One-Way Functions and Boundary Hardness of Randomized Time-Bounded Kolmogorov Complexity

Yanyi Liu
Cornell Tech, New York, NY, USA

Rafael Pass
Cornell Tech, Technion, TAU, New York, NY, USA

Abstract

We revisit the question of whether worst-case hardness of the time-bounded Kolmogorov complexity problem, $\mathsf{MINK}^{\mathsf{poly}}$ – that is, determining whether a string is "structured" (i.e., $K^t(x) < n-1$) or "random" (i.e., $K^{\mathsf{poly}(t)} \geq n-1$) – suffices to imply the existence of one-way functions (OWF). Liu-Pass (CRYPTO'25) recently showed that worst-case hardness of a *boundary* version of $\mathsf{MINK}^{\mathsf{poly}}$ – where, roughly speaking, the goal is to decide whether given an instance x, (a) x is K^{poly}-random (i.e., $K^{\mathsf{poly}(t)}(x) \geq n-1$), or just close to K^{poly}-random (i.e., $K^t(x) < n-1$ *but* $K^{\mathsf{poly}(t)} > n - \log n$) – characterizes OWF, but with either of the following caveats (1) considering a non-standard notion of *probabilistic* K^t, as opposed to the standard notion of K^t, or (2) assuming somewhat strong, and non-standard, derandomization assumptions.

In this paper, we present an alternative method for establishing their result which enables significantly weakening the caveats. First, we show that boundary hardness of the more standard *randomized* K^t problem suffices (where randomized $K^t(x)$ is defined just like $K^t(x)$ except that the program generating the string x may be randomized).

As a consequence of this result, we can provide a characterization also in terms of just "plain" K^t under the most standard derandomization assumption (used to derandomize just BPP into P) – namely $\mathsf{E} \not\subseteq \mathsf{ioSIZE}[2^{o(n)}]$.

Our proof relies on language compression schemes of Goldberg-Sipser (STOC'85); using the same technique, we also present the the first worst-case to average-case reduction for the *exact* $\mathsf{MINK}^{\mathsf{poly}}$ problem (under the same standard derandomization assumption), improving upon Hirahara's celebrated results (STOC'18, STOC'21) that only applied to a *gap* version of the $\mathsf{MINK}^{\mathsf{poly}}$ problem, referred to as $\mathsf{GapMINK}^{\mathsf{poly}}$, where the goal is to decide whether $K^t(x) \leq n - O(\log n)$) or $K^{\mathsf{poly}(t)}(x) \geq n-1$ and under the same derandomization assumption.

2012 ACM Subject Classification Theory of computation → Computational complexity and cryptography

Keywords and phrases One-way functions, Time-Bounded Kolmogorov Complexity, Worst-case to Average-case Reductions

Digital Object Identifier 10.4230/LIPIcs.ITCS.2026.97

Funding *Yanyi Liu*: Supported in part by NSF Award CNS 2149305.
Rafael Pass: Supported in part by NSF Award CNS 2149305, AFOSR Award FA9550-24-1-0267, ISF Award 2338/23 and ERC Advanced Grant KolmoCrypt - 101142322. 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, the AFOSR, the European Union or the European Research Council Executive Agency.

1 Introduction

A *one-way function* [5] (OWF) is a function f that can be efficiently computed (in polynomial time), yet no probabilistic polynomial-time (PPT) algorithm can invert f with inverse polynomial probability for infinitely many input lengths n. Whether one-way functions

exist is unequivocally the most important open problem in Cryptography (and arguably the most important open problem in the theory of computation, see e.g., [32]): OWFs are both necessary [27] and sufficient for many of the most central cryptographic primitives and protocols (e.g., pseudorandom generators [3, 13], pseudorandom functions [9], private-key encryption [10], digital signatures [49], commitment schemes [44], identification protocols [6], coin-flipping protocols [2], and more). These primitives and protocols are often referred to as *private-key primitives*, or "Minicrypt" primitives [26] as they exclude the notable task of public-key encryption [5, 48]. Additionally, as observed by Impagliazzo [11, 26], the existence of a OWF is equivalent to the existence of polynomial-time method for sampling hard *solved* instances for an NP language (i.e., hard instances together with their witnesses).

Worst-case Characterizations of OWFs. We here focus on the question of whether there exists a natural complexity-theoretic problem whose *worst-case* hardness characterizes OWFs. Recently, [33] presented an *average-case* characterization of OWFs through a natural computational problem – the *time-bounded Kolmogorov complexity problem* [31, 50, 30, 12]: Let the Kolmogorov complexity of a string x, denoted $K(x)$, be defined as the length of the shortest program that outputs x, and the $t(\cdot)$-bounded version, $K^t(x)$, be defined as the length of the shortest program that outputs the string x within time $t(|x|)$. While determining (or deciding for some particular threshold $s(\cdot)$) the Kolmogorov complexity of a string x is uncomputable, as surveyed by Trakhtenbrot [51], the problem of efficiently computing/deciding K^t-complexity (when t is a polynomial) predates the theory of NP-completeness and was studied in the Soviet Union since the 60s as a candidate for a problem that requires "brute-force search". Indeed, so far no non-trivial attacks are known in the uniform setting (and only very recently a non-trivial attack of circuit size roughly $2^{4n/5}$ was presented in the non-uniform setting [43, 18]).

The work of [33] showed that mild average-case hardness of computing K^t (or simply deciding whether it is large or small) w.r.t. the *uniform* distribution over instances, characterizes the existence of OWF. (Subsequently, several other average-case characterizations of OWF were obtained, e.g., [34, 47, 1, 35, 36].)

Recently, also worst-case characterizations of OWFs were obtained [37, 20], considering some variants of the time-bounded Kolmogorov complexity problem. In particular:

- [37] characterize OWFs through the problem of determining whether $K^t(x)$ is large or small, but restricting attention to instances x with very large *unbounded* Kolmogorov complexity (i.e., $K(x) > n - O(\log n)$), or alternatively, to strings x whose so-called "computational depth", $cd^t(x) = K^t(x) - K(x) < O(\log n)$, is small (whereby "restricting attention" means that we consider worst-case hardness of a promise problem that only considers those instances; that is, any efficient algorithm must fail on one of those instances in the promise).
- [20] provide a worst-case characterization of a variant of OWFs, referred to as infinitely-often OWFs, through the problem of "estimating the probability that a random time-bounded program outputs a certain string" (which can be shown to be related to the notion of probabilistic K^t), while restricting attention to instances satisfying an analog of small computational depth (with respect to this complexity notion).[1]

[1] An alternative type of worst-case characterization of OWFs was also obtained in [20], where it is shown that OWFs exists iff NP $\not\subseteq$ BPP and a certain "distributional"-K^t problem is NP-complete w.r.t. a certain type of (restricted) reductions. Note that simply worst-case hardness of the distributional K^t problem alone is not sufficient to get OWFs. Rather, the characterization is in terms of both a hardness assumption (NP $\not\subseteq$ BPP) combined with an feasibility assumption (the existence of a certain type of NP-completeness reduction).

The unappealing aspect of the above characterizations is that once we add the "conditioning", it becomes less clear what the intuitive interpretation of the problems is. On a technical level, the property that we condition on (i.e., computational depth being small, or Kolmogorov complexity being large) is not *decidable*.

Boundary-Hardness of Time-bounded Kolmogorov Complexity. Very recently, [39] presented an approach towards a more natural problems whose worst-case hardness may characterize OWFs. Ideally, we would like to show that the classic $\mathsf{MINK}^{\mathsf{poly}}$ problem – that is, the problem of simply determining whether a string is "structured" (i.e., $K^t(x) < n-1$) or "random" (i.e., $K^{\mathsf{poly}(t)} \geq n-1$) – suffices to imply the existence of one-way functions (OWF). [39] recently showed that worst-case hardness of a *boundary* version of $\mathsf{MINK}^{\mathsf{poly}}$ – where, roughly speaking, the goal is to decide whether given an instance x, deciding whether (a) x is K^{poly}-random (i.e., $K^{\mathsf{poly}(t)}(x) \geq n-1$), or just close to K^{poly}-random (i.e., $K^t(x) < n-1$ *but* $K^{\mathsf{poly}(t)} > n - \log n$) – characterizes OWF, but with either of the following caveats:

- Considering a non-standard notion of *probabilistic* K^t [8], as opposed to the standard notion of K^t.[2]
- Assuming somewhat strong, and non-standard, derandomization assumptions. In more detail, assuming that there exists a constant $\varepsilon > 0$ such that $\mathsf{E} \not\subseteq \mathsf{ioNTIME}[2^{kn}]/2^{\varepsilon n}$ for every $k \in \mathbb{N}$. This is a strengthening of the classic derandomization assumption that $\mathsf{E} \not\subseteq \mathsf{ioSIZE}[2^{o(n)}]$ (used to e.g. show that $\mathsf{BPP} = \mathsf{P}$ [45, 28]).

Our Results in a Nutshell. Our main results will improve upon the above result in two ways:

- Instead of showing an unconditional characterization in terms of *probabilistic* K^t, we will show that the same result holds with respect to the notion of *randomized* K^t [41, 46, 40]. Roughly speaking, randomized $K^t(x)$ – denoted $rK^t(x)$ – is defined as the length of the shortest *randomized* program that generates x with some probability δ (think of δ as $2/3$.) Note that despite the similarity in the names between probabilistic-K^t and randomized-K^t, there is a major conceptual difference. In $rK^t(x)$, the *same* program Π is supposed to generate x with high probability over the randomness string r (i.e., Π_r generates x with high probability over r) – this is arguably the most natural way of generalizing K^t to consider randomized programs; in contrast, for pK^t, we simply require that with high probability over the string r, there exists some "short" program Π such that Π_r generates x.
- When considering simply the boundary version of K^t, we are able to singificantly simplify (and weaken) the derandomization assumption and obtain a characterization under the most classic derandomization assumption that $\mathsf{E} \not\subseteq \mathsf{ioSIZE}[2^{o(n)}]$ (originally used to show that $\mathsf{BPP} = \mathsf{P}$ [45, 28]).

1.1 Our Results

To state our results more formally, let us first recall the standard $\mathsf{MINK}^{\mathsf{poly}}$ problem; the randomized version of the problem, $\mathsf{MINrK}^{\mathsf{poly}}$, is defined analogously (see Section 4). For any polynomials $t_1 < t_2$, let MINK^{t_1,t_2} denote the promise problem consisting of:

[2] Roughly speaking, for some constant δ (think of $\delta = 2/3$), $pK^t_\delta(x)$ is defined as the smallest ω such at with probability δ over the choice of a random string r, there exists a program of length at most ω that generates x if being provided r. One can think of this notion as time-bounded Kolmogorov complexity in the presence of a "Common Random String" (CRS), but note that program is allowed to depend on the CRS.

- YES: $x \in \{0,1\}^n$, $K^{t_1}(x) \leq n-2$.
- NO: $x \in \{0,1\}^n$, $K^{t_2}(x) \geq n-1$,

In essence, the goal is to distinguish between so-called *time-bounded Kolmogorov-random strings* (i.e., x s.t. $K^t(x) \geq n-1$) and those that are not. We say that $\mathsf{MINK}^{\mathsf{poly}} \notin \mathsf{ioBPP}$ if for all polynomials t_1, t_2, $t_2(n) \geq t_1(n) \geq 2n$, $\mathsf{MINK}^{t_1,t_2} \notin \mathsf{ioBPP}$.[3]

The *boundary* version (denoted by $\mathsf{boundary\text{-}MINK}^{t_1,t_2}$) is defined as the promise problem consisting of:

- YES: $x \in \{0,1\}^n$, $K^{t_1}(x) \leq n-2$ and $K^{t_2}(x) > n - \log n$.
- NO: $x \in \{0,1\}^n$, $K^{t_2}(x) \geq n-1$.

and $\mathsf{boundary\text{-}MINK}^{\mathsf{poly}}$ is analogously defined. In other words, the problem requires:

> *Deciding whether a string x is* **(a)** *time-bounded Kolmogorov random, or* **(b)** *just "near" time-bounded Kolmogorov random (i.e., having "intermediate" time-bounded Kolmogorov complexity.)*

We are now ready to state our main theorems:

▶ **Theorem 1.1.** *Assume there exists some constant ϵ such that $\mathsf{E} \not\subseteq \mathsf{ioSIZE}[2^{\epsilon n}]$. Then, OWFs exist if and only if $\mathsf{boundary\text{-}MINK}^{\mathsf{poly}} \notin \mathsf{ioBPP}$.*

▶ **Theorem 1.2.** *OWFs exist if and only if $\mathsf{boundary\text{-}MINrK}^{\mathsf{poly}} \notin \mathsf{ioBPP}$.*

Comparing our Derandomization Assumption to the one used in [39]. As mentioned, we are relying on the most classic derandomization assumption in the literature; that is, the existence of a function $f\colon \{0,1\}^n \to \{0,1\}$ computable in 2^{cn} time for some constant c, that cannot be computed in by circuits of size $2^{\epsilon n}$ for some constant ϵ. This is the assumption used by [28] to demonstrate that $\mathsf{BPP} = \mathsf{P}$.

This assumption was first strengthened by [29] to require hardness against non-deterministic circuits. Following [4], [39], considered an even stronger version, requiring the existence of c, ϵ such that for every k (i.e., we require a *hierarchy* of hard functions), there exists a function running in time $2^{c \cdot kn}$ that is secure against non-deterministic attackers with running time $2^{\epsilon kn}$ but still only $2^{\epsilon n}$ advice. Note that compared to this assumption, ours is weaker in two respects: (a) we no longer require non-deterministic hardness, and most importantly (b) we no longer require an infinite hierarchy of hard functions over a fixed input length.

Does Worst-case Hardness of just K^t Suffice? The above results still require worst-case of the *boundary* version of $\mathsf{MINK}^{\mathsf{poly}}$. As mentioned, ideally, we would like to base OWFs on worst-case hardness of just the $\mathsf{MINK}^{\mathsf{poly}}$ problem. Beyond being interesting in its own right, this question is motivated by the fact that in recent years, there has been great progress (see e.g. [21, 25, 22] [23, 36, 17, 24]) towards showing that problem is NP-complete. Notably, if OWFs can be based on the worst-case hardness of this problem, and the problem can be shown to be NP-complete, then we would base OWFs on just the assumption that $\mathsf{NP} \not\subseteq \mathsf{BPP}$ (a problem left open since the work by Diffie and Hellman [5], and referred to as the "holy-grail").

Intriguingly, a result by Hirahara [14] shows that if we were to consider a "gap" version of the K^t problem, referred to as $\mathsf{GapMINK}^{\mathsf{poly}}[s, s+n^\epsilon]$, where the goal is to distinguish between x such that $K^{t_1}(x) < s$ and $K^{t_2}(x) > s + n^\epsilon$, for all $t_2 = \mathsf{poly}(t_1)$, then worst-case

[3] Recall that ioBPP denotes the class of promise problems that admit a probabilistic polynomial time algorithm on infinitely many input lengths.

hardness of this problem implies what is referred to as *errorless* average-case hardness (w.r.t. deterministic algorithms). Roughly speaking, *errorless* average-case hardness refers to average-case hardness w.r.t. algorithms that either provide the right answer or $\perp$ (and $\perp$ should only be output with small probability). In other words, hardness holds w.r.t. algorithms that "know" when they fail. This is in contrast to the notion of *two-sided* error hardness where we allow the attacker to fail (without knowing it) on some small set of inputs. (Unfortunately, to apply the result of [33], we require the more standard notion of *two-sided* error average-case hardness, so this result is not helpful in terms of getting OWFs, at least given current machinery.) Hirahara [16], subsequently showed that under the assumption that $\mathsf{E} \not\subseteq \mathsf{ioSIZE}[2^{o(n)}]$, the same result can be established when the "gap" is just $\Omega(\log n)$, as opposed to n^{ϵ} (i.e., for $\mathsf{GapMINK}^{\mathsf{poly}}[s, s + O(\log n)]$).

As our final result, and using the same technique as the proof of our main theorem, we show how to strengthen Hirahara's result to directly apply to $\mathsf{MINK}^{\mathsf{poly}}$ (as opposed to the gap version of this problem).

▶ **Theorem 1.3.** *Assuming* $\mathsf{E} \not\subseteq \mathsf{ioSIZE}[2^{\varepsilon n}]$ *for some* $\varepsilon > 0$, $\mathsf{MINK}^{\mathsf{poly}} \notin \mathsf{BPP}$ *if and only if* $\mathsf{MINK}^{\mathsf{poly}} \notin \mathsf{AvgBPP}$ *(which in turn implies that* $\mathsf{NP} \not\subseteq \mathsf{AvgBPP}$*).*

As mentioned, Hirahara [14, 15] shows (under the same derandomization assumption) a worst-case to average-case reduction for a *gap* version of the $\mathsf{MINK}^{\mathsf{poly}}$ problem, where there is a gap of $\Omega(\log n)$ between the K^{poly} complexity of YES and NO instances (on top of the gap between the running times). Theorem 1.3 improves the results of [14, 15, 8] by removing the gap in the large threshold regime (i.e, when deciding whether K^{poly} is larger or smaller than $s(n) = n - O(1)$): As far as we know, Theorem 1.3 is the first worst-case to average-case reduction for simply the $\mathsf{MINK}^{\mathsf{poly}}$ problem (and not the $\mathsf{GapMINK}^{\mathsf{poly}}$ problem).

We mention, however, that while our proof technique is very different from the one in [14, 15], and is also significantly simpler, it has the disadvantage that it only works in the large threshold regime (i.e., $s(n) = n - O(1)$) , whereas Hirhara's technique also works for smaller thresholds $s(n)$ (but again, only for the gap version).

Additionally, if focusing on rK^t (as opposed to K^t) and only considering average-case hardness against deterministic polynomial-time algorithms (as in [14]), then we can show unconditionally obtain a worst-case to average-case reduction.

▶ **Theorem 1.4.** $\mathsf{MINK}^{\mathsf{poly}} \notin \mathsf{P}$ *if and only if* $\mathsf{MINrK}^{\mathsf{poly}} \notin \mathsf{AvgP}$ *(which in turn implies that* $\mathsf{NP} \not\subseteq \mathsf{AvgP}$*).*

Taken together, our results provide the appealing characterization of the worst-case hardness of the time-bounded Kolmogorov complexity problem (under standard derandomization assumption) provided in Figure 1.

1.2 Proof Overview

We proceed to a proof outline of our results. As mentioned, [39] obtained a characterization of OWFs through the boundary hardness of time-bounded Kolmogorov complexity, but considering a non-standard notion of probabilistic K^t, pK^t. We focus on explaining how to prove that the same result holds w.r.t. the notion of randomized K^t, rK^t. Note that the notion of pK^t was crucial in obtaining a OWF from the boundary hardness. So we will primarily discuss how to prove the "if" direction of Theorem 1.2. The converse direction will follow mostly from [39], using the techniques from [37], which passed through the notion of an "entropy-preserving PRG" constructed in [33] (and later improved in [37]). Our weakening of the derandomization assumption in Theorem 1.1 follows from the fact that we are able to deal with rK^t (as opposed to pK^t).

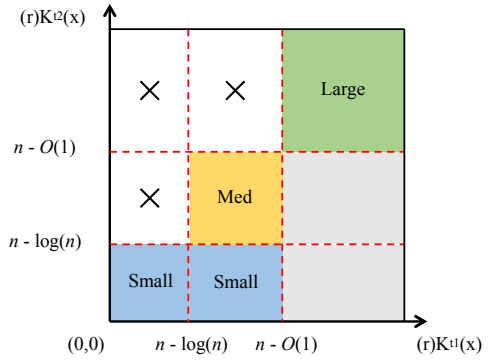

Figure 1 An illustrative example of the boundary-MINK^{t_1,t_2} problem where the two polynomials are t_1, t_2, $t_1 \leq t_2$. Worst-case hardness of deciding between Large and Med characterizes OWFs (see Theorem 1.1), whereas worst-case hardness of deciding between Large and (Med $\cup$ Small) implies NP $\notin$ AvgBPP (see Theorem 1.3).

Towards this, it is instructive to briefly review how [39] established the existence of OWFs from the boundary hardness of pK^t.

Why pK^t is Needed in [39]. Roughly speaking, the results of [39] first rely on the result of [33, 38] showing that average-case hardness of pK^t implies the existence of OWFs. In more detail, [33, 38] present a reduction for solving MINpK$^{\text{poly}}$ on average given a OWF inverter (for their specific OWF) (referred to as the LP20 reduction). [39] showed that the LP20 reduction, in fact, also correctly decides MINpK$^{\text{poly}}$ on all instances that "are along boundary".

This was proved in the following two steps: (1) [39] showed that elements on which the reduction fails can be sampled with probability $\geq n^c/2^n$ for any c, by an algorithm running in time $T = \text{poly}(n^c)$; and (2) relying on a so-called "Coding Theorem" [42], [39] proved the pK^T complexity of an element is, roughly, at most the *logarithm* of the inverse of the probability by which it can be sampled. Taken together, the reduction only fails on the elements of small pK^{poly} complexity (roughly at most $\log(2^n/n^c) = n - c\log n$), and thus will succeed on elements of either "intermediate" or "large" complexity – instances that are along the boundary.

We plan to rely on the same outline: take the LP20 reduction for rK^t and prove that it is correct on all boundary instances. However, to make the above proof also work w.r.t. rK^t, we would need a "Coding Theorem" for rK^t without assuming any derandomization assumptions, which is still open. A step towards such a Coding Theorem was recently taken by Hirahara et al [19], which shows that an "average-case coding theorem for rK^t" can be achieved assuming access to a OWF inverter. While in our setting, it is allowed to assume a OWF inverter (since we aim to establish the existence of OWFs), their notion unfortunately will not suffice for us – we require a coding theorem (that achieves non-trivial compression) for *all* strings in the support of the sampler.

Language Compression and rK^t. While it does not apply directly, it is instructive here to recall the techniques used in [19] to achieve rK^t-type compression. [19] relied on the result of Goldberg and Sipser [7], in which they showed that all sparse languages in BPP

admit a so-called *language compression scheme*, where a language compression scheme allows to compress all instances in a language by logarithmically many bits. [19] observed that admitting a language compression also implies that every instance in the language has small rK^t complexity (at most $n - O(\log n)$).

Thus, it would be sufficient for us to prove that the language of all the elements on which the LP20 reduction fails, denoted by the language Bad, admits a language compression scheme. Bad is indeed a sparse language (since, as mentioned, [39] showed that any instance in Bad can be sampled with probability $\geq n^c/2^n$ – there can be at most $2^n/n^c$ of them). However, it is unclear that Bad is in BPP: to check whether the reduction fails, we would need a "rK^t-witness", which cannot be efficiently found.

Our key observation is that instead of considering Bad, the set of bad instances, we can consider the set of rK^t-witnesses of bad instances, denoted as $\mathsf{BadWtns}$. We first remark that as a corollary of the [33] reduction, we have that not only Bad is sparse, but also $\mathsf{BadWtns}$. In addition, $\mathsf{BadWtns}$ is in BPP since one can produce the bad instance from a bad witness (by running the witness as a program), run the reduction on the instance, and check whether it fails using the witness. Finally, notice that if all rK^t-witness of bad instances have small rK^{poly} complexity, the instances themselves will also have small rK^{poly}-complexity, since, again, the instances can be obtained from running the witnesses. The above simple approach suffices to show that worst-case hardness of deciding whether $K^t(x)$ is large or small conditioned on $rK^{\mathsf{poly}(t)} > n - \log n$. To obtain the full result from just boundary hardness of rK^t we need to adjust the OWF construction from [33] to consider also randomized programs, which requires additional care – we refer the reader to the formal proof for the details.

Dealing with just K^t using Derandomization. We highlight that the only reason the above approach requires using rK^t as opposed to K^t is that the decoding ("decompression") algorithm of [7] is randomized. However under standard derandomization assumption, we can simply derandomize this algorithm (by enumerating all seeds to a complexity-theoretic PRG and picking the most likely output) and obtain a deterministic decoding. This suffices to show the same result but starting with just boundary hardness of K^t.

Let us briefly mention how this contrasts with the approach in [39] (and in particular, how we are able to weaken the derandomization assumption). There, the authors established an unconditional characterization of OWFs in terms of boundary of pK^t and next relied on a strong derandomization assumption to obtain a similar bound w.r.t. K^t; the fact that we can get the characterization in terms of rK^t is what enables us to use a much weaker derandomization assumptions.[4]

A New Worst-case to Errorless Average-case Reduction. We finally outline the idea behind our new worst-case to average-case reduction for K^t under derandomization assumptions. For simplicity, we here argue that worst-case hardness implies errorless hardness against deterministic algorithms (but the argument also readily extends to the case of randomized heuristic with some care). The key point is that instances on which an errorless reduction fails are sparse and explicitly identifiable (since the heuristic needs to output $\perp$). Thus, we can again use the language compression algorithm of [7] to compress them, and thus by the above argument, their K^t-complexity must be bounded by $n - \log n$ (under derandomization

[4] We highlight that it is an open problem that obtain a tight bound on the distance of pK^t and K^t based on standard derandomization assumptions.

assumptions, and unconditionally if considering rK^t). So any errorless heuristic can be turned in to a worst-case decider for K^t (resp. rK^t) by simply outputting YES whenever the heuristic outputs $\perp$!

2 Preliminaries

2.1 One-Way Functions

We recall the definition of one-way functions [5]. Roughly speaking, a function f is one-way if it is polynomial-time computable, but hard to invert for PPT attackers.

▶ **Definition 2.1.** *Let $f : \{0,1\}^* \to \{0,1\}^*$ be a polynomial-time computable function. f is said to be a* one-way function (OWF) *if for every* PPT *algorithm $\mathcal{A}$, there exists a negligible function μ such that for all $n \in \mathbb{N}$,*

$$\Pr[x \leftarrow \{0,1\}^n; y = f(x) : A(1^n, y) \in f^{-1}(f(x))] \leq \mu(n)$$

We may also consider a weaker notion of a *weak one-way function* [52], where we only require all PPT attackers to fail with probability noticeably bounded away from 1:

▶ **Definition 2.2.** *Let $f : \{0,1\}^* \to \{0,1\}^*$ be a polynomial-time computable function. f is said to be an* α-weak one-way function (α-weak OWF) *if for every* PPT *algorithm $\mathcal{A}$, for all sufficiently large $n \in N$,*

$$\Pr[x \leftarrow \{0,1\}^n; y = f(x) : A(1^n, y) \in f^{-1}(f(x))] < 1 - \alpha(n)$$

We say that f is simply a weak one-way function (weak OWF) *if there exists some polynomial $q > 0$ such that f is a $\frac{1}{q(\cdot)}$-weak OWF.*

Yao's hardness amplification theorem [52] shows that any weak OWF can be turned into a (strong) OWF.

▶ **Theorem 2.3** ([52])**.** *Assume there exists a weak one-way function. Then there exists a one-way function.*

2.2 Language Compression Schemes for BPP

We introduce the notion of language compression we rely on. We highlight here that we do not consider the complexity of the compressing algorithm (and the compression can be just existential).

We generalize the notion of language compression to promise problems; additionally, we want the definition to apply for all input lengths where the language is sparse (i.e, the compression works input-length by input-length). We say that a promise problem Π is *s-sparse* over input of length n if $|\{0,1\}^n \setminus \Pi_{\mathsf{NO}}|/2^n \leq s$.

▶ **Definition 2.4.** *For any promise problem $\Pi = (\Pi_{\mathsf{YES}}, \Pi_{\mathsf{NO}})$, we say that Π admits a* $(\alpha(\cdot), \ell(\cdot))$-language-compression scheme *if there exists a* PPT *decompressing* Dec *such that the following holds. For all sufficiently large $n \in \mathbb{N}$, if Π is $\alpha(n)$-sparse over input of length n, then for any $x \in \{0,1\}^n \cap \Pi_{\mathsf{YES}}$, there exists a string $y \in \{0,1\}^{\ell(|x|)}$, such that*

$$\Pr[\mathsf{Dec}(y) = x] \geq \frac{2}{3}$$

We recall the language compression scheme of Goldberg and Sipser [7]. (We note that while the language compression scheme considers languages in P, as pointed out in [7], their scheme also works when the language is a BPP promise problem; and we observe that the compression works in an input-length by input-length manner.)

▶ **Theorem 2.5** ([7]). *For any promise problem $\Pi \in \mathsf{BPP}$, any integer $k > 3$, Π admits a $(1/n^k, n - (k-3)\log n)$-language-compression scheme.*

2.3 Errorless Average-case Complexity

We finally recall the definition of *errorless* average-case complexity with respect to the *uniform* distribution on instances.

▶ **Definition 2.6** (AvgBPP). *For a promise problem $\Pi = (\Pi_{\mathsf{YES}}, \Pi_{\mathsf{NO}})$, we say that $\Pi \in \mathsf{AvgBPP}$ if for all polynomial $p(\cdot)$, there exists a probabilistic polynomial-time heuristic $\mathcal{H}$, such that for all sufficiently large n, for every $x \in \Pi_{\mathsf{YES}} \cap \{0,1\}^n$ (or $x \in \Pi_{\mathsf{NO}} \cap \{0,1\}^n$),*[5]

$$\Pr[\mathcal{H}(x) \in \{b, \perp\}] \geq 0.9,$$

(where $b = 1$ if $x \in \Pi_{\mathsf{YES}}$ and $b = 0$ if $x \in \Pi_{\mathsf{NO}}$) and

$$\Pr[x \leftarrow \mathcal{U}_n : \mathcal{H}(x) = \perp] \leq \frac{1}{p(n)}.$$

We will refer to problems in AvgBPP as problems that admit *errorless* heuristics. To better understand the class AvgBPP, it may be useful to compare it to the class AvgP (problems solvable by *deterministic* errorless heuristics): $\Pi \in \mathsf{AvgP}$ if for every polynomial $p(\cdot)$, there exists some deterministic polynomial-time heuristic $\mathcal{H}$ such that (a) for every input x, $\mathcal{H}(x)$ outputs either the correct answer or $\perp$, and (b) the probability over uniform n-bit inputs x that $\mathcal{H}$ outputs $\perp$ is bounded by $\frac{1}{p(n)}$. In other words, the *only* way an errorless heuristic may make a "mistake" is by saying $\perp$ ("I don't know"). AvgBPP is simply the natural "BPP-analog" of AvgP where the heuristic is allowed to be randomized.

3 Time-Bounded Kolmogorov Complexity Problems

Before presenting the main results in this work, let us formally define the computational problems needed in our results.

We first recall the notion of time-bounded Kolmogorov complexity [31, 50, 30, 12], and its randomized extension [41, 46, 40].

Time-bounded Kolmogorov Complexity [31, 50, 30, 12]. Roughly speaking, let the t-bounded Kolmogorov complexity of a string x, denoted $K^t(x)$, be defined as the length of the shortest program $\Pi = (M, y)$ such that $M(y)$ outputs the string x within time $t(|x|)$. Formally, fix U to be a universal Turing machine. For any string $x \in \{0,1\}^*$, define

$$K^t(x) = \min_{\Pi \in \{0,1\}^*} \{|\Pi| : U(\Pi, 1^{t(|x|)}) = x\}$$

[5] We remark that the constant 0.9 can be made arbitrarily small – any constants bounded away from $\frac{2}{3}$ works as we can amplify it using a standard Chernoff-type argument.

Randomized Time-bounded Kolmogorov complexity [41, 46, 40]. Roughly speaking, the *randomized t-time-bounded Kolmogorov complexity*, $rK^t(x)$, of a string $x \in \{0,1\}^*$ is the length of the shortest *randomized* program $\Pi = (M, y)$ such that Π_r (where Π_r denotes the program Π with its random tape fixed to be r) outputs x in $t(|x|)$ steps for at least a $2/3$ fraction of its random tapes r.

Formally, fix U to be a universal Turing machine. For any string $x \in \{0,1\}^*$, define

$$rK^t(x) = \min_{\Pi \in \{0,1\}^*} \{|\Pi| : \Pr_{r \in \{0,1\}^{t(|x|)}}[U(\Pi_r, 1^{t(|x|)}) = x] \geq 2/3\}$$

where for any (randomized) program Π, let Π_r denote the deterministic program where the random tape is fixed to be r. We will also consider a generalized version where the probability $2/3$ can be replaced by δ: For any $0 < \delta < 1$, we define $rK^t_\delta(x)$ as the same quantity $rK^t(x)$ but with $2/3$ replaced by δ.

The K^{poly} Problem and Its Boundary Version. For any polynomials t_1, t_2, $t_1(n) \leq t_2(n)$, define the time-bounded Kolmogorov complexity problem (denoted by MINK^{t_1,t_2}) as the promise problem consisting of

- YES: strings $x \in \{0,1\}^n$, $K^{t_1}(x) \leq n - 2$.
- NO: strings $x \in \{0,1\}^n$, $K^{t_2}(x) \geq n - 1$.

In addition, we can define its boundary version (denoted by $\mathsf{boundary\text{-}MINK}^{t_1,t_2}$) as

- YES: strings $x \in \{0,1\}^n$, $K^{t_1}(x) \leq n - 2$ and $K^{t_2}(x) > n - \log n$.
- NO: strings $x \in \{0,1\}^n$, $K^{t_2}(x) \geq n - 1$.

rK^{poly} Problem and Its Boundary Version. For any polynomials t_1, t_2, $t_1(n) \leq t_2(n)$, and a parameter $\delta(n) > 0$, define the time-bounded randomized Kolmogorov complexity problem (denoted by $\mathsf{MINrK}^{t_1,t_2}_\delta$) as the promise problem consisting of

- YES: strings $x \in \{0,1\}^n$, $rK^{t_1}_\delta(x) \leq n - \lceil \log(1/(1-\delta)) \rceil - 2$.
- NO: strings $x \in \{0,1\}^n$, $rK^{t_2}_{1-\delta}(x) \geq n - \lceil \log(1/(1-\delta)) \rceil - 1$.

We remark that the threshold (compared with the threshold in MINK^{t_1,t_2}) is shifted from $n-2$ to $n - \lceil \log(1/(1-\delta)) \rceil - 2$ to ensure that the NO instance set is not empty.

Analogously, we can also define the boundary version (denoted by $\mathsf{boundary\text{-}MINrK}^{t_1,t_2}_\delta$) as

- YES: strings $x \in \{0,1\}^n$, $rK^{t_1}_\delta(x) \leq n - \lceil \log(1/(1-\delta)) \rceil - 2$ and $rK^{t_2}_{1-\delta}(x) > n - \log n$.
- NO: strings $x \in \{0,1\}^n$, $rK^{t_2}_{1-\delta}(x) \geq n - \lceil \log(1/(1-\delta)) \rceil - 1$.

We will consider $\delta = \frac{2}{3}$ and simply denote $\mathsf{MINrK}^{t_1,t_2}_\delta$ (resp $\mathsf{boundary\text{-}MINrK}^{t_1,t_2}_\delta$) by MINrK^{t_1,t_2} (resp $\mathsf{boundary\text{-}MINrK}^{t_1,t_2}$) where $\delta = \frac{2}{3}$.

Finally, for K^{poly} (resp rK^{poly}), we say that $\mathsf{MINK}^{\mathsf{poly}} \notin \mathsf{ioBPP}$ (resp $\mathsf{MINrK}^{\mathsf{poly}} \notin \mathsf{ioBPP}$) if for all polynomials t_1, t_2, $t_2(n) \geq t_1(n) \geq 2n$, $\mathsf{MINK}^{t_1,t_2} \notin \mathsf{ioBPP}$ (resp $\mathsf{MINrK}^{t_1,t_2} \notin \mathsf{ioBPP}$).

4 New Characterizations of OWF Through Boundary Hardness

4.1 Main Theorems

We are ready to describe the main results in this work. Our first result shows that the existence of OWFs is equivalent to the worst-case hardness of the boundary version of $\mathsf{MINrK}^{\mathsf{poly}}$.

▶ **Theorem 4.1.** *OWFs exist if and only if* $\mathsf{boundary\text{-}MINrK}^{\mathsf{poly}} \notin \mathsf{ioBPP}$.

Proof. The theorem follows from Theorem 4.3 (which together with Yao's hardness amplification Theorem, Theorem 2.3), proves the "if" direction; the "only-if" direction is proved in Theorem 4.7. ◀

Our next result extends this theorem to obtain a characterization of OWFs based on the worst-case hardness of the boundary version of $\mathsf{MINK}^{\mathsf{poly}}$; this time, however, we will rely on a standard derandomization assumption.

▶ **Theorem 4.2.** *Assume that there exists a constant $\varepsilon > 0$ such that $\mathsf{E} \not\subseteq \mathsf{ioSIZE}[2^{\varepsilon n}]$. Then, OWFs exist if and only if* $\mathsf{boundary\text{-}MINK}^{\mathsf{poly}} \notin \mathsf{ioBPP}$.

Proof. The "if" direction follows from Theorem 4.6 together with Yao's hardness amplification Theorem (Theorem 2.3); the "only-if" direction was already proved in [39] (without relying on any derandomization assumption). ◀

4.2 OWFs from Boundary Hardness of $(r)K^{\mathsf{poly}}$

We first show a construction of OWF from the worst-case hardness of $\mathsf{boundary\text{-}MINrK}^{\mathsf{poly}}$. Since rK^{poly} can be derandomized to K^{poly} under standard derandomization assumption, this result will directly also extend to $\mathsf{boundary\text{-}MINK}^{\mathsf{poly}}$ under the same derandomization assumption.

▶ **Theorem 4.3.** *Assume that* $\mathsf{boundary\text{-}MINrK}^{\mathsf{poly}} \notin \mathsf{ioBPP}$. *Then, weak one-way functions exist.*

Proof. Consider any polynomial $t_1(n)$, and assume that for all $t_2 \geq t_1$, $\mathsf{boundary\text{-}MINrK}^{t_1,t_2} \notin \mathsf{ioBPP}$. We consider the function $f : \{0,1\}^{\lceil \log(n) \rceil + n + t_1(n) \cdot n^3} \to \{0,1\}^*$, which takes an input $\ell || \Pi' || r_1 || \dots || r_{n^3}$ where $|\ell| = \lceil \log(n) \rceil$, $|\Pi'| = n$ and $|r_i| = t_1(n)$ for each $i \in [n^3]$, computes for each $i \in [n^3]$

$$x_i = U(\Pi_{r_i}, 1^{t_1(n)})$$

where Π is the ℓ-bit prefix of Π' (and the bit-string ℓ is interpreted as an integer $\in [n]$), and Π_{r_i} denotes the program Π with random tape r_i. If there exists a string x such that the majority among x_i equals x, it outputs

$$f(\ell || \Pi' || r_1 || \dots || r_{n^3}) = \ell || x || r_1 || \dots || r_{n^3}$$

Otherwise (i.e., there is no majority among x_i) it outputs $\perp$.

This function is only defined over some input lengths, but by an easy padding trick, it can be transformed into a function f' defined over all input lengths, such that if f is weakly one-way (over the restricted input lengths), then f' will be weakly one-way (over all input lengths): $f'(x')$ simply truncates its input x' (as little as possible) so that the (truncated) input x now becomes of length $n' = \lceil \log(n) \rceil + n + t_1(n) \cdot n^3$ for some n and outputs $f(x)$. This will decrease the input length by a polynomial factor (since t_1 is a polynomial) so the padding trick can be applied here.

We show that f is a weak OWF (over the restricted input length). Let $q(n) = 24(2n)^6$. We assume for contradiction that f is not $\frac{1}{q}$-weak one-way. (In the proof below, although the input length of f we consider is $m = \lceil \log(n) \rceil + n + t_1(n) \cdot n^3$ for some n, we will view n as the "security parameter", computing the running time and the inverting success probability in terms of the security parameter n, but analyzing the one-wayness of f on input length $m = m(n)$. Since n and m are polynomially related, we can still conclude that f is weak one-way.) Then, there exists a PPT attacker $\mathcal{A}$ that inverts f with probability at least $1 - \frac{1}{q(n)}$ for infinitely many n.

We turn to constructing a PPT algorithm M deciding whether an input x is of rK^{t_1}-complexity $< |x| - 3$. Our algorithm M, on input $x \in \{0,1\}^n$, samples n^3 random strings $r_1, \ldots, r_{n^3} \in \{0,1\}^{t_1(n)}$. Then, it runs $\mathcal{A}(i||x||r_1|| \ldots ||r_{n^3})$ for every $i \in [n]$ where i is represented as a $\lceil \log(n) \rceil$-bit string. In addition, $M(x)$ checks whether $\mathcal{A}$ successfully finds a valid pre-image of f on $i||x||r_1|| \ldots ||r_{n^3}$. Finally, $M(x)$ outputs YES if and only if the shortest index i on which $\mathcal{A}$ succeeds in inverting f is $< n - 3$. (Otherwise, output NO.) Since $\mathcal{A}$ runs in polynomial time, the algorithm M will also terminate in polynomial time.

We will show that the algorithm M decides the promise problem boundary-MINrK^{t_1,t_2} for some polynomial t_2 (which will be fixed later). Toward this, we consider a promise problem Fail defined as follows: For any $y \in \{0,1\}^{n+\lceil \log(n) \rceil}$, we view y as $\ell||\Pi'$ where $\ell = \lceil \log(n) \rceil$ and $|\Pi'| = n$. Let p_y denote the probability that $\mathcal{A}$ fails to invert f on $f(\ell||\Pi'||r_1|| \ldots ||r_{n^3})$ where $r_1, \ldots, r_{n^3} \in \{0,1\}^{t_1(n)}$ are sampled at random. We define $y \in \mathsf{Fail}_{\mathsf{YES}}$ if $p_y \geq 1/6$ (and $y \in \mathsf{Fail}_{\mathsf{NO}}$ if $p_y < 1/12$). Since both f and $\mathcal{A}$ are efficient, we have that Fail $\in$ BPP.

Pick $k = 6$. By Theorem 2.5, there exists a $(1/n'^6, n' - 3\log n')$-language-compression scheme for Fail (where n' denotes the input length for Fail). Let Dec denote the decoding (or decompressing) algorithm for the scheme.

Select $t_2(n)$ to be the runtime bound for the following algorithm Recon: Run the decoding algorithm Dec (on some valid compression of y) to get an instance $y \in \mathsf{Fail}_{\mathsf{YES}}$, $y \in \{0,1\}^{n+\lceil \log(n) \rceil}$. Since y can be viewed in the form of $\ell||\Pi'$, let $y = \ell||\Pi'$ and let Π be the ℓ-bit prefix of the string Π'. Run Π over a random random-tape r for $t_1(n)$ of steps and output the output of Π_r.

Now we are ready to show that M decides boundary-MINrK^{t_1,t_2} for infinitely many input lengths n. Fix some sufficiently large input length n where $\mathcal{A}$ inverts the function f with probability $1 - \frac{1}{q(n)}$ on n. We first show that if x is a NO instance of boundary-MINrK^{t_1,t_2}, $M(x)$ outputs NO with probability $\geq 2/3$.

▷ Claim 4.4. If $rK^{t_2}_{1/3}(x) \geq n - 3$, then $M(x)$ outputs NO with probability $\geq 2/3$.

Proof. We show that $M(x)$ outputs YES with probability $\leq 1/3$. Recall that $M(x)$ will output YES if there exists an index $i \leq n - 4$ and a program Π of length i such that Π_{r_j} outputs x for a half fraction of random-tapes r_j, $j \in [n^3]$. We refer to such $r_1, \ldots, r_{n^3}$ as being "bad".

We turn to showing that among all possible choices of random tapes, there are at most a $1/3$ fraction of them being bad. Consider any program Π'' of length $w \leq n - 4$. Since $rK^{t_2}_{1/3}(x) \geq n - 3$, by a standard Chernoff bound, there is no more than $2^{-n\sqrt{n}+O(1)}$ fraction of $r_1, \ldots, r_{n^3}$ such that Π''_{r_j} outputs x for a majority of r_j. Taking a Union bound over all possible Π'', we conclude that the probability that $r_1, \ldots, r_{n^3}$ is bad is at most

$$2^{n-3} \cdot 2^{-n\sqrt{n}+O(1)} \leq \frac{1}{3}$$

when n is sufficiently large. ◁

Finally, we show that if x is a YES instance of boundary-MINrK^{t_1,t_2}, $M(x)$ will output YES with probability $\geq 2/3$. Since $rK^{t_1}_{2/3}(x) < n - 3$, it follows that there exists a program Π of length $< n - 3$ such that Π_r will output x within $t_1(n)$ steps over at least $2/3$ fraction of random-tape r. Let w denote $rK^{t_1}_{2/3}(x)$. If $\mathcal{A}$ finds a valid pre-image of f on $w||x||r_1|| \ldots ||r_{n^3}$, $M(x)$ will outputs YES. We show that this will happen with high probability.

▷ Claim 4.5. $\mathcal{A}(w||x||r_1|| \ldots ||r_{n^3})$ will return a valid pre-image of f with probability $\geq 2/3$ over random $r_1, \ldots, r_{n^3}$.

Proof. Since Π is a $rK^{t_1}_{2/3}$-witness of x of length w, by the Chernoff bound, there exists a valid pre-image of f on $w||x||r_1||\dots||r_{n^3}$ for at least a $1-1/6$ fraction of $r_1,\dots,r_{n^3}$. Assume for contradiction that $\mathcal{A}$ fails to find a valid pre-image on $w||x||r_1||\dots||r_{n^3}$ with probability $> 1/3$. We will show that

$$rK^{t_2}_{1/3}(x) < n - \log n$$

which is a contradiction since x is a YES instance of $\mathsf{boundary\text{-}MINrK}^{t_1,t_2}$.

Consider any string Π' of length n such that its w-bit prefix is Π. By taking a Union bound, it follows that $\mathcal{A}$ fails to invert f on $f(w||\Pi'||r_1||\dots||r_{n^3})$ with probability $\geq 1/3-1/6 \geq 1/6$. Thus, the concatenation $w||\Pi'$ will be an YES-instance of Fail (where Fail is the promise problem we defined before). We next show that Fail is $\frac{1}{n'^6}$-sparse on input of length $n' = n + \lceil \log(n) \rceil$. Suppose not. Then, it follows that the inverter $\mathcal{A}$ fails to invert f with probability

$$\frac{1}{n'^6} \cdot \frac{1}{12} > q(n)$$

which contradicts to the fact that $\mathcal{A}$ is a good inverter on n. Recall that Dec is the decoding algorithm for the $(1/n'^6, n' - 3\log n')$-language-compress scheme for Fail. Thus, there exists a string z (i.e., the compression of $w||\Pi'$), $|z| \leq n' - 3\log n' \leq n - 2\log n$, such that $\mathsf{Dec}(z)$ outputs $w||\Pi'$ with probability $2/3$. It follows (from our definition of algorithm Recon) that $\mathsf{Recon}(z)$ outputs x with probability $(2/3)\cdot(2/3) > 1/3$. Note that $\mathsf{Recon}(z)$ can be written as a program of length $< n - \log n$ and it outputs x with probability $> 1/3$ within $t_2(n)$ steps (due to our choice of t_2), and therefore

$$rK^{t_2}_{1/3}(x) < n - \log n$$

which completes the proof. ◁

◀

We proceed to extend this result to work also for $\mathsf{boundary\text{-}MINK}^{\mathsf{poly}}$ under standard derandomization assumptions.

▶ **Theorem 4.6.** *Assume that there exists a constant $\varepsilon > 0$ such that $\mathsf{E} \not\subseteq \mathsf{ioSIZE}[2^{\varepsilon n}]$. Then, $\mathsf{boundary\text{-}MINK}^{\mathsf{poly}} \notin \mathsf{ioBPP}$ implies that weak one-way functions exist.*

Proof (sketch). The proof will proceed largely as the proof of Theorem 4.3, but with minor adjustments. In particular, we will go back to the original OWF construction of [33] (which is exactly the same as the one in Theorem 4.3 except that we no longer consider randomized programs Π, so no longer including the random strings $r_1,\dots$). Specifically, we define

$$f'(\ell||\Pi') = U(\Pi, 1^{t_1(n)})$$

where notations are as in Theorem 4.3.

With the new OWF construction, we will be needing to adapt the rK^{poly} decider M to deal with K^{poly}. The new algorithm M' follows roughly M, except that (1) M' no longer needs to sample random strings $r_1,\dots$; and (2) it replaces the threshold $n-3$ with $n-1$ (as required by the Theorem).

Finally, we need to show that Claim 4.4 and Claim 4.5 hold w.r.t. K^{poly}, the new construction f', threshold $n-1$, and algorithm M'. Claim 1 trivially holds since M' outputs YES only when it finds a K^{t_1}-witness of length $< n-1$. Note that $K^{t_1} \geq K^{t_2}$, so there will

never be such a witness when $K^{t_2} \geq n-1$. Claim 2 is where we require some additional work. Note that to conclude Claim 2, we need to show that if $\mathcal{A}$ fails, then $K^{t_2}(x) < n - \log n$ (as opposed to rK^{t_2}). Recall that in Claim 2, it is shown that the string x can be generated w.p 2/3 by a randomized decoding algorithm $\mathsf{Dec}(z)$ on input a string z of length $n - 2\log n$. So to demonstrate the above claim, we just need to show how to derandomize Dec, which can be done using standard techniques. In more detail, let $C_{x,z}$ denote the circuit that receives a random-tape for Dec as input, and outputs 1 if the random-tape leads $\mathsf{Dec}(z)$ to output x. By [28], under the assumption of the theorem, we have that there exists a (complexity-theoretic) PRG G with seed length $O(\log n)$ that fools $C_{x,z}$. Thus, Dec can be derandomized by emulating Dec on all (pseudo) random-tapes generated by running G on all possible seeds, and outputting the string that appears the most often. ◀

4.3 Boundary Hardness of rK^{poly} from OWFs

We here prove the worst-case hardness of $\mathsf{boundary\text{-}MINrK}^{\mathsf{poly}}$ from the existence of OWFs.

▶ **Theorem 4.7.** *Assume that OWFs exist. Then,* $\mathsf{boundary\text{-}MINrK}^{\mathsf{poly}} \notin \mathsf{ioBPP}$.

Proof. The proof for this theorem closely follows from the proofs in [39], in which they show that OWFs imply the boundary hardness of the same problem but with respect to another complexity measure (pK^{poly}). Their proof proceeds in two steps: First, they show that OWFs implies the existence of a so-called conditionally-secure entropy-preserving pseudorandom generator [33] (Cond-EP PRG); second, they show that a Cond-EP PRG (with certain parameters) implies the hardness of the boundary pK^{poly} problem (in [39, Lemma 6.3]).

In order to prove the above theorem, the first step follows from the same proof. The following two claims are needed in the proof for the second step: (1) the output of Cond-EP PRG has "small" rK^{poly}-complexity (which follows from the fact that the output of Cond-EP PRG can be generated using a small program (without using the random tape)) and (2) for any string x and runtime bound t, $pK^t(x) \leq rK^t(x)$ (c.f., [8, Proposition 19]). Thus, the second step follows from the proof of [39, Lemma 6.3] with pK^{poly} replaced by rK^{poly}. ◀

5 New Worst-case to (Errorless) Average-case Reductions

We move on to proving that the worst-case hardness of $\mathsf{MINK}^{\mathsf{poly}}$ is equivalent to its errorless average-case hardness under standard derandomization assumptions; additionally, we show the same result unconditionally holds for rK^t (i.e., without derandomization assumptions).

We start by showing the result for rK^t as its proof is simpler; the proof for K^t is very similar but requires some additional work to deal with the derandomization.

▶ **Theorem 5.1.** *If for all polynomials* t_1, t_2, $t_2(n) \geq t_1(n) \geq 2n$, *it holds that*

$$\mathsf{MINrK}^{t_1,t_2} \notin \mathsf{P}$$

then for all polynomials t_1, t_2, $t_2(n) \geq t_1(n) \geq 2n$, *it holds that*

$$\mathsf{MINrK}^{t_1,t_2} \notin \mathsf{AvgP}$$

Moreover, the converse is also true.

Proof. To see that the backward direction is true, simply consider any t_1, t_2 and notice that if $\mathsf{MINK}^{t_1,t_2} \in \mathsf{P}$, the worst-case algorithm for MINrK^{t_1,t_2} is trivially an (average-case) errorless heuristic for the same problem, and thus $\mathsf{MINrK}^{t_1,t_2} \in \mathsf{AvgP}$.

We turn to proving the forward direction. Consider any polynomial t_1, t_2, and let $q(n) = n^4$ be a polynomial. Suppose for contradiction that $\mathsf{MINrK}^{t_1,t_2} \in \mathsf{AvgP}$. Therefore, there exists a deterministic errorless heuristic $\mathcal{H}$ for MINrK^{t_1,t_2} such that $\Pr[x \leftarrow \{0,1\}^n : \mathcal{H}(x) = \bot] \leq \frac{1}{q(n)}$ for all sufficiently large $n \in \mathbb{N}$. Let L be the set of all x's such that $H(x) = \bot$; note that this language is in P (by the definition of $\mathcal{H}$) and is also sparse (by the above). Thus, by Theorem 2.5, L admits a ℓ-language-compression scheme for $\ell(n) \leq n - (\log q(n) - 3\log n) = n - \log n$. Let Dec be the PPT decompressing algorithm in the language compression scheme, and let t_3 be its running time. Note that for any $x \in L$, $n = |x|$, there exists some string y of length $\leq \ell(n)$ (i.e., the compression of x) such that $\mathsf{Dec}(y) = x$ with probability $2/3$; thus

$$rK^{t_3}(x) \leq n - \log n$$

We are now ready to present an efficient algorithm for MINK^{t_1,t_3}, which will be a contradiction. On input $x \in \{0,1\}^n$, our algorithm $\mathcal{A}$ simply runs $\mathcal{H}(x)$; if $\mathcal{H}(x)$ outputs $\bot$, $\mathcal{A}$ outputs 1, and otherwise it simply outputs whatever $\mathcal{A}$ outputs. Note that when $\mathcal{H}(x)$ does not output $\bot$, its answer needs to be correct. And when $\mathcal{H}(x)$ outputs $\bot$, as argued above, we have that $rK^{t_3}(x) \leq n - \log n$, and thus $\mathcal{A}$ will also provide the right answer. ◀

We move on to show the result for rK^t (under standard derandomization assumptions).

▶ **Theorem 5.2.** *Assume that there exists a constant $\varepsilon > 0$ such that $\mathsf{E} \not\subseteq \mathsf{ioSIZE}[2^{\varepsilon n}]$. If for all polynomials t_1, t_2, $t_2(n) \geq t_1(n) \geq 2n$, it holds that*

$$\mathsf{MINK}^{t_1,t_2} \notin \mathsf{BPP}$$

then for all polynomials t_1, t_2, $t_2(n) \geq t_1(n) \geq 2n$, it holds that

$$\mathsf{MINK}^{t_1,t_2} \notin \mathsf{AvgBPP}$$

Moreover, the converse is also true.

Proof. Let $s(n) = n - 2$ (and recall that for any $x \in \mathsf{MINK}^{t_1,t_2}_{\mathsf{YES}}$, we have that $K^{t_1}(x) \leq s(|x|)$ and for any $x \in \mathsf{MINK}^{t_1,t_2}_{\mathsf{NO}}$, we have that $K^{t_2}(x) > s(|x|)$).

To see that the backward direction is true, simply consider any t_1, t_2 and notice that if $\mathsf{MINK}^{t_1,t_2} \in \mathsf{BPP}$, the worst-case algorithm for MINK^{t_1,t_2} is trivially an (average-case) errorless heuristic for the same problem, and thus $\mathsf{MINK}^{t_1,t_2} \in \mathsf{AvgBPP}$.

We turn to proving the forward direction. Consider any polynomial t_1, t_2, and let $q(n) = 20n^5$ be a polynomial. Suppose for contradiction that $\mathsf{MINK}^{t_1,t_2} \in \mathsf{AvgBPP}$. Therefore, there exists an errorless heuristic $\mathcal{H}$ for MINK^{t_1,t_2} such that $\Pr[x \leftarrow \{0,1\}^n : \mathcal{H}(x) = \bot] \leq \frac{1}{q(n)}$ for all sufficiently large $n \in \mathbb{N}$. Note that for any $x \in \{0,1\}^*$, $K^{t_1}(x) \leq s(|x|)$, we have that $\Pr[\mathcal{H}(x) = \{\bot, 1\}] \geq 0.9$ (and also for any x such that $K^{t_2}(x) > s(|x|)$, $\Pr[\mathcal{H}(x) = \{\bot, 0\}] \geq 0.9$).

We will be needing the following BPP promise problem Π, where YES instances of Π consist of all instances x on which $\Pr[\mathcal{H}(x) = \bot] \geq 0.1$ (and NO instances are those x on which $\Pr[\mathcal{H}(x) = \bot] \leq 0.05$). Since we assume that there exists a constant $\varepsilon > 0$ such that $\mathsf{E} \not\subseteq \mathsf{ioSIZE}[2^{\varepsilon n}]$, $\mathsf{BPP} = \mathsf{P}$ [28]. Let L be the language $\in \mathsf{P}$ that Π derandomizes to. Since $\mathcal{H}$ is a good errorless heuristic, on all sufficiently large n, it holds that

$$|L \cap \{0,1\}^n| \leq \frac{1}{q(n)} \cdot \frac{1}{0.05} \leq \frac{20}{q(n)}$$

Thus, by Theorem 2.5, L admits a ℓ-language-compression scheme for $\ell(n) \leq n - (\log(q(n)/20) - 3\log n) = n - 2\log n$. Let Dec be the PPT decompressing algorithm in the language compression scheme. We are going to show that there exists a polynomial t_3 such that for any sufficiently long $x \in L$, $n = |x|$,

$$K^{t_3}(x) \leq n - \log n$$

Consider any such x, and let y be the string of length $\leq \ell(n)$ (i.e., the compression of x) guaranteed to exist by Theorem 2.5. Notice that $\mathsf{Dec}(y)$ will output x with probability $\geq 2/3$. It remains to derandomize Dec using the derandomization assumption. Let $C_{x,y}$ denote the circuit that receives a random-tape for Dec as input, and outputs 1 if the random-tape leads Dec to output x on input y. By [28], there exists a (complexity-theoretic) PRG G with seed length $O(\log n)$ that fools $C_{x,y}$ for any such x, y. Thus, Dec can be derandomized by emulating Dec on all (pseudo) random-tapes generated by running G on all possible seeds, and outputting the string that appears the most often. Denote this algorithm by Dec', and notice that Dec' runs in polynomial time. Let $t_3 \geq t_2$ be a polynomial that upper bounds the running time of Dec'. Since the machine with Dec' and y hardcoded will generate x in time $t_3(n)$ and it has description length $\leq O(1) + |y| \leq \log n + \ell(n) \leq n - \log n$, we conclude that

$$K^{t_3}(x) \leq n - \log n$$

We are now ready to present an efficient algorithm for MINK^{t_1,t_3}, which will be a contradiction. On input $x \in \{0,1\}^n$, our algorithm $\mathcal{A}$ first checks whether $x \in L$. If so, $\mathcal{A}(x)$ simply outputs 1. Otherwise, $\mathcal{A}(x)$ outputs $\mathcal{H}(x)$ if $\mathcal{H}(x) \neq \perp$ (otherwise it output 0). Observe that $\mathcal{A}$ runs in polynomial time (since $L \in \mathsf{P}$ and $\mathcal{H}$ is a PPT algorithm).

We turn to analysis the correctness of $\mathcal{A}$. For any x such that $K^{t_1}(x) \leq n - 2$ (namely that x is a YES instance), if $x \in L$, it follows that $\mathcal{A}(x)$ outputs 1. If $x \notin L$, it follows that $\mathcal{H}(x)$ outputs $\perp$ with probability at most 0.1. Since $\mathcal{H}(x)$ outputs either $\perp$ or 1 with probability at least 0.9, by a Union bound, $\mathcal{A}(x)$ output 1 with probability at least 0.8. For any x such that $K^{t_3}(x) > n - 2$ (namely that x is a NO instance), it follows that $x \notin L$. In addition, $K^{t_2}(x) \geq K^{t_3}(x) > n - 2$, and thus $\mathcal{H}(x)$ outputs either $\perp$ or 0 with probability 0.9, and therefore $\mathcal{A}(x)$ output 0 with probability 0.9. ◀

References

1. Eric Allender, Mahdi Cheraghchi, Dimitrios Myrisiotis, Harsha Tirumala, and Ilya Volkovich. One-way functions and a conditional variant of MKTP. *Electron. Colloquium Comput. Complex.*, 28:9, 2021.
2. Manuel Blum. Coin flipping by telephone - A protocol for solving impossible problems. In *COMPCON'82, Digest of Papers, Twenty-Fourth IEEE Computer Society International Conference, San Francisco, California, USA, February 22-25, 1982*, pages 133–137. IEEE Computer Society, 1982.
3. Manuel Blum and Silvio Micali. How to generate cryptographically strong sequences of pseudo-random bits. *SIAM Journal on Computing*, 13(4):850–864, 1984. doi:10.1137/0213053.
4. Lijie Chen and Roei Tell. Simple and fast derandomization from very hard functions: eliminating randomness at almost no cost. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pages 283–291, 2021. doi:10.1145/3406325.3451059.
5. Whitfield Diffie and Martin Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.
6. Uriel Feige and Adi Shamir. Witness indistinguishable and witness hiding protocols. In *STOC '90*, pages 416–426, 1990. doi:10.1145/100216.100272.

7 Andrew Goldberg and Michael Sipser. Compression and ranking. In *Proceedings of the seventeenth annual ACM symposium on Theory of computing*, pages 440–448, 1985. doi:10.1145/22145.22194.

8 Halley Goldberg, Valentine Kabanets, Zhenjian Lu, and Igor C Oliveira. Probabilistic kolmogorov complexity with applications to average-case complexity. In Shachar Lovett, editor, *37th Computational Complexity Conference (CCC 2022)*, volume 234 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 16:1–16:60, Dagstuhl, Germany, 2022. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.CCC.2022.16.

9 Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. In *FOCS*, 1984.

10 Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *J. Comput. Syst. Sci.*, 28(2):270–299, 1984. doi:10.1016/0022-0000(84)90070-9.

11 Yuri Gurevich. The challenger-solver game: variations on the theme of p=np. In *Logic in Computer Science Column, The Bulletin of EATCS*. 1989.

12 J. Hartmanis. Generalized kolmogorov complexity and the structure of feasible computations. In *24th Annual Symposium on Foundations of Computer Science (sfcs 1983)*, pages 439–445, November 1983. doi:10.1109/SFCS.1983.21.

13 Johan Håstad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. A pseudorandom generator from any one-way function. *SIAM J. Comput.*, 28(4):1364–1396, 1999. doi:10.1137/S0097539793244708.

14 Shuichi Hirahara. Non-black-box worst-case to average-case reductions within NP. In *59th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2018*, pages 247–258, 2018. doi:10.1109/FOCS.2018.00032.

15 Shuichi Hirahara. Non-disjoint promise problems from meta-computational view of pseudorandom generator constructions. In Shubhangi Saraf, editor, *35th Computational Complexity Conference (CCC 2020)*, volume 169 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 20:1–20:47, Dagstuhl, Germany, 2020. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.CCC.2020.20.

16 Shuichi Hirahara. Symmetry of information in heuristica. Manuscript, 2021.

17 Shuichi Hirahara. Np-hardness of learning programs and partial mcsp. In *2022 IEEE 63rd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 968–979. IEEE, 2022. doi:10.1109/FOCS54457.2022.00095.

18 Shuichi Hirahara, Rahul Ilango, and Ryan Williams. Beating brute force for compression problems. *Electronic Colloquium on Computational Complexity*, 2024. URL: https://eccc.weizmann.ac.il/report/2023/171/.

19 Shuichi Hirahara, Zhenjian Lu, and Mikito Nanashima. Optimal coding for randomized kolmogorov complexity and its applications. *arXiv preprint arXiv:2409.12744*, 2024. doi:10.48550/arXiv.2409.12744.

20 Shuichi Hirahara and Mikito Nanashima. Learning in pessiland via inductive inference. In *64nd Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE, 2023. doi:10.1109/FOCS57990.2023.00033.

21 Rahul Ilango. Approaching MCSP from above and below: Hardness for a conditional variant and AC^0[p]. In *11th Innovations in Theoretical Computer Science Conference, ITCS 2020*, pages 34:1–34:26, 2020. doi:10.4230/LIPIcs.ITCS.2020.34.

22 Rahul Ilango. The minimum formula size problem is (eth) hard. In *2021 IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 427–432. IEEE, 2021. doi:10.1109/FOCS52979.2021.00050.

23 Rahul Ilango. Constant depth formula and partial function versions of mcsp are hard. *SIAM Journal on Computing*, (0):FOCS20–317, 2022.

24 Rahul Ilango. Sat reduces to the minimum circuit size problem with a random oracle. In *2023 IEEE 64th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 733–742, 2023. doi:10.1109/FOCS57990.2023.00048.

25 Rahul Ilango, Bruno Loff, and Igor Carboni Oliveira. NP-hardness of circuit minimization for multi-output functions. In *35th Computational Complexity Conference, CCC 2020*, pages 22:1–22:36, 2020.

26 Russell Impagliazzo. A personal view of average-case complexity. In *Structure in Complexity Theory '95*, pages 134–147, 1995. doi:10.1109/SCT.1995.514853.

27 Russell Impagliazzo and Michael Luby. One-way functions are essential for complexity based cryptography (extended abstract). In *30th Annual Symposium on Foundations of Computer Science, Research Triangle Park, North Carolina, USA, 30 October - 1 November 1989*, pages 230–235, 1989. doi:10.1109/SFCS.1989.63483.

28 Russell Impagliazzo and Avi Wigderson. $P = BPP$ if e requires exponential circuits: Derandomizing the xor lemma. In *STOC '97*, pages 220–229, 1997. doi:10.1145/258533.258590.

29 Adam R Klivans and Dieter van Melkebeek. Graph nonisomorphism has subexponential size proofs unless the polynomial-time hierarchy collapses. *SIAM Journal on Computing*, 31(5):1501–1526, 2002. doi:10.1137/S0097539700389652.

30 Ker-I Ko. On the notion of infinite pseudorandom sequences. *Theor. Comput. Sci.*, 48(3):9–33, 1986. doi:10.1016/0304-3975(86)90081-2.

31 A. N. Kolmogorov. Three approaches to the quantitative definition of information. *International Journal of Computer Mathematics*, 2(1-4):157–168, 1968.

32 L. A. Levin. The tale of one-way functions. *Problems of Information Transmission*, 39(1):92–103, 2003. doi:10.1023/A:1023634616182.

33 Yanyi Liu and Rafael Pass. On one-way functions and Kolmogorov complexity. In *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020, Durham, NC, USA, November 16-19, 2020*, pages 1243–1254. IEEE, 2020. doi:10.1109/FOCS46700.2020.00118.

34 Yanyi Liu and Rafael Pass. Cryptography from sublinear time hardness of time-bounded kolmogorov complexity. In *STOC*, 2021.

35 Yanyi Liu and Rafael Pass. On the possibility of basing cryptography on EXP $\neq$ BPP. In *CRYPTO*, 2021.

36 Yanyi Liu and Rafael Pass. On one-way functions from np-complete problems. In *Proceedings of the 37th Computational Complexity Conference*, pages 1–24, 2022. doi:10.4230/LIPIcs.CCC.2022.36.

37 Yanyi Liu and Rafael Pass. On one-way functions and the worst-case hardness of time-bounded kolmogorov complexity. *Cryptology ePrint Archive*, page 1086, 2023. URL: https://eprint.iacr.org/2023/1086.

38 Yanyi Liu and Rafael Pass. One-way functions and the hardness of (probabilistic) time-bounded kolmogorov complexity w.r.t. samplable distributions. In *CRYPTO'23*, 2023.

39 Yanyi Liu and Rafael Pass. Hardness along the boundary: Towards one-way functions from the worst-case hardness of time-bounded kolmogorov complexity. In *CRYPTO*, 2025.

40 Zhenjian Lu and Igor C Oliveira. An efficient coding theorem via probabilistic representations and its applications. In Nikhil Bansal, Emanuela Merelli, and James Worrell, editors, *48th International Colloquium on Automata, Languages, and Programming (ICALP 2021)*, volume 198 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 94:1–94:20, Dagstuhl, Germany, 2021. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.ICALP.2021.94.

41 Zhenjian Lu, Igor C Oliveira, and Rahul Santhanam. Pseudodeterministic algorithms and the structure of probabilistic time. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pages 303–316, 2021. doi:10.1145/3406325.3451085.

42 Zhenjian Lu, Igor C Oliveira, and Marius Zimand. Optimal coding theorems in time-bounded kolmogorov complexity. In MikoLargeaj Bojańczyk, Emanuela Merelli, and David P. Woodruff, editors, *49th International Colloquium on Automata, Languages, and Programming (ICALP 2022)*, volume 229 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 92:1–92:14, Dagstuhl, Germany, 2022. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.ICALP.2022.92.

43 Noam Mazor and Rafael Pass. The non-uniform perebor conjecture for time-bounded kolmogorov complexity is false. *Cryptology ePrint Archive*, 2023.

44 Moni Naor. Bit commitment using pseudorandomness. *J. Cryptology*, 4(2):151–158, 1991. doi:10.1007/BF00196774.

45 Noam Nisan and Avi Wigderson. Hardness vs randomness. *J. Comput. Syst. Sci.*, 49(2):149–167, 1994. doi:10.1016/S0022-0000(05)80043-1.

46 Igor Carboni Oliveira. Randomness and intractability in kolmogorov complexity. In Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi, editors, *46th International Colloquium on Automata, Languages, and Programming (ICALP 2019)*, volume 132 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 32:1–32:14, Dagstuhl, Germany, 2019. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.ICALP.2019.32.

47 Hanlin Ren and Rahul Santhanam. Hardness of KT characterizes parallel cryptography. *Electron. Colloquium Comput. Complex.*, 28:57, 2021. URL: https://eccc.weizmann.ac.il/report/2021/057.

48 Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. A method for obtaining digital signatures and public-key cryptosystems (reprint). *Commun. ACM*, 26(1):96–99, 1983. doi:10.1145/357980.358017.

49 John Rompel. One-way functions are necessary and sufficient for secure signatures. In *STOC*, pages 387–394, 1990. doi:10.1145/100216.100269.

50 Michael Sipser. A complexity theoretic approach to randomness. In *Proceedings of the 15th Annual ACM Symposium on Theory of Computing, 25-27 April, 1983, Boston, Massachusetts, USA*, pages 330–335. ACM, 1983. doi:10.1145/800061.808762.

51 Boris A Trakhtenbrot. A survey of Russian approaches to perebor (brute-force searches) algorithms. *Annals of the History of Computing*, 6(4):384–400, 1984. doi:10.1109/MAHC.1984.10036.

52 Andrew Chi-Chih Yao. Theory and applications of trapdoor functions (extended abstract). In *23rd Annual Symposium on Foundations of Computer Science, Chicago, Illinois, USA, 3-5 November 1982*, pages 80–91, 1982. doi:10.1109/SFCS.1982.45.