandomized; the next step is to solve L using A:

Claim A.3. $L \in \mathbf{R}^*\mathbf{L}^A$.

Proof. Consider the machine M' that samples $r_1, \ldots, r_t$ and answers $1((x, r_1, \ldots, r_t) \in A)$ according to its oracle. Observe that M' uses space $O(\log t + s[M]) = O(\log n)$ by referring to the random tape instead of holdings the r_i's; its running time, dominated by t, is polynomial; hence M' is a P*LM.

Let $x \in L$. Then, since the randomness is of length $|w| = |r_i| \leq t$,

$$\begin{aligned}
\Pr_{r_1,\ldots,r_t}[(x, \vec{r}) \notin \textsc{Yes}] &= \Pr_{r_1,\ldots,r_t}\Big[\Pr_w[\exists i \in [t], \quad M[x, w \oplus r_i] = 1] < 1\Big] \\
&= \Pr_{r_1,\ldots,r_t}[\exists w \forall i, \quad M[x, w \oplus r_i] = 0] \\
&\leq \sum_w \Pr_{r_1,\ldots,r_t}[\forall i, \quad M[x, w \oplus r_i] = 0] \\
&= \sum_w \prod_{i \in [t]} \Pr_{r_i}[M[x, w \oplus r_i] = 0] \\
&\leq \sum_w (2^{-n})^t \\
&\leq 2^t \cdot 2^{-nt} \\
&\leq 2^{-n},
\end{aligned}$$

and thus M' would accepts with high probability:

$$\Pr_{r_1,\ldots,r_t}\big[M'[x, \vec{r}] = 1\big] \geq \Pr_{r_1,\ldots,r_t}[(x, \vec{r}) \in \textsc{Yes}] \geq 1 - 2^{-n}.$$

Suppose now $x \notin L$. Then, for every $r_1, \ldots, r_t$,

$$\Pr_w[\exists i, \quad M[x, w \oplus r_i] = 1] \leq \sum_i \Pr_w[M[x, w \oplus r_i] = 1] \leq t \cdot 2^{-n} \ll 0.5,$$

since $t = \mathrm{poly}(n)$; so any fixation of r_i leads to No and hence rejection by M':

$$\Pr_{r_1,\ldots,r_t}\big[M'[x, \vec{r}] = 0\big] \geq \Pr_{r_1,\ldots,r_t}[(x, \vec{r}) \in \textsc{No}] = 1.$$

□

It now follows that $L \in \mathbf{R}^*\mathbf{L}^{\mathbf{R}^*\mathbf{L}} \subseteq \mathbf{L}$. □

B Random objects are good

Proof of Claim 4.2. Fix a machine M and target $\sigma \in \{0,1\}^n$. Since M runs in time n^a, its randomness y is bounded by length $\leq n^a$.

For every seed $s \in \{0,1\}^{10\log n}$, denote by X_s the indicator random variable, over f, y, whether M accept on $\sigma, f(s), y$:

$$X_s \stackrel{def}{=} 1(M[\sigma, f(s), y] = 1)$$

Observe that for every seed s, since f is a random function,

$$\mathop{\mathbb{E}}_{f,y}[X_s] = \mathop{\mathbb{E}}_{f,y}[M[\sigma, f(s), y]] = \mathop{\mathbb{E}}_{r,y}[M[\sigma, r, y]].$$

Thus, by Hoeffding's bound, the probability that f is not a PRG for $M[\sigma,\cdot,\cdot]$, is at most,

$$\Pr_f\left[\left|\frac{1}{n^{10}}\sum_{s\in\{0,1\}^{10\log n}} X_s - \mathop{\mathbb{E}}_{\substack{r\in_R\{0,1\}^n \\ y\in_R\{0,1\}^{n^a}}}[M[\sigma,r,y]]\right| \geq \varepsilon\right] \leq 2^{-\frac{(n^{10}\varepsilon)^2}{n^{10}}} = 2^{-n^{10}\varepsilon^2} \leq 2^{-n^2},$$

Hence the probability that f does not fools some $M \in \{0,1\}^{\log\log\log n}$ is at most:

$$\Pr_f[\exists M \text{ s.t. } f \text{ does not } \varepsilon\text{-fools } M[\sigma,\cdot,\cdot]] \leq \sum_M 2^{-n^2} \leq \log\log n \cdot 2^{-n^2} \leq 2^{-n}.$$

□

C Equivalence to [DT23]

Definition C.1 ([DT23, Defintion 1.1])**.** *We say that $P \in \{0,1\}^*$ is an S-space compressed version of $f \in \{0,1\}^*$ if P is a description, of length $\sqrt{|f|}$, of a TM M that satisfies the following: On input $i \in [|f|]$, the machine M runs in space $S(|i|)$ and outputs f_i.*

The following is based on [DT23, Assumption 3]; the only modification is the reference to 2-way access to the random tape and the polynomial time bound.

Assumption C.1. *For a sufficiently large constant C, there exists a function $f\colon \{0,1\}^* \to \{0,1\}^*$ mapping n bits to n^2 bits, that is computable in space $(C+1)\cdot\log n$, and satisfies the following. For every 2-way probabilistic algorithm R running in space $C\log n + O(\log n)$ and polynomial time $\mathrm{poly}(n)$, there are at most finitely many $x \in \{0,1\}^*$ for which*

$$\Pr_r[R[x,r] \text{ prints a } (C\log n)\text{-space compressed version of } f(x)] \geq \frac{2}{3}.$$

Claim C.1. *There exists constants $a, b, d, C = O(1)$ so the following holds. Let*

$$\mathsf{Adv} \stackrel{def}{=} \mathrm{TISP}(n^b, d\cdot C\log n),$$
$$\mathsf{Leak} \stackrel{def}{=} \mathrm{R^*TISP}(n^b, C\log n + a\log n).$$

Then, Assumption C.1 holds if and only if there exists an $(\mathsf{Adv}, \mathsf{Leak}, \sqrt{n})$-almost-all-inputs-leakage resilience locally hard function $f\colon \{0,1\}^n \to \{0,1\}^n$ which is computable in space $(C+3)\log n$ space, up to constantly shorter leakage.

Proof. First we assume that Assumption C.1 require length-preserving function; this assumption will be resolved later using a simple padding.

Observe that the space complexity of f in both definitions is the same. It remains to show that either hardness implies the other one. Let U be a UTM.

1. Assume f is a $(\mathsf{Adv}, \mathsf{Leak}, \sqrt{n})$-almost-all-inputs leakage resilience locally hard function. Let $d \coloneqq d_{uni}$ be the space complexity overhead imposed by U.[6] Observe that every adversary R as in Assumption C.1 belongs to $R \in \mathsf{Leak}$, and the UTM U belongs to $U \in \mathsf{Adv}$. Then, if $x \in \{0,1\}^n$ is a hard input in the sense of leakage resilience, it is in particular hard for the pair of leakage-adversary (R, U), where R is arbitrary from Assumption C.1. Thus x is also hard input for f in the sense of Assumption C.1.

2. Assume f satisfies Assumption C.1. We claim that f is a $(\mathsf{Leak}, \mathsf{Adv}, \sqrt{n} - O(1))$-almost-all-inputs leakage resilience locally hard function, for the constant $d \coloneqq 1$.

   Otherwise, there exists $(R, A) \in (\mathsf{Leak} \times \mathsf{Adv})$ and infinitely many inputs $x \in \{0,1\}^n$ that violates the leakage-resilience locally hardness of f. Consider the algorithm R', that on input x, computes $y \stackrel{def}{=} R(x, f(x))$, and prints the description A with hard-coded input y. To simulate R properly on $f(x)$, R' computes f.

   Since $|A| = O(1)$ (because a TM has a constant length description), it follows that $|R'(x)| \leq |R(x, f(x))| - O(1) + O(1) \leq \sqrt{n}$. Moreover, $R'(x)$ is computable in space

$$\leq (C \log n + a \log n) + (C+1) \log n + \underbrace{O(\log n)}_{\text{simulation's overhead}} \leq (C + O(a)) \log n.$$

   Thus, since R' is as in Assumption C.1, it follows that R' prints a $C \log n$ compressed version of $f(x)$. Since there is 1-1 correspondence in the x's, we have contradicted Assumption C.1.

We now resolve the expansion property of f. Assuming $f \colon \{0,1\}^n \to \{0,1\}^n$ is length-preserving leakage-resilient hard function, and consider the function $f'(x) = f(x) \circ 1^n$. We claim that f' is hard as in Assumption C.1: otherwise, there exists an adversary R, in the sense of Assumption C.1, that compresses $f'(x)$. It could be utilized as before to violate the leakage-resistance property of f, by simply modifying the adversary/leakage algorithms to compute f' instead of f (by concatenating the padding).

On the other hand, if $f' \colon \{0,1\}^n \to \{0,1\}^{n^2}$ is hard as in Assumption C.1, consider the function

$$f(x \circ 1^{n^2-n}) \stackrel{def}{=} f'(x).$$

Consider any adversary-leakage pair (A, L) attackers to f. Define the algorithm $L'(x, f'(x)) \stackrel{def}{=} L(x \circ 1^n, f'(x))$. It is clear that (A, L') is an adversary-leakage pair attackers to f, and thus could be used (as previously explained) to violate Assumption C.1.

In summary, both directions increase the space complexity of f by additional $\leq 2 \log n$ factor to account for the padding. □

[6] Specifically, d_{uni} is defined as follows. For every description of a deterministic TM $\Pi \in \{0,1\}^*$, that is computable in space $s(n)$ on inputs length n, the simulation of Π under U is computable in space $d_{uni} \cdot s(n)$.

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