Failure of Symmetry of Information for Randomized Computations

Jinqiao Hu* Yahel Manor† Igor C. Oliveira‡

February 16, 2026

Abstract

Symmetry of Information (SoI) is a fundamental result in Kolmogorov complexity stating that for all n-bit strings x and y, $\mathsf{K}(x,y) = \mathsf{K}(y) + \mathsf{K}(x \mid y)$ up to an additive error of $O(\log n)$ [ZL70]. In contrast, understanding whether SoI holds for *time-bounded* Kolmogorov complexity measures is closely related to longstanding open problems in complexity theory and cryptography, such as the P versus NP question [LW95, Hir22] and the existence of one-way functions [HIL+23, HLO24, HLN24].

In this paper, we prove that SoI fails for rKt complexity, the randomized analogue of Levin's Kt complexity [Lev84]. This is the first unconditional result of this type for a randomized notion of time-bounded Kolmogorov complexity. More generally, we establish a close relationship between the validity of SoI for rKt and the existence of randomized algorithms approximating $\mathsf{rKt}(x)$. Motivated by applications in cryptography, we also establish the failure of SoI for a related notion called pKt complexity [HLO24], and provide an extension of the results to the average-case setting. Finally, we prove a near-optimal lower bound on the complexity of estimating conditional rKt, a result that might be of independent interest.

Our findings complement those of [Ron04], who demonstrated the failure of SoI for Kt complexity. In contrast, the randomized setting poses a significant challenge, which we overcome using insights from [KK25], structural results about rKt implied by SoI, and techniques from meta-complexity [Oli19] and the theory of computational pseudorandomness [TV07].

*University of Warwick. Email: jinqiao.hu@warwick.ac.uk
†University of Haifa. Email: yahel.manor49@gmail.com
‡University of Warwick. Email: igor.oliveira@warwick.ac.uk

Contents

1 Introduction

1.1 Overview

The *Symmetry of Information* (SoI) principle is a fundamental result in the theory of Kolmogorov complexity [ZL70]. It states that for all n-bit strings x and y,

$$\mathsf{K}(x, y) = \mathsf{K}(y) + \mathsf{K}(x \mid y) \pm O(\log n).$$

The classical proof of SoI relies on an exhaustive search argument. In particular, the argument does not appear to extend to time-bounded settings.

As noted by Levin [Lev03], Kolmogorov speculated that SoI could serve as a test case for demonstrating that certain tasks inherently require exhaustive search. Over the last three decades, a series of works have strengthened and formalized Kolmogorov's intuition that the behavior of SoI in resource-bounded contexts is deeply connected to core problems in computational complexity. For example, [LM93, LW95] showed that the failure of SoI for K^t complexity implies $\mathsf{NP} \not\subseteq \mathsf{P}$, while its validity rules out the existence of cryptographic one-way functions. More recently, [GK22, Hir22] established that the failure of SoI for K^t implies the average-case hardness of NP, i.e., $\mathsf{DistNP} \not\subseteq \mathsf{AvgP}$. Similarly, [GKLO22] established that the failure of SoI for the probabilistic measure pK^t implies that $\mathsf{DistNP} \not\subseteq \mathsf{AvgBPP}$. For many other results relating SoI to complexity theory, including applications in areas such as learning theory and meta-complexity, we refer to [LR05, Lee06, Hir20, Hir21, Ila23, HN25, GHL+25] and references therein.

These results suggest two broad research directions:

(*i*) *Computational Characterizations of SoI.* While previous results uncovered deep links between SoI and complexity theory, currently there is no computational statement known to be *equivalent* to SoI for a time-bounded measure.

(*ii*) *Unconditional Failure of SoI.* Given the above connections, it would be highly significant to determine, unconditionally, whether SoI fails in a variety of resource-bounded settings.

Regarding (*i*), [HIL+23] proved that one-way functions exist if and only if SoI fails *on average* for the probabilistic measure pK^t, i.e., for most strings drawn from some polynomial-time samplable distribution. Similar results hold for the measures rK^t [HLN24] and pKt [HLO24]. More recently, [KK25] established an equivalence between a generalization of SoI and a corresponding complexity lower bound. However, it remains open whether the standard formulation of SoI for measures such as K^t and pK^t admits an exact complexity-theoretic characterization.

In the direction of (*ii*), [Ron04] proved unconditionally that SoI fails for Levin's Kt complexity, and [LP22] later showed that a stronger formulation of SoI for K^t does not hold. Beyond these, no other unconditional failures of SoI in the time-bounded setting are known. Given the difficulty of showing failure of SoI, some authors have considered instead relativized settings, and established the existence of oracle worlds where time-bounded variants of SoI do not hold [LR05, Lee06, GHL+25].

Since probabilistic notions of Kolmogorov complexity have played a significant role in several recent developments in time-bounded Kolmogorov complexity and meta-complexity (see, e.g., [Ila23, San23, HIL+23, HN23, LP23, Hir23, GK23, HLO24, HLN24, LS24, HKLO24, GK24, LORS24, HN25, KK25, LP25]), it would be particularly useful to understand the validity of SoI in the context of randomized computations. A natural attempt to adapt the argument of [Ron04] — which shows the failure of SoI for Kt — to the randomized setting of rKt runs into a significant obstacle. In more detail, in the deterministic setting, a brute-force method can construct, in time $2^n \cdot \mathsf{poly}(n)$, an n-bit string x_n with $\mathsf{Kt}(x_n) \approx n$. In contrast, it is unclear how to construct, in randomized time $2^n \cdot \mathsf{poly}(n)$, an n-bit string y_n with $\mathsf{rKt}(y_n) \approx n$ (see [LOS21] for related results).[1] The difficulty lies in the lack of a canonical (pseudodeterministic) output produced by a corresponding brute-force procedure, a feature that is essential for extending [Ron04]'s argument. Although there has been notable progress in the study of pseudodeterministic algorithms, current results are insufficient to overcome this barrier.[2]

Using an approach completely different from [Ron04], we are able to make progress on directions (*i*) and (*ii*). We prove, *unconditionally*, that SoI fails for rKt complexity with error term $\mathsf{poly}(\log n)$. We obtain a similar result for pKt. More generally, we *characterize* the failure of SoI for rKt in terms of the worst-case computational complexity of approximating rKt. Then, building on these results and the underlying techniques, we advance the study of the *average-case failure of SoI*, a direction that is both necessary and sufficient to establish the existence of one-way functions. Finally, we obtain near-optimal bounds in a setting where a longer conditional string appears in the SoI equation.

Conceptually, our results and techniques, which build on those of [Hir22, KK25] and related work, fully exploit the interplay between symmetry of information and complexity lower bounds in the rKt setting. In particular, symmetry of information can be used to derive lower bounds, while, conversely, such lower bounds yield the failure of symmetry of information.

Next, we describe our contributions in more detail.

[1] In other words, we need the randomized procedure to output a *fixed* string with this property with high probability.

[2] In more detail, we need a strong enough pseudodeterministic PRG to adapt [Ron04]'s argument to rKt. Existing unconditional results [OS17b, LOS21, CLO+23] do not provide the necessary PRG.

1.2 Results

The definition of each relevant time-bounded Kolmogorov complexity measure is reviewed in Section 2.2. Before formally stating our results, we also need to introduce the following notions.

For a function $e\colon \mathbb{N} \to \mathbb{N}$ and a Kolmogorov complexity measure κ (e.g., $\kappa = \mathsf{rKt}$), we say that *Symmetry of Information* (SoI) holds for κ with error $e(n)$ if for every large enough n and for all strings $x, y \in \{0,1\}^n$, we have

$$\kappa(x \mid y) \leq \kappa(x, y) - \kappa(y) + e(n).$$

For instance, using this terminology, we know that SoI holds for $\kappa = \mathsf{K}$ (time-unbounded Kolmogorov complexity) with error term $e(n) = O(\log n)$.

For functions $\alpha\colon \{0,1\}^* \to \mathbb{N}$ and $T, \gamma\colon \mathbb{N} \to \mathbb{N}$, we say that α can be approximated up to an additive term of order γ in randomized time T if there is a randomized algorithm A such that, for every large enough n and $x \in \{0,1\}^n$, $A(x)$ runs in time at most $T(n)$ and

$$\Pr_A\left[|A(x) - \alpha(x)| \leq \gamma(n)\right] \geq 1 - \frac{1}{n}.$$

This probability bound can be amplified using standard techniques.

We are now ready to state our results.

Theorem 1 (SoI for rKt Yields Algorithm for Estimating rKt)**.**
If SoI holds for rKt *with error term* $e(n)$*, then given* $x \in \{0,1\}^n$*, the value* $\mathsf{rKt}(x)$ *can be approximated up to an additive term of order* $\gamma(n) = O(e(n) \cdot \log n)$ *in randomized time* $T(n) = 2^{O(e(n)\cdot \log n)}$.

Theorem 1 and the unconditional quasi-polynomial time complexity lower bound for estimating rKt from [Oli19] immediately imply the following result.

Corollary 1 (SoI Fails for rKt with Polylogarithmic Error)**.**
For every constant $c \geq 1$*, there are infinitely many values of* n *and strings* $x, y \in \{0,1\}^n$ *such that* $\mathsf{rKt}(x \mid y) > \mathsf{rKt}(x, y) - \mathsf{rKt}(y) + (\log n)^c$.

Corollary 1 provides the first unconditional result showing that symmetry of information fails for a randomized notion of time-bounded Kolmogorov complexity.

Next, we show a converse to Theorem 1.

Theorem 2 (Algorithm for Estimating rKt Yields SoI for rKt)**.**
Let $T, \gamma\colon \mathbb{N} \to \mathbb{N}$ *be monotone functions. If given* $x \in \{0,1\}^n$*, the value* $\mathsf{rKt}(x)$ *can be approximated up to an additive term of order* $\gamma(n)$ *in randomized time* $T(n)$*, then SoI holds for* rKt *with error term* $e(n) = O(\log T(O(n)) + \gamma(O(n)) + \log^3 n)$.

We obtain the following immediate consequence from Theorem 1 and Theorem 2.

Corollary 2 (Equivalence Between SoI for rKt and Its Meta-Complexity)**.**
For any monotone function $T(n)$ *satisfying* $n \leq T(n) \leq 2^n$*, the following statements are equivalent:*[3]

1. *SoI holds for* rKt *with error* $e(n) = \widetilde{O}(\log T(O(n)))$.
2. *Given an input string* $x \in \{0,1\}^n$*, the value* $\mathsf{rKt}(x)$ *can be approximated up to an additive term of order* $\widetilde{O}(\log T(O(n)))$ *in time* $2^{\widetilde{O}(\log T(O(n)))}$.

[3] We write $a(n) = \widetilde{O}(b(n))$ to denote that $a(n) \leq b(n) \cdot \mathsf{poly}(\log n)$ for large n.

Corollary 2 provides a computational characterization of worst-case symmetry of information for a natural time-bounded Kolmogorov complexity measure.

Theorem 3 (SoI for pKt Yields Algorithm for Estimating pKt)**.**
If SoI holds for pKt *with error term* $e(n)$*, then given* $x \in \{0,1\}^n$ *and* $\varepsilon > 0$ *such that*

$$\varepsilon \geq \frac{C \cdot (e(n) + \log n)}{n},$$

where $C \geq 1$ *is an absolute constant, the value* $\mathsf{pKt}(x)$ *can be approximated up to an additive term* εn *in randomized time* $2^{O((\log n + e(n))/\varepsilon)}$.

Theorem 3 (using, say, $\varepsilon = 1/10$) and the unconditional quasi-polynomial time complexity lower bound for estimating pKt from [HLO24] immediately imply the following result.

Corollary 3 (SoI Fails for pKt with Polylogarithmic Error)**.**
For every constant $c \geq 1$*, there are infinitely many values of* n *and strings* $x, y \in \{0,1\}^n$ *such that* $\mathsf{pKt}(x \mid y) > \mathsf{pKt}(x,y) - \mathsf{pKt}(y) + (\log n)^c$.

Recall that [HLO24] showed that one-way functions exist if and only if SoI for pKt fails *on average*, i.e., there is a polynomial-time samplable distribution $\mathcal{D} = \{\mathcal{D}_n\}_{n\geq 1}$, where each $\mathcal{D}_n$ is supported over $\{0,1\}^n \times \{0,1\}^n$, such that with probability at least $1/\mathsf{poly}(n)$ over $(x,y) \sim \mathcal{D}_n$, $\mathsf{pKt}(x \mid y) > \mathsf{pKt}(x,y) - \mathsf{pKt}(y) + e(n)$, where $e(n) = \omega(\log n)$. Corollary 3 makes progress on this front by showing that SoI for pKt fails in the *worst case* with error $\mathsf{poly}(\log n)$.

These results motivate the investigation of the failure of SoI on average over a polynomial-time samplable distribution. We show that, for every constant $C \geq 1$, we can sample in polynomial time a pair (x,y) such that SoI fails with probability $1/\mathsf{poly}(n)$ with error parameter $e(n) = C \cdot \log n$, for infinitely many values of n.[4]

Theorem 4 (Average-Case Failure of SoI with Error $e(n) = O(\log n)$)**.**
Let $\kappa \in \{\mathsf{rKt}, \mathsf{pKt}\}$*. For every constant* $C \geq 1$*, there is a polynomial-time samplable distribution family* $\mathcal{D} = \{\mathcal{D}_n\}_{n\geq 1}$*, where each* $\mathcal{D}_n$ *is supported over* $\{0,1\}^n \times \{0,1\}^n$*, for which the following holds. There is a positive constant* k *such that, for infinitely many values of* n*, we have*

$$\Pr_{(x,y)\sim\mathcal{D}_n}[\kappa(x \mid y) \geq \kappa(x,y) - \kappa(y) + C \cdot \log n] \geq \frac{1}{n^k}.$$

These results leave open the possibility that SoI for rKt might hold with an error term $e(n)$ larger than $\mathsf{poly}(\log n)$, such as $e(n) = n^\varepsilon$ for some $\varepsilon > 0$. By Corollary 2, this is equivalent (up to $\mathsf{poly}(\log n)$ factors) to the existence of an algorithm that approximates rKt up to an additive term of order n^ε in time 2^{n^ε}.

Remark on $\mathsf{BPTIME}[T]$ **versus** $\mathsf{prBPTIME}[T]$ **lower bounds.** Let $s\colon \mathbb{N} \to \mathbb{N}$ with $1 \leq s(n) \leq n$ be a polynomial-time computable function. The algorithm for estimating rKt obtained from Theorem 1 establishes that the corresponding computational problem $\mathsf{Gap\text{-}MrKtP}[s-\gamma, s+\gamma]$ is in $\mathsf{prBPTIME}[T]$, for an appropriate function $T\colon \mathbb{N} \to \mathbb{N}$. Note that there might be strings x of intermediate complexity $s(|x|) - \gamma(|x|) < \mathsf{rKt}(x) < s(|x|) + \gamma(|x|)$ that are accepted by the algorithm with probability exponentially close to $1/2$. For this reason, we need to use lower bounds against the promise class $\mathsf{prBPTIME}[T]$ to

[4] Note the change in the order of quantifiers: given the constant C, there is a distribution $\mathcal{D} = \{\mathcal{D}_n\}_{n\geq 1}$ such that the statement holds. The time required to sample $(x,y) \sim \mathcal{D}_n$ is bounded by a polynomial that depends on C.

derive Corollary 1. In particular, we cannot rely on the quantitatively stronger lower bounds for Gap-MrKtP against $\mathsf{BPTIME}[2^{o(n)}]$ obtained in [Hir22].

We are able to make further progress on the failure of SoI in the setting where the conditional string can be significantly longer. More precisely, we say that *conditional Symmetry of Information* holds for rKt with error $e(n)$ if for every large enough length $n \in \mathbb{N}$, strings $x, y \in \{0,1\}^n$, and $w \in \{0,1\}^*$,[5]

$$\mathsf{rKt}(x \mid y, w) \leq \mathsf{rKt}(x, y \mid w) - \mathsf{rKt}(y \mid w) + e(n).$$

Note that the error term $e(n)$ in the inequality above does not depend on the length of w.

Theorem 5 (Failure of Conditional SoI with Linear Error)**.** *There is a constant $\varepsilon > 0$ such that conditional SoI with error $e(n) \leq \varepsilon \cdot n$ fails for* rKt.

In light of our results and techniques relating SoI and meta-complexity, it is also possible to obtain a corresponding unconditional lower bound on the complexity of estimating conditional rKt complexity.

Theorem 6 (Exponential Hardness of Estimating Conditional rKt)**.** *For every $0 < \alpha < \beta < 1$ there is a constant $\varepsilon > 0$ such that the following holds: There is no randomized algorithm running in time $2^{\varepsilon \cdot |x|} \cdot \mathsf{poly}(|w|)$ that accepts every pair (x, w) with $\mathsf{rKt}(x \mid w) \geq \beta \cdot |x|$ and rejects every pair (x, w) with $\mathsf{rKt}(x \mid w) \leq \alpha \cdot |x|$.*

We observe that the complexity of estimating conditional time-bounded Kolmogorov complexity by a brute-force procedure depends primarily on the length of the non-conditional string x (see, e.g., Lemma 3). Consequently, the aforementioned complexity lower bound is essentially optimal. We also note that the statement is incomparable to the lower bound against $\mathsf{BPTIME}[2^{o(n)}]$ from [Hir22] mentioned above, which does not require a conditional string but cannot be extended to $\mathsf{prBPTIME}[\cdot]$.

1.3 Techniques

We build on several previous works in meta-complexity, pseudorandomness, and time-bounded Kolmogorov complexity. First, we explain the techniques used to establish that symmetry of information fails for rKt in the worst-case (Theorem 1 and Corollary 1).

As explained above, it is not clear how to apply Ronneburger's technique to randomized computations, since the corresponding brute-force procedure is not pseudodeterministic. A natural approach is to use derandomization methods to obtain a pseudodeterministic construction of a string with large rKt complexity, as required in the argument. However, it seems that a direct implementation of this approach can only show failure of SoI for rKt under a derandomization hypothesis (see [HLO24]). To overcome this limitation, we take a different route and avoid relying on [Ron04]'s technique.

Conceptually, our approach is to show that, assuming SoI holds for rKt with a small error term, there exists a non-trivial randomized algorithm for estimating rKt complexity. We then invoke an existing unconditional lower bound on the complexity of estimating rKt [Oli19], obtaining a contradiction. A key subtlety is that the estimation algorithm must be correct for *all* input strings of bounded rKt complexity. Prior results showing that "SoI for a measure κ yields algorithms for estimating κ" did not guarantee this — they produced *two-sided* error algorithms that were correct on most strings, but not necessarily on all low-complexity

[5] Some results in the literature that consider conditional time-bounded Kolmogorov complexity explore that the running time is less than the length of the conditional string. This is not the case here, and we do not require random access to the conditional string.

strings (see, e.g., [LW95], [Hir22], or the algorithm obtained by combining [LP20] and [HIL+23]). Notably, [KK25] recently observed that algorithms with *one-sided* error can be obtained from a stronger form of SoI, the "chain rule". One of our main conceptual ideas is that, in the case of rKt complexity, the standard formulation of SoI already suffices to obtain the required algorithm.

Under the assumption that SoI holds for rKt complexity with error term $e(n) = \mathsf{poly}(\log n)$, we obtain two consequences that are central to our approach. We discuss them next.

1) Generation of Candidate Programs. First, we employ an idea from the investigation of search-to-decision reductions in time-bounded Kolmogorov complexity (see, e.g., [HKLO24]). For a string $x \in \{0,1\}^n$, we consider the rKt complexity of strings representing programs that encode x. In more detail, let $\Pi \in \{0,1\}^*$ be any string representing a randomized program that with high probability runs in time at most t_Π and outputs x, and such that $|\Pi| + \log t_\Pi \approx \mathsf{rKt}(x)$. Under the validity of SoI for rKt, it is possible to show that

$$\mathsf{rKt}(\Pi \mid x) = O(\log n + e(n)).$$

Consequently, given x, the string Π admits a short and time-efficient representation. In particular, we can produce a list of *candidate* programs for x by inspecting programs extracted from descriptions of bounded length. A key difficulty is that testing if a program in the list indeed offers a representation of x requires large running time.

In order to overcome the issue highlighted above, we show that, under SoI, there is always a program whose overall complexity is not much larger and that runs in bounded time. We explain this next.

2) A Structural Result About rKt. Recall that

$$\mathsf{rKt}(x) = \min_{\substack{\Pi \in \{0,1\}^* \\ t \in \mathbb{N}}} \left\{ |\Pi| + \lceil \log t \rceil \;\middle|\; \mathop{\mathbf{Pr}}_{r \sim \{0,1\}^t} [\Pi(r) \text{ outputs } x \text{ within } t \text{ steps}] \geq 2/3 \right\}.$$

In light of the inequality established in Item 1 above, we are interested in understanding if x admits a program Π of running time t_Π such that $|\Pi| + \log t_\Pi \approx \mathsf{rKt}(x)$ and, crucially, t_Π is small. If this is the case, we can run the program and *test* in bounded time if it offers a valid representation of x.

We show that, assuming SoI for rKt with error $e(n) = \mathsf{poly}(\log n)$, every n-bit string x admits a program Π with $\log t_\Pi = \mathsf{poly}(\log n)$ such that $(|\Pi| + \log t_\Pi) - \mathsf{rKt}(x) \leq \mathsf{poly}(\log n)$. In other words, x admits near-optimal representations in the sense of rKt where the running time contribution to the rKt bound is small.

Roughly speaking, the proof of this structural result employs a recursive approach where in each step we show that either the first half x^1 of the string or its second half x^2 (given x^1) admits a time-efficient representation. To argue this, we make use of the time bottleneck enforced by SoI when comparing $\mathsf{rKt}(x^1x^2)$ with $\mathsf{rKt}(x^1)$ and $\mathsf{rKt}(x^2 \mid x_1)$. We recurse on the less efficient part, and after at most $\log n$ steps, we obtain a time-efficient representation of the entire string that is not much longer than the best possible representation.

Crucially, in this approach we only need to invoke SoI $\log n$ times. This leads to significantly better parameters compared with the approach of [KK25] (see Section 5).

Given the results from Items 1 and 2 described above, it is not difficult to estimate the rKt complexity of a given string x, as explained next.

3) Approximating $\mathsf{rKt}(x)$. By Items 1 and 2, for every string $x \in \{0,1\}^n$, if $\mathsf{rKt}(x) \leq k$ there is a program Π such that:

(a) Π outputs x with high probability in time at most t_Π;

(b) $|\Pi| + \log t_\Pi \leq k + \mathsf{poly}(\log n)$;

(c) $\log t_\Pi \leq \mathsf{poly}(\log n)$;

(d) $\mathsf{rKt}(\Pi \mid x) = \mathsf{poly}(\log n)$.

Consequently, given any $x \in \{0,1\}^n$, we can decide within randomized quasi-polynomial time whether $\mathsf{rKt}(x) \leq k + \mathsf{poly}(\log n)$ as follows: generate (with high probability) a list S of all programs Π satisfying condition (d), and for each $\Pi \in S$ verify whether conditions (a)–(c) hold (for a candidate value of k). If some $\Pi \in S$ passes all tests, we conclude that $\mathsf{rKt}(x) \leq k + \mathsf{poly}(\log n)$.

The key observation is that, if $\mathsf{rKt}(x) \leq k$, then with high probability at least one program in S will pass the tests; whereas if $\mathsf{rKt}(x) \gg k + \mathsf{poly}(\log n)$, then with high probability none will. Therefore, Items (a)–(d) yield a quasi-polynomial-time algorithm that approximates $\mathsf{rKt}(x)$ to within an additive $\mathsf{poly}(\log n)$ term with high probability.

4) Hardness of Approximating $\mathsf{rKt}(x)$. In summary, the above argument shows that if SoI holds for rKt with error term $e(n) = \mathsf{poly}(\log n)$, then there exists a randomized quasi-polynomial-time algorithm for estimating rKt up to an additive error of order $\mathsf{poly}(\log n)$. However, [Oli19] proved that no algorithm of this complexity can distinguish strings with rKt complexity at most $n^{o(1)}$ from those with complexity at least $n - 1$. This contradiction establishes that SoI fails for rKt in the worst case.

The lower bound in [Oli19] relies on techniques from pseudorandomness [TV07] and on an indirect diagonalization argument. A quantitatively stronger bound would directly imply the failure of SoI for rKt with a larger error term. As noted in Corollary 2, achieving such a bound is equivalent to refuting weaker forms of SoI.

The proof of Theorem 2 relies on methods from computational pseudorandomness. It is a simple adaptation of existing techniques [Hir22, GK22], and we refer to Section 4 for the details.

For the proof of Theorem 3, we rely on an idea from [KK25]. By multiple applications of SoI, it is possible to prove that $\mathsf{pKt}(x)$ is approximately $\sum_{i=1}^{k} \mathsf{pKt}(x_i \mid x_{i-1} \dots x_1)$, where we write $x = x_k \dots x_1$. Thus we can reduce the task of approximating $\mathsf{pKt}(x)$ to that of approximating each value $\mathsf{pKt}(x_i \mid x_{i-1} \dots x_1)$, provided that SoI with bounded error holds for all the corresponding pairs $(x_i, x_{i-1} \dots x_1)$ of strings obtained from x in this way. The main advantage here is that the length of each x_i is considerably less than the length of the original string x, thus $\mathsf{pKt}(x_i \mid x_{i-1} \dots x_1)$ is easier to approximate. This allows us to show that pKt is easy to approximate under the assumption that SoI holds in the worst case.

The same idea can also be used as part of the proof of Corollary 1. However, this approach is not enough to achieve the stronger parameters in Theorem 1 and Corollary 2.

In order to establish the failure of SoI on average with error term $e(n) = O(\log n)$ (Theorem 4), we combine insights from the aforementioned proofs with techniques employed in [KK25] and [Oli19]. Below we highlight some conceptual differences and key ideas, referring to Section 6 for the technical details. For concreteness, we consider the case of rKt complexity. The argument for pKt is essentially the same.

Average-Case Failure of SoI (Theorem 4). We also proceed by contradiction, but this time we need to work under a weaker assumption: for any efficiently samplable distribution, SoI holds for most string pairs (x, y) (instead of for all pairs of strings). In particular, this weaker hypothesis does not yield an algorithm that estimates the complexity of an arbitrary input string, as employed in the previous proofs.

Instead, by adapting the idea behind the proof of Theorem 3, we are able to obtain an efficient algorithm that correctly estimates the rKt complexity of *most strings*. However, since we have limited control over the strings on which the algorithm succeeds, it is not possible to invoke existing unconditional lower bounds [Oli19], which require one-sided error algorithms for this task.

Inspecting the proof of the unconditional complexity lower bound, a key ingredient is the ability to break a pseudorandom generator under the assumption that rKt complexity can be approximated with one-sided error. This is then employed to efficiently compute a PSPACE-complete language L_{TV} [TV07] with special properties, such as *random-self-reducibility*.

A central idea in our proof is to analyse the queries the algorithm for L_{TV} needs to make to the algorithm approximating rKt. Roughly speaking, we argue that either these queries induce an efficiently samplable distribution of strings for which SoI fails with probability $1/\mathsf{poly}(n)$ (similarly to the sketch of the proof of Theorem 3 given above), or with overwhelming probability the queries are over strings for which we succeed in approximating rKt. Since the former case does not happen due to the assumption that SoI holds on average, we obtain that with high probability the queries are correctly answered.

Note that the win-win situation described above cannot be applied to each input string x for L_{TV}, since the first case employs that the queries are produced by a samplable distribution. But by considering a random input string x as part of the analysis, we are able to leverage this idea to show that the PSPACE-complete language L_{TV} can be efficiently computed on most inputs. Then, using that this language is also random-self-reducible, we get that L_{TV} can be efficiently computed in the worst case. In turn, given that this language is complete for PSPACE, it follows that PSPACE is contained in BPP.

Finally, adapting a construction from [KK25], we show that if PSPACE is contained in BPP and SoI holds on average, then given 1^n one can efficiently construct a string of large rKt complexity. But this is impossible by the definition of rKt, i.e., a string that is easy to construct has small rKt complexity. This contradiction completes the proof.

On the Connection to One-Way Functions and the Role of Adaptivity. We remark that the bounded error term $e(n) = C \cdot \log n$ is used twice in the argument sketched above. Firstly, it guarantees that queries to the algorithm estimating rKt can be efficiently generated, and consequently, we obtain an efficiently samplable distribution of pairs of strings over which SoI is considered. Secondly, during the final step of the proof, the bound is used to provide a polynomial-time algorithm that constructs a string of large rKt complexity, resulting in a contradiction.

There is some slackness in the second application of the bound $e(n) = C \cdot \log n$. The bottleneck related to $e(n)$ is in the first use of the bound. In more detail, as the constant C grows, the running time needed to produce the distribution of queries (and corresponding distribution over pairs of strings) also grows. This is because the queries can be *adaptive*, i.e., to obtain the next query we need to produce a candidate answer to previous queries. In turn, in our proof the running time of the algorithm that attempts to correctly answer each query is of the form $t(n) = \mathsf{poly}(n^C)$.

The adaptivity of the queries is a result of the procedure employed to show that L_{TV} is in BPP using an algorithm that approximates rKt complexity. It relies on another special property of L_{TV} called *downward-self-reducibility*. This leads to *adaptive* queries to the oracle approximating rKt during the computation of L_{TV} on an input y, where the degree of adaptivity corresponds to the length of y.[6] Moreover, there is evidence that adaptivity might be essential (see [HW20, SS22] and references therein).

In contrast, for the connection to one-way functions, one needs to sample in polynomial time a distribution for which SoI fails on average with error term $e(n) = \omega(\log n)$.

Finally, we discuss the proof of Theorem 5 and the role of measuring the SoI error and time complexity as a function of the length of the non-conditional string only. The proof of Theorem 6 employs similar ideas,

[6] In more detail, the algorithm for L_{TV}, on an input string y of length n, first constructs a sequence of n circuits $C_1, \ldots, C_n$, where each circuit C_i decides L_{TV} on inputs of length i, then evaluates the final circuit C_n on y. The i-th stage of the construction relies on the circuit C_{i-1} and issues a new batch of queries that need to be answered before producing C_i. We refer to [IW01, OS17a] for more details.

and we refer the reader to Section 7.

Stronger Failure of Conditional SoI (Theorem 5). We consider here a weaker setting of parameters for simplicity; the near-optimal bound can be obtained by optimizing the argument. In order to derive a contradiction, assume that conditional SoI holds for rKt with error $e(n) = n^{o(1)}$. We note that, by exploring the connection to meta-complexity, this allows us to estimate $\mathsf{rKt}(y \mid z)$ in time sub-exponential in $|y|$. A key idea is to revisit one of the main lemmas in [Oli19]. The lemma states that $L_{\mathsf{TV}} \in \mathsf{BPP}^{\mathsf{GapMrKtP}[n^{\varepsilon}, n-1]}$. In particular, we can efficiently decide the PSPACE-complete language L_{TV} if rKt can be estimated in polynomial time, or more generally, a time-$T(n)$ algorithm for the meta-computational problem allows us to compute L_{TV} in time $T(\mathsf{poly}(n))$. This lemma is shown by analysing $\mathsf{rKt}(y)$, where $y \triangleq G^{L_{\mathsf{TV}}}(z) \in \{0,1\}^n$, $z \sim \{0,1\}^{n^{\Omega(1)}}$ is a random seed, and G is a pseudorandom generator with a suitable reconstruction routine.

An important insight is to analyse instead $\mathsf{rKt}(y \mid z)$, with y as above, and to show how to iteratively *bootstrap* the upper bound on $\mathsf{rKt}(y \mid z)$ and on the complexity of computing L_{TV} (*without an oracle*) using

- the reduction (via breaking pseudorandomness) behind the inclusion $L_{\mathsf{TV}} \in \mathsf{BPP}^{\mathsf{GapMrKtP}[n^{\varepsilon}, n-1]}$;
- and that conditional rKt can be estimated in time sub-exponential in the length of the first string.

For instance, combining these two items, we can compute L_{TV} in time $2^{n^{o(1)}}$. Consequently, it is possible to derive a stronger version of the first bullet where it is enough to have an oracle that distinguishes complexity "$n^{o(1)}$ versus $n-1$" instead of "n^{ε} versus $n-1$", since with a faster algorithm for L_{TV} each output string $G^{L_{\mathsf{TV}}}(z)$ admits a better upper bound on its conditional rKt complexity.[7] Moreover, by a standard observation that holds in the setting of breaking a generator, we can truncate $y = G^{L_{\mathsf{TV}}}(z)$ and consider only a prefix of this string, which allows us to reduce the length of y while maintaining the length of the conditional string z (the random seed). This is desirable since the upper bound on the running time in the second item only depends on the length of the first string.

Crucially, once we get a better upper bound on $\mathsf{rKt}(y \mid z)$ and on the complexity of computing L_{TV} without an oracle using one application of the aforementioned idea, we can apply the same argument *again* to achieve further gains. In a bit more detail, once we have a better upper bound on the complexity of computing L_{TV}, we derive a better upper bound on the conditional Kolmogorov complexity of the strings produced by the generator when given oracle access to L_{TV}. In turn, this allows us to truncate the output of the generator even more while still obtaining a correct distinguisher using the algorithm that estimates conditional complexity (and as explained above, shorter strings lead to smaller running times in the second item). (This informal exposition omits a relevant point: the analysis of the pseudorandom generator G and of its reconstruction procedure should remain valid even if the random seed is exposed. It is possible to verify that this is the case; see Lemma 16.)

By a careful implementation of this strategy, it is possible to fully bootstrap the original lemma from [Oli19] and to compute L_{TV} in polynomial time *without oracle access*.[8] Perhaps interestingly, the iterative applications of the bootstrapping argument described above take place at the *meta level*, i.e., during the analysis of the conditional rKt complexity of the involved strings, and not at the *algorithmic level* itself.

Once we obtain that L_{TV} is easy to compute and a corresponding upper bound for PSPACE, with some additional work it is possible to derive a contradiction using the initial assumption that conditional SoI holds. For more details about the proof of Theorem 5, see Section 7.

[7] Note that an oracle that estimates conditional complexity can also be used to estimate complexity by setting the conditional string to be the empty string.

[8] There is a certain parallel between this approach and the Learning Speedup Lemma from [OS17a] and its proof, though the technical details are different.

Acknowledgements. We thank Valentine Kabanets and Zhenjian Lu for discussions on symmetry of information, and Shuichi Hirahara and Rahul Santhanam for discussions on the role of adaptivity in pseudorandomness and reconstruction routines. We also thank the anonymous STOC reviewers for useful comments about the presentation. This work was supported in part by the UKRI Frontier Research Guarantee Grant EP/Y007999/1 and the Centre for Discrete Mathematics and its Applications (DIMAP) at the University of Warwick.

2 Preliminaries

2.1 Notation

We denote the empty string by ϵ, and the length of a string x by $|x|$. For strings $x \in \{0,1\}^n$ and $y \in \{0,1\}^m$, their concatenation is written as $x \circ y \in \{0,1\}^{n+m}$.

If $x \sim \mathcal{D}$, then x is sampled from the distribution $\mathcal{D}$. For any a in the support of $\mathcal{D}$, $\mathcal{D}(a)$ denotes its probability. We write $\mathcal{U}_\ell$ for the uniform distribution over $\{0,1\}^\ell$. We say that a distribution family $\mathcal{D} = \{\mathcal{D}_n\}_{n\geq 1}$ can be sampled in time $T(n)$ if there is a randomized algorithm $A(1^n)$ running in time at most $T(n)$ such that, for every $n \geq 1$, the output of A on input 1^n is distributed exactly as $\mathcal{D}_n$.

2.2 Time-Bounded Kolmogorov Complexity

We fix a time-efficient universal machine U, and define time-bounded Kolmogorov complexity with respect to U (see [LV19] for the required background in Kolmogorov complexity). In other words, whenever we say that a program Π outputs x in t steps, formally, this means that $U(\Pi) = x$ when U runs for at most t steps on input Π. Our results are robust with respect to encoding choices.

Kt Complexity. For strings $x, y \in \{0,1\}^*$, the (Levin) *time-bounded Kolmogorov complexity of x conditioned on y* [Lev84] is defined as

$$\mathsf{Kt}(x \mid y) \triangleq \min_{\substack{\Pi\in\{0,1\}^* \\ t\in\mathbb{N}}} \{|\Pi| + \lceil \log t \rceil \mid \Pi(y) \text{ outputs } x \text{ within } t \text{ steps}\}.$$

We define $\mathsf{Kt}(x)$ as $\mathsf{Kt}(x \mid \epsilon)$, where ϵ denotes the empty string.

rKt Complexity. Similarly, for $x, y \in \{0,1\}^*$ and $0 < \lambda \leq 1$, the *λ-randomized time-bounded Kolmogorov complexity of x conditioned on y* [Oli19] is defined as

$$\mathsf{rKt}_\lambda(x \mid y) = \min_{\substack{\Pi\in\{0,1\}^* \\ t\in\mathbb{N}}} \left\{ |\Pi| + \lceil \log t \rceil \;\middle|\; \mathop{\mathbf{Pr}}_{r\sim\{0,1\}^t} [\Pi(y, r) \text{ outputs } x \text{ within } t \text{ steps}] \geq \lambda \right\}.$$

We omit the subscript λ when $\lambda = 2/3$.

pKt Complexity. For $x \in \{0,1\}^*$ and $0 < \lambda \leq 1$, the *λ-probabilistic time-bounded Kolmogorov complexity* of x [HLO24], denoted by $\mathsf{pKt}_\lambda(x)$, is defined to be the minimum $k \in \mathbb{N}$ such that with probability at least λ over $r \sim \{0,1\}^{2^k}$, there exist a program $\Pi \in \{0,1\}^*$ and a time bound $t \in \mathbb{N}$ such that $|\Pi| + \lceil \log t \rceil \leq k$ and $\Pi(r)$ outputs x within t steps. Equivalently

$$\mathsf{pKt}_\lambda(x) \triangleq \min \left\{ k \in \mathbb{N} \;\middle|\; \mathop{\mathbf{Pr}}_{r\sim\{0,1\}^{2^k}} [\mathsf{Kt}(x \mid r) \leq k] \geq \lambda \right\}.$$

We omit the subscript λ when $\lambda = 2/3$. This definition can be extended to *conditional* probabilistic Kolmogorov complexity in the natural way.

For technical reasons, we also need the following definition.

$\mathsf{rK}^{t,\mathcal{O}}$ Complexity. For $x \in \{0,1\}^*$, $t \in \mathbb{N}$, and an oracle $\mathcal{O} \subseteq \{0,1\}^*$, the *randomized t-time-bounded Kolmogorov complexity of x* is defined as

$$\mathsf{rK}^{t,\mathcal{O}}(x) = \min_{\Pi \in \{0,1\}^*} \left\{ |\Pi| \;\middle|\; \Pr_{r \sim \{0,1\}^t} \left[\Pi^{\mathcal{O}}(r) \text{ outputs } x \text{ within } t \text{ steps} \right] \geq \frac{2}{3} \right\}$$

where $\Pi^{\mathcal{O}}$ denotes the computation of Π with oracle access to $\mathcal{O}$.

2.3 Useful Results

The following results show that the complexity measures rKt_λ and pKt_λ are robust with respect to the probability threshold λ.

Lemma 1 (Success Amplification for rKt [LO21])**.** *For any string $x \in \{0,1\}^n$, $y \in \{0,1\}^*$ and $0 < \lambda \leq 1$,*

$$\mathsf{rKt}(x \mid y) \leq \mathsf{rKt}_\lambda(x \mid y) + O(\log(1/\lambda)).$$

Lemma 2 (Success Amplification for pKt [HLO24])**.** *For any string $x \in \{0,1\}^n$, $y \in \{0,1\}^*$, and $0 \leq \alpha < \beta \leq 1$, we have*

$$\mathsf{pKt}_\beta(x \mid y) \leq \mathsf{pKt}_\alpha(x \mid y) + O(\log(q/\alpha) + \log n),$$

where $q \triangleq \ln(1/(1-\beta))$.

It is not difficult to estimate rKt and pKt in exponential time. More precisely, using standard techniques (see [Oli19]), it is possible to establish the following results.[9]

Lemma 3 (Estimating rKt in Exponential Time)**.** *There exists a randomized algorithm $\mathsf{A} : \{0,1\}^* \times \{0,1\}^* \to \mathbb{N}$, such that for any string $x \in \{0,1\}^n$ and $y \in \{0,1\}^m$, $\mathsf{A}(x,y)$ runs in time $2^{O(n)} \cdot \mathsf{poly}(m)$, and*

$$\Pr_{\mathsf{A}} [\mathsf{rKt}(x \mid y) - O(1) \leq \mathsf{A}(x,y) \leq \mathsf{rKt}(x \mid y)] \geq 1 - 2^{-n^2}. \tag{1}$$

Lemma 4 (Estimating pKt in Exponential Time)**.** *There exists a randomized algorithm $\mathsf{A} : \{0,1\}^* \times \{0,1\}^* \to \mathbb{N}$, such that for any string $x \in \{0,1\}^n$ and $y \in \{0,1\}^m$, $\mathsf{A}(x,y)$ runs in time $2^{O(n)} \cdot \mathsf{poly}(m)$, and*

$$\Pr_{\mathsf{A}} [\mathsf{pKt}(x \mid y) - O(1) \leq \mathsf{A}(x,y) \leq \mathsf{pKt}(x \mid y)] \geq 1 - 2^{-n^2}. \tag{2}$$

Finally, the following simple lemma will be useful.

Lemma 5 (Padding)**.** *There exists a constant C such that the following holds. Let $\kappa \in \{\mathsf{rKt}, \mathsf{pKt}\}$. Then for any $n \in \mathbb{N}$ and any strings $x, y \in \{0,1\}^*$ satisfying $|x|, |y| \leq n$, we have*

$$|\kappa(x \mid y) - \kappa(x \circ 1^{n-|x|} \mid y \circ 1^{n-|y|})| \leq C \cdot \log n.$$

[9] We observe that the additive error $O(1)$ appearing in the approximation in Lemma 3 and Lemma 4 depends on details of the computational model. More precisely, this overhead is caused by the need to consider the complexity measure with respect to different probability thresholds (e.g., $\mathsf{rKt}_{\lambda_1}(x)$ and $\mathsf{rKt}_{\lambda_2}(x)$ with $\lambda_1 = 2/3$ and $\lambda_2 = 3/4$) and the corresponding difference in complexity. We note that employing a computational model for which the additive approximation overhead is $O(\log n)$ instead of $O(1)$ does not affect our results.

2.4 Pseudorandomness

We will need the following pseudorandom generator construction.

Lemma 6 (See, e.g., [Hir22] and [HIL+23, Lemma 26])**.** *There exists a polynomial p and a function family $G = \{G_{n,m}\}$ such that, for all sufficiently large $n, m, t \in \mathbb{N}$ such that $m \leq 2n$ and $n \leq t \leq 2^n$,*

$$G_{n,m} \colon \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m,$$

and for every $x \in \{0,1\}^n$ and any function $D \colon \{0,1\}^m \times \{0,1\}^t \to \{0,1\}$, if

$$\left| \Pr_{\substack{z \sim \{0,1\}^d \\ w' \sim \{0,1\}^t}} \left[D(G_m(x;z);w') = 1 \right] - \Pr_{\substack{w \sim \{0,1\}^m \\ w' \sim \{0,1\}^t}} \left[D(w;w') = 1 \right] \right| \geq \frac{1}{m},$$

then

$$\mathsf{rK}^{p(n,t),D}(x) \leq m + O(\log^3 n).$$

Here, $d = O(\log^3 n)$ and $G_{n,m}$ can be computed in time $\mathsf{poly}(n)$.

2.5 The Trevisan-Vadhan PSPACE-Complete Language

Definition 1 (Downward Self Reducibility)**.** We say that a language L is *downward self reducible* if there is a deterministic polynomial-time algorithm A such that for every n and all $x \in \{0,1\}^n$, $\mathsf{A}^{L_{n-1}}(x) = L_n(x)$.

Definition 2 (Random Self Reducibility)**.** We say that a language L is $\delta(n)$-*random-self-reducible* if there is a probabilistic polynomial-time algorithm $\mathsf{Cor}^{(\cdot)}$ such that, for any function $g : \{0,1\}^n \to \{0,1\}$ satisfying $\Pr_{x \sim \mathcal{U}_n}[g(x) = L(x)] \geq 1 - \delta(n)$, $\Pr_{\mathsf{Cor}}[\mathsf{Cor}^g(x) = L(x)] \geq 5/6$ for any $x \in \{0,1\}^n$. We say L is *random-self-reducible* if it is $\frac{1}{p(n)}$-random-self-reducible for some polynomial $p(n)$.

Lemma 7 ([TV07])**.** *There exists a PSPACE-complete language L_{TV} that is both downward self reducible and random self reducible.*

We will use such L_{TV} in many different places throughout this paper.

3 SoI to Meta-Complexity: Proof of Theorem 1

The proof of Theorem 1 is based on the following lemma showing that, under SoI, $\mathsf{rK}^{2^{O(e(n))}}(x)$ is approximately equal to $\mathsf{rKt}(x)$. This lemma allows us to approximate rK^t instead of rKt, and thus we only need to consider efficient programs instead of arbitrary programs.

Lemma 8. *Suppose that SoI holds for rKt with error term $e(n)$. Then there is $t \leq 2^{O(e(n)+\log n)}$ such that for every string $x \in \{0,1\}^n$ it holds that $\mathsf{rK}^t(x) \leq \mathsf{rKt}(x) + O(e(n) \cdot \log n)$.*

Proof. We start with the following observation. Let $x \in \{0,1\}^r$ and $y \in \{0,1\}^k$ such that $r + k \leq n$ be arbitrary strings, and let us split x into two halves x_1, x_2. Let m_1, m_2 be programs that generate x_1 given y and x_2 given x_1, y and witness the values $\mathsf{rKt}(x_1 \mid y), \mathsf{rKt}(x_2 \mid x_1, y)$, respectively. We denote their

respective running times by t_1, t_2. We note that we can generate x given y by running m_1 and then m_2. This employs a program of length $|m_1| + |m_2| + O(\log n)$ and running time $O(t_1 + t_2)$. Thus, we get that

$$\begin{aligned}\mathsf{rKt}(x \mid y) &\leq |m_1| + |m_2| + \log(t_1 + t_2) + O(\log n) \\ &\leq \mathsf{rKt}(x_1 \mid y) + \mathsf{rKt}(x_2 \mid x_1, y) - \log \min(t_1, t_2) + O(\log n).\end{aligned}$$

On the other hand, by SoI it holds that

$$\begin{aligned}\mathsf{rKt}(x \mid y) &\geq \mathsf{rKt}(x_1, x_2, y) - \mathsf{rKt}(y) \\ &\geq \mathsf{rKt}(x_2 \mid x_1, y) + \mathsf{rKt}(x_1, y) - \mathsf{rKt}(y) - e(n) \\ &\geq \mathsf{rKt}(x_2 \mid x_1, y) + \mathsf{rKt}(x_1 \mid y) - 2e(n).\end{aligned}$$

Combining both inequalities, it follows that

$$\begin{aligned}\mathsf{rKt}(x_1 \mid y) + \mathsf{rKt}(x_2 \mid x_1, y) - 2e(n) &\leq \mathsf{rKt}(x_1 \mid y) + \mathsf{rKt}(x_2 \mid x_1, y) - \log \min(t_1, t_2) + O(\log n) \\ \log \min(t_1, t_2) &\leq 2e(n) + O(\log n).\end{aligned}$$

That is, either m_1 or m_2 run in time less than $2^{2e(n)} \cdot \mathsf{poly}(n)$.

Given $x \in \{0,1\}^n$, we construct a program that witnesses the upper bound on $\mathsf{rK}^t(x)$. The program contains a partition of x into $\log n$ contiguous parts and $\log n$ sub-programs. The i-th program generates the i-th part of the partition, given all the parts before it. To construct the partition, we initially divide x into two (almost) equal parts x_1, x_2. By the observation appearing above, one of the strings has a program that runs in time $2^{2e(n)} \cdot \mathsf{poly}(n)$. We put this part into the partition and its program, and continue to divide recursively the other part by the same procedure. After $\log n$ steps, we remain with a part of length $O(1)$ that we add to the partition, and take some $O(1)$ length and time program for it. By amplifying the success probabilities of each of the programs to $1 - \frac{1}{3n}$, we can get that the program returns the string x with probability at least $\frac{2}{3}$.

It remains to bound the running time of the program and its description length. It is easy to see that the running time of each sub-program is at most $2^{2e(n)} \cdot \mathsf{poly}(n)$ (even after amplification) and thus the total running time is at most $2^{O(e(n))} \cdot \mathsf{poly}(n)$. Regarding the description length, let $x_1, \ldots, x_{\log n}$ be the partition of x, and let $m_1, \ldots, m_{\log n}$ be the sub-programs. Then, the overall length of the sub-programs is

$$\begin{aligned}\sum_{i=1}^{\log n} |m_i| &\leq \sum_{i=1}^{\log n} \mathsf{rKt}(x_i \mid x_1 \ldots x_{i-1}) \\ &\leq \mathsf{rKt}(x) + e(n) \cdot \log n.\end{aligned}$$

Due to its recursive description, the partition can be described using additional $O(\log n)$ bits, and thus the total length of the program is at most $\mathsf{rKt}(x) + O(e(n) \cdot \log n)$, as desired. □

Before describing the algorithm that estimates rKt, we observe the following fact. One does not have to go over all strings to find an (almost) optimal description for a string x, but rather it is sufficient to just go over the strings that can be generated from x in bounded time. In more detail, let $x \in \{0,1\}^n$ be a string, and let m be a program that witnesses $\mathsf{rK}^t(x) \leq \mathsf{rKt}(x) + O(e(n) \cdot \log n)$ for $t \leq 2^{O(e(n)+\log n)}$. Then it holds that

$$\begin{aligned}\mathsf{rKt}(m \mid x) &\leq \mathsf{rKt}(m, x) - \mathsf{rKt}(x) + e(n) \\ &\leq |m| + \log t + O(\log n) - \mathsf{rKt}(x) + e(n) \\ &\leq \mathsf{rK}^t(x) - \mathsf{rKt}(x) + O(e(n) + \log n) \\ &\leq O(e(n) \cdot \log n),\end{aligned}$$

where the second inequality holds as there exists a program that prints m then runs m to produce x.

We now describe the algorithm for estimating $\mathsf{rK}^t(x)$.

1. Given $x \in \{0,1\}^n$, the algorithm runs all programs of length $O(e(n) \cdot \log n)$ given x. It repeats each program n times, generating $2^{O(e(n)\cdot \log n)}$ strings. We interpret the strings as programs.
2. The algorithm runs each program $O(n)$ times, each run for at most $2^{O(e(n)\cdot \log n)}$ steps. For a given program, if more than $\frac{3}{4}$ of the runs output x, the algorithm stores this description of x.
3. The algorithm returns the length of the minimal description of x found in the last step.

It is easy to see that the running time of the algorithm is at most $2^{O(e(n)\cdot \log n)}$. We now prove its correctness. First, note that if a description of x has probability less than $\frac{2}{3}$ of generating x, then the probability that it is accepted in Item 2 is at most 2^{-2n} by a Chernoff bound. By a union bound, we get that, with probability at least $1 - 2^{-n}$, the minimal description that the algorithm finds is a valid description of x that runs in time $2^{O(e(n)\cdot \log n)}$. Consequently,

$$\mathsf{rK}^t(x) \geq \mathsf{rKt}(x) - \log t = \mathsf{rKt}(x) - O(e(n) \cdot \log n).$$

On the other hand, let m be a program description that witnesses $\mathsf{rK}^t_{1-2^{-n}}(x) \leq \mathsf{rKt}(x) + O(e(n) \cdot \log n)$ for $t \leq 2^{O(e(n)+\log n)}$. This program exists by Lemma 8 by amplifying the success probability from $\frac{2}{3}$ to $1 - 2^{-n}$. By the above calculation, we have that $\mathsf{rKt}(m \mid x) \leq O(e(n) \cdot \log n)$ and therefore the algorithm goes over the program m' that generates m given x. As the algorithm runs m' a total of $O(n)$ times, it succeeds in generating m with probability at least $1 - 2^{-n}$. Then, it is easy to see that this program passes step Item 2 with high probability. Thus the output is at most $\mathsf{rKt}(x) + O(e(n) \cdot \log n)$, as desired.

4 Meta-Complexity to SoI: Proof of Theorem 2

Since the argument is similar to other proofs in the literature (see, e.g., [Hir22] and [HKLO24]), here we only provide a sketch and a discussion of the relevant parameters.

Let $T, \gamma \colon \mathbb{N} \to \mathbb{N}$ be monotone functions. Assume there exists a randomized algorithm $A(v)$ that runs in time at most $T(\ell)$ on any input $v \in \{0,1\}^\ell$ and approximates $\mathsf{rKt}(v)$ up to an additive term of order $\gamma(\ell)$. We aim to show that, for every $x, y \in \{0,1\}^n$,

$$\mathsf{rKt}(x \mid y) \leq \mathsf{rKt}(x, y) - \mathsf{rKt}(y) + e(n), \tag{3}$$

where $e(n) = O(\log T(\ell) + \gamma(\ell) + \log^3 n)$ and $\ell = O(n)$. We can assume that for every large enough ℓ, $\gamma(\ell) \leq \varepsilon \cdot \ell$ and $T(\ell) \leq 2^{\varepsilon \cdot \ell}$, for a small enough constant $\varepsilon > 0$, since otherwise the inequality is trivial.

We assume without loss of generality that $\gamma(\ell) \geq 1$. For $\ell \geq 1$, consider the sets

$$\begin{aligned} S^{\mathsf{yes}}_\ell &\triangleq \left\{ v \in \{0,1\}^\ell \mid \mathsf{rKt}(v) \leq \ell - 4 \cdot \gamma(\ell) \right\}, \\ S^{\mathsf{no}}_\ell &\triangleq \left\{ v \in \{0,1\}^\ell \mid \mathsf{rKt}(v) \geq \ell - 1 \right\}. \end{aligned}$$

Note that these sets can be separated by algorithm A in randomized time $T(\ell)$ for any large enough input length ℓ.

Fix $x, y \in \{0,1\}^n$. Let m and m' be parameters to be fixed later. Let $G = \{G_{n,m}\}$ be the generator from Lemma 6 with corresponding randomness parameter $d = O(\log^3 n)$ and output length m. Consider the following distributions supported over strings of length $\ell \triangleq m + m'$:

$$\begin{array}{lcccll} \mathcal{D}_1 & \triangleq & G_{n,m}(x,z) & \circ & G_{n,m'}(y,z') & \text{where } z, z' \sim \mathcal{U}_d, \\ \mathcal{D}_2 & \triangleq & w & \circ & G_{n,m'}(y,z') & \text{where } w \sim \mathcal{U}_m \text{ and } z' \sim \mathcal{U}_d, \\ \mathcal{D}_3 & \triangleq & w & \circ & w' & \text{where } w \sim \mathcal{U}_m \text{ and } w' \sim \mathcal{U}_{m'}. \end{array}$$

Jumping ahead, we will select m and m' in a way such that distributions $\mathcal{D}_1$ and $\mathcal{D}_2$ are distinguished by any uniform randomized test that separates the sets S^{yes}_ℓ and S^{no}_ℓ defined above. In turn, this will allow us to distinguish distributions $G_{n,m}(x, \mathcal{U}_d)$ and $\mathcal{U}_m$ when given y and access to A, which together with our choice of parameters and properties of the generator, will allow us to establish Equation (3). We provide more details next.

By a standard hybrid argument, in order to distinguish distributions $\mathcal{D}_1$ and $\mathcal{D}_2$, it is enough to show that $\mathcal{D}_2$ and $\mathcal{D}_3$ are indistinguishable, while $\mathcal{D}_1$ and $\mathcal{D}_3$ can be distinguished. The notion of indistinguishability considered below refers to randomized uniform algorithms running in time at most $\mathsf{poly}(T(\ell))$ and possibly making use of $O(\log n)$ bits of advice.

Choice of m' and indistinguishability of $\mathcal{D}_2$ and $\mathcal{D}_3$. Note that it is enough to establish the indistinguishability of $G_{n,m'}(y, \mathcal{U}_d)$ and $\mathcal{U}_{m'}$. By Lemma 6, indistinguishability with respect to a randomized procedure D running in time $T(m')$ holds provided that $\mathsf{rK}^{\mathsf{poly}(T(m')),D}(y) > m' + O(\log^3 n)$. Oracle access to D can be eliminated if D is uniform and requires at most $O(\log n)$ bits of advice. Since $\mathsf{rKt}(y) \leq \mathsf{rK}^{\mathsf{poly}(T(m'))}(y) + O(\log T(m'))$, with respect to uniform algorithms running in time $\mathsf{poly}(T(m'))$, we can guarantee indistinguishability if we set

$$m' \triangleq \mathsf{rKt}(y) - O(\log^3 n) - O(\log T(m')).$$

For m' to be well defined, we must have $\mathsf{rKt}(y) \geq C \cdot \log^3 n + C \cdot \log T(m')$ for a large enough C. This can be assumed without loss of generality, since otherwise Equation (3) already holds for the desired bound $e(n)$. In addition, note that $m' \leq 2n$.

Choice of m and distinguishability of $\mathcal{D}_1$ and $\mathcal{D}_3$. First, a simple argument shows that every string $u \in \mathsf{Support}(\mathcal{D}_1)$ satisfies $\mathsf{rKt}(u) \leq \mathsf{rKt}(x,y) + 2d + O(\log n) = \mathsf{rKt}(x,y) + O(\log^3 n)$. On the other hand, with probability $\Omega(1)$, a string $u \sim \mathcal{D}_3$ satisfies $\mathsf{rKt}(u) \geq m + m' - 1 = \ell - 1$. Therefore, the sets S^{yes}_ℓ and S^{no}_ℓ distinguish these distributions if $\mathsf{rKt}(x,y) + O(\log^3 n) \leq \ell - 4 \cdot \gamma(\ell)$. Since $\ell = m + m'$, using the definition of m' and the monotonicity of γ and T, it is enough to set

$$m \triangleq \mathsf{rKt}(x,y) - \mathsf{rKt}(y) + O(\log^3 n) + O(\gamma(O(n))) + O(\log T(O(n))).$$

Under our assumption that $T(\ell) \leq 2^{\varepsilon \cdot \ell}$ and $\gamma(\ell) \leq \varepsilon \cdot \ell$, we get that $m = O(n)$, and consequently $\ell = O(n)$.

It follows from the above discussion that distributions $\mathcal{D}_1$ and $\mathcal{D}_2$ can be distinguished by running $A(u)$ and comparing the output value with a fixed threshold. The latter can be specified using $O(\log n)$ bits of advice.

Next, we argue that with this we can distinguish the distributions $G_{n,m}(x, \mathcal{U}_d)$ and $\mathcal{U}_m$, given y and access to A. To see this, it is enough to consider the randomized distinguisher D_y that, given a string

$v \in \{0,1\}^m$, produces the string $u = v \circ G_{n,m'}(y, \mathcal{U}_d)$ and invokes the aforementioned distinguisher for $\mathcal{D}_1$ and $\mathcal{D}_2$. By a simple reduction, D_y distinguishes $G_{n,m}(x, \mathcal{U}_d)$ and $\mathcal{U}_m$. It follows from Lemma 6 that

$$\mathsf{rK}^{\mathsf{poly}(n), D_y}(x) \leq m + O(\log^3 n).$$

Moreover, since D_y can be computed when given access to y by running A on inputs of length $O(n)$, we get that

$$\mathsf{rKt}(x \mid y) \leq m + O(\log^3 n) + O(\log T(O(n))).$$

Finally, Equation (3) follows from this upper bound and the definition of m. This completes the proof.

5 The Case of pKt Complexity: Proof of Theorem 3

For the sake of contradiction, assume that SoI holds for pKt with error term $e(n)$. By padding strings of length less than n, it is easy to see that SoI also holds for strings of length at most n with error term $e(n) + O(\log n)$. That is, for any strings $x \in \{0,1\}^*$, $y \in \{0,1\}^*$, where $|x| \leq n$ and $|y| \leq n$, we have

$$\mathsf{pKt}(x, y) \geq \mathsf{pKt}(x \mid y) + \mathsf{pKt}(y) - e(n) - O(\log(n)).$$

Since we have $\mathsf{pKt}(x, y) \leq \mathsf{pKt}(x \mid y) + \mathsf{pKt}(y) + O(\log(n))$ by computing y and x in sequence, we get

$$|\mathsf{pKt}(x, y) - \mathsf{pKt}(x \mid y) - \mathsf{pKt}(y)| \leq e(n) + O(\log(n)).$$

Moreover, by applying SoI twice and padding strings if necessary, it is not hard to see that SoI also holds when conditioning on a string z such that $|y| + |z| \leq n$ and $|x| + |y| \leq n$ (see, e.g., the proof of Lemma 8). In other words, we have the following bound:

$$|\mathsf{pKt}(x, y \mid z) - \mathsf{pKt}(x \mid y, z) - \mathsf{pKt}(y \mid z)| \leq 2 \cdot e(n) + O(\log(n)). \tag{4}$$

Let C_1, C_2 be the constants hidden by big O notations in Equation (2) and Equation (4) respectively, and let $C = \max(C_1, C_2)$. Now for a given $x \in \{0,1\}^n$ and $\varepsilon > 0$ satisfying the appropriate constraint, we set $k = \lfloor \frac{\varepsilon n}{2 \cdot e(n) + 2C \log n} \rfloor$. Note that the value k is well defined due to the assumed lower bound on ε. We split x into k pieces $x = x_1 \circ x_2 \circ \cdots \circ x_k$, where each x_i has length $\lceil \frac{n}{k} \rceil$ except for x_k, which has length $n - \lceil \frac{n}{k} \rceil \cdot (k-1)$. Now by applying Equation (4) k times, we get

$$\left| \mathsf{pKt}(x) - \sum_{i=1}^{k} \mathsf{pKt}(x_i \mid x_{i+1} \circ \cdots \circ x_k) \right| \leq k \cdot (2 \cdot e(n) + C \log n).$$

Then for each i, we use the algorithm A in Lemma 4 to estimate $\mathsf{pKt}(x_i \mid x_{i+1} \circ \cdots \circ x_k)$ in time $2^{O(|x_i|)} \cdot \mathsf{poly}(n) = 2^{O(n/k + \log n)} = 2^{O((e(n) + \log n)/\varepsilon)}$. Let r_i be the estimation for $\mathsf{pKt}(x_i \mid x_{i+1} \circ \cdots \circ x_k)$. Then, by a union bound, with probability at least $1 - k \cdot 2^{-n}$ over the randomness of A, for each i, we get $|r_i - \mathsf{pKt}(x_i \mid x_{i+1} \circ \cdots \circ x_k)| \leq C$. Consequently,

$$\begin{aligned}
\left| \mathsf{pKt}(x) - \sum_{i=1}^{k} r_i \right| &\leq \left| \mathsf{pKt}(x) - \sum_{i=1}^{k} \mathsf{pKt}(x_i \mid x_{i+1} \circ \cdots \circ x_k) \right| + \sum_{i=1}^{k} |r_i - \mathsf{pKt}(x_i \mid x_{i+1} \circ \cdots \circ x_k)| \\
&\leq k \cdot (2 \cdot e(n) + C \log n + C) \\
&\leq \varepsilon n.
\end{aligned}$$

Hence we have an algorithm running in time $k \cdot 2^{O((e(n) + \log n)/\varepsilon)} = 2^{O((e(n) + \log n)/\varepsilon)}$, that with high probability, estimates $\mathsf{pKt}(x)$ within εn additive error.

6 Failure of SoI on Average: Proof of Theorem 4

We start with a proof for rKt, and we note that the proof for pKt is almost identical (see Remark 1). For the sake of contradiction, we assume that *SoI holds on average*, i.e., there exists a constant $C_1 \geq 0$ such that for every polynomial-time samplable distribution family $\mathcal{D} = \{\mathcal{D}_n\}_{n\geq 1}$, where each $\mathcal{D}_n$ is supported over $\{0,1\}^n \times \{0,1\}^n$, and every polynomial $p(n)$, it holds for all large enough n that

$$\Pr_{(x,y)\sim\mathcal{D}_n}[\mathsf{rKt}(y \mid x) + \mathsf{rKt}(x) \leq \mathsf{rKt}(y,x) + C_1 \log n] \geq 1 - \frac{1}{p(n)}. \tag{5}$$

We consider the promise problem Gap-MrKtP$[0.1n, 0.9n]$, where the "yes" instances are given by $\{x \mid \mathsf{rKt}(x) \geq 0.9|x|\}$, and the "no" instances are given by $\{x \mid \mathsf{rKt}(x) \leq 0.1|x|\}$. It will also be useful to view this promise problem as the relation consisting of all valid solutions. We say that $(x, \text{Yes}) \in$ Gap-MrKtP$[0.1n, 0.9n]$ if $\mathsf{rKt}(x) \geq 0.9|x|$, and $(x, \text{No}) \in$ Gap-MrKtP$[0.1n, 0.9n]$ if $\mathsf{rKt}(x) \leq 0.1|x|$. We start by showing that the validity of SoI on average implies that Gap-MrKtP can be efficiently solved on average (i.e., with respect to polynomial-time samplable distributions).

We now describe a randomized algorithm $\mathsf{Esti}(x, 1^n)$, which will be useful for our proof. Let C_1 be the constant from Equation (5), C_2 be the constant hidden by $O(1)$ in Lemma 3, and C_3 be the constant in Lemma 5. Let $k \triangleq \left\lfloor \frac{|x|}{3C_2+3(C_1+3C_3)\log n} \right\rfloor$. For an input string x, let $x_1, \ldots, x_k$ be its decomposition into k parts of almost equal size. Then, for each $i \in [k]$, the algorithm estimates $\mathsf{rKt}(x_i \mid x_{i-1} \circ \cdots \circ x_1)$ using the algorithm A from Lemma 3. We denote the estimate by e_i. Then the algorithm outputs Yes if $\sum_i e_i \geq 0.5n$, and No otherwise. It is not hard to see that $\mathsf{Esti}(x, 1^n)$ runs in time polynomial in n and $|x|$.

Lemma 9. *If SoI for* rKt *holds on average, then for every polynomial $p(n), q(n) \geq n$ and any polynomial-time samplable distribution $\mathcal{D} = \{\mathcal{D}_n\}_{n\geq 1}$ where each $\mathcal{D}_n$ is supported over $\{0,1\}^{\leq q(n)}$, it holds for every sufficiently large $n \in \mathbb{N}$ that*

$$\Pr_{x\sim\mathcal{D}_n, \mathsf{Esti}}\left[(x, \mathsf{Esti}(x, 1^{q(n)})) \in \text{Gap-MrKtP}[0.1n, 0.9n]\right] \geq 1 - \frac{1}{p(n)}.$$

Proof. For each $x \in \mathcal{D}_n$, let $x_1, \ldots, x_k$ be the division of x in $\mathsf{Esti}(x, 1^{q(n)})$. Let D be the sampler such that $\mathsf{D}(1^n)$ samples $\mathcal{D}_n$ in polynomial time. We note that the following two conditions are sufficient for $(x, \mathsf{Esti}(x, 1^n)) \in$ Gap-MrKtP$[0.1n, 0.9n]$:

1. For all $i \in [k]$, A estimates $\mathsf{rKt}(x_i \mid x_{i-1} \circ \cdots \circ x_1)$ with at most C_2 additive error.
2. For all $i \in [k]$, $|\mathsf{rKt}(x_i \circ \cdots \circ x_1) - \mathsf{rKt}(x_i \mid x_{i-1} \circ \cdots \circ x_1) - \mathsf{rKt}(x_{i-1} \circ \cdots \circ x_1)| \leq (C_1 + 3C_3) \cdot \log q(n)$.

This is because when both conditions hold, by telescoping we get

$$\left|\sum_{i=1}^{k} e_i - \mathsf{rKt}(x)\right| \leq ((C_1 + 3C_3) \cdot \log q(n) + C_2) \cdot k \leq \frac{|x|}{3}$$

for all sufficiently large n. Therefore, if $\mathsf{rKt}(x) \geq 0.9|x|$ then $\sum e_i \geq (0.9 - 0.34)|x| \geq 0.5|x|$ and $\mathsf{Esti}(x, 1^{q(n)})$ outputs Yes, and similarly when $\mathsf{rKt}(x) \leq 0.1|x|$, $\mathsf{Esti}(x, 1^{q(n)})$ outputs No. Next, we estimate the probability of one of the conditions failing.

By Lemma 3 and a union bound over $i \in [k]$, Item 1 fails with probability at most $t(n) \cdot 2^{-\Omega(\log^2 n)}$, which is less than $\frac{1}{2p(n)}$ for all sufficiently large n. We claim that the probability of Item 2 failing is upper

bounded by $\frac{1}{2p(n)}$ for all sufficiently large n. Otherwise, we define a sampler D' which on input 1^n, runs in polynomial time, and samples a pair of strings from $\{0,1\}^n \times \{0,1\}^n$: $\mathsf{D}'(1^n)$ first runs $\mathsf{D}(1^{q^{-1}(n)})$ to obtain x, then samples $i \sim [k]$, and finally outputs $(x_i \circ 1^{n-|x_i|}, x_{i-1} \circ \cdots \circ x_1 \circ 1^{n-|x_{i-1}|-\cdots-|x_1|})$. By Lemma 5 and our assumption that Item 2 fails with probability $\frac{1}{2p(n)}$ for infinitely many n, we get

$$\Pr_{(x,y)\leftarrow \mathsf{D}'(1^n)}[|\mathsf{rKt}(x \circ y) - \mathsf{rKt}(x \mid y) - \mathsf{rKt}(y)| \geq C_1 \log n] \geq \frac{1}{2p(q^{-1}(n)) \cdot n} \geq \frac{1}{2n \cdot p(n)}$$

for infinitely many n, which contradicts our assumption that SoI holds on average for rKt. Hence by a union bound, for all sufficiently large n, the probability that both Item 1 and Item 2 are satisfied is at least $1 - 1/p(n)$. □

Let $t(n)$ be the polynomial upper bound on the number of random bits used by $\mathsf{Esti}(\cdot, 1^n)$. We would like to define a randomized oracle $\mathcal{G}_n : \{0,1\}^{\leq n} \to \{\text{Yes}, \text{No}\}$ that is correlated with $\mathsf{Esti}(\cdot, 1^n)$ and agrees with $\mathsf{Gap\text{-}MrKtP}[0.1n, 0.9n]$. To do this, for any function $F : \{0,1\}^{\leq n} \to \{0,1\}^{t(n)}$, we denote by $\mathsf{Esti}^F(\cdot, 1^n)$ the algorithm that uses $F(x)$ as the (internal) randomness for $\mathsf{Esti}(x, 1^n)$. We then define the oracle $\mathcal{G}_n^F$ as follows:

$$\mathcal{G}_n^F(x) = \begin{cases} \text{Yes}, & \mathsf{rKt}(x) \geq 0.9|x| \\ \text{No}, & \mathsf{rKt}(x) \leq 0.1|x| \\ \mathsf{Esti}^F(x, 1^n), & 0.1|x| < \mathsf{rKt}(x) < 0.9|x| \end{cases}$$

The next lemma shows that $\mathcal{G}_n^F(\cdot)$ and $\mathsf{Esti}^F(\cdot, 1^n)$ cannot be distinguished with significant advantage when accessed as oracles:

Lemma 10. *Assume that SoI for rKt holds on average. Let $s(n)$ be any polynomial, and let $\mathsf{R}^{(\cdot)} : \{0,1\}^n \times \{0,1\}^{s(n)} \to \{0,1\}$ be any deterministic polynomial-time algorithm with oracle access running in time $s(n)$. Let $p(n)$ be any polynomial, and let $\mathcal{D} = \{\mathcal{D}_n\}_{n\geq 1}$ be any polynomial-time samplable distribution where each $\mathcal{D}_n$ is supported over $\{0,1\}^n$. Then for every large enough n, we have*

$$\Pr_{\substack{x\sim\mathcal{D}_n, r\sim\mathcal{U}_{s(n)}, \\ F\sim\{0,1\}^{\leq r(n)}\to\{0,1\}^{t(s(n))}}} \left[\mathsf{R}^{\mathsf{Esti}^F(\cdot,1^{s(n)})}(x,r) = \mathsf{R}^{\mathcal{G}_{s(n)}^F(\cdot)}(x,r)\right] \geq 1 - \frac{1}{p(n)}.$$

Proof. We define a polynomial-time sampler D' that works as follows.

> On input 1^n, D' samples $x \sim \mathcal{D}_n$, $r \sim \mathcal{U}_{s(n)}$, and $i \sim [s(n)]$. Then it runs $\mathsf{R}^{(\cdot)}(x,r)$. For the first $(i-1)$ oracle queries $y_1, \ldots, y_{i-1}$ by $\mathsf{R}^{(\cdot)}(x,r)$, D' returns $\mathsf{Esti}(y_1, 1^{s(n)}), \ldots, \mathsf{Esti}(y_{i-1}, 1^{s(n)})$ respectively. When $\mathsf{R}^{(\cdot)}(x,r)$ makes the i-th query y_i, D' outputs y_i and halts. If $\mathsf{R}^{(\cdot)}(x,r)$ halts before making the i-th query, then D' outputs $1^{s(n)}$.

It is not hard to see that $\mathsf{D}'(1^n)$ runs in time $\mathsf{poly}(n)$, and outputs a string of length at most $s(n)$. By Lemma 9, we have

$$\Pr_{y\leftarrow\mathsf{D}'(1^n),\mathsf{Esti}}\left[(y, \mathsf{Esti}(y, 1^{s(n)}) \notin \mathsf{Gap\text{-}MrKtP}[0.1n, 0.9n])\right] \leq \frac{1}{s(n) \cdot p(n)}.$$

By the definition of $\mathcal{G}_n^F$, we get

$$\Pr_{y\leftarrow\mathsf{D}'(1^n),F\sim\{0,1\}^{\leq s(n)}\to\{0,1\}^{t(s(n))}}\left[\mathsf{Esti}^F(y, 1^{s(n)}) \neq \mathcal{G}_{s(n)}^F(y)\right] \leq \frac{1}{s(n) \cdot p(n)}. \tag{6}$$

Now over the random choice of x, r, F, for $i \in [s(n)]$, let E_i denote the event that, the first $(i-1)$ oracle queries of $\mathsf{R}^{\mathsf{Esti}^F(\cdot,1^{s(n)})}(x,r)$ and $\mathsf{R}^{\mathcal{G}^F_{s(n)}(\cdot)}(x,r)$ returned the same answer, while the i-th oracle query returned different answers. Note that we can think of $\mathsf{D}'(1^n)$ as first sampling x, r, F (F is sampled implicitly when sampling the randomness for Esti), and then sampling $i \sim [s(n)]$, and finally outputting y_i. By the definition of E_i, when D' samples i and E_i holds, it must hold that $\mathsf{Esti}^F(y_i, 1^{s(n)}) \neq \mathcal{G}^F_{s(n)}(y_i)$. Hence we have

$$\Pr_{y\leftarrow\mathsf{D}'(1^n),F\sim\{0,1\}^{\leq s(n)}\to\{0,1\}^{t(s(n))}}\left[\mathsf{Esti}^F(y,1^{s(n)}) \neq \mathcal{G}^F_{s(n)}(y)\right] \geq \frac{1}{s(n)}\cdot\sum_{i=1}^{s(n)}\Pr[E_i]. \tag{7}$$

Combining Equations (6) and (7), we get

$$\sum_{i=1}^{s(n)}\Pr[E_i] \leq \frac{1}{p(n)}.$$

Since R is deterministic, $\mathsf{R}^{\mathsf{Esti}^F(\cdot,1^{s(n)})}(x,r)$ and $\mathsf{R}^{\mathcal{G}^F_{s(n)}(\cdot)}(x,r)$ outputting different answers implies that at least one of the oracle queries returned different answers. Hence by a union bound, we get

$$\Pr_{\substack{x\sim\mathcal{D}_n, r\sim\mathcal{U}_{s(n)},\\ F\sim\{0,1\}^{\leq r(n)}\to\{0,1\}^{t(s(n))}}}\left[\mathsf{R}^{\mathsf{Esti}^F(\cdot,1^{s(n)})}(x,r) \neq \mathsf{R}^{\mathcal{G}^F_{s(n)}(\cdot)}(x,r)\right] \leq \sum_{i=1}^{s(n)}\Pr[E_i] \leq \frac{1}{p(n)},$$

which completes the proof. □

Lemma 11 ([Oli19])**.** *Let L_{TV} be the* PSPACE*-complete language in Lemma 7. Then there exists a probabilistic polynomial-time algorithm* R*, such that for any randomized oracle $\mathcal{O}$ satisfying*

$$\Pr[\mathcal{O}(x)=1] \geq \frac{2}{3}, \quad \forall x \text{ s.t. } \mathsf{rKt}(x) \geq 0.9|x|,$$
$$\Pr[\mathcal{O}(x)=1] \leq \frac{1}{3}, \quad \forall x \text{ s.t. } \mathsf{rKt}(x) \leq 0.1|x|,$$

$\mathsf{R}^{\mathcal{O}}$ *decides L_{TV} with error probability at most 2^{-n}.*

Lemma 12. *If SoI for* rKt *holds on average, then* $\mathsf{PSPACE} \subseteq \mathsf{BPP}$.

Proof. Assume that L_{TV} from Lemma 7 is $\frac{1}{p(n)}$-random-self-reducible for a polynomial $p(n)$, and let $s(n)$ be the polynomial bound on the running time of R in Lemma 11. We view R as a deterministic algorithm taking two inputs, the first being the original input $x \in \{0,1\}^n$, and the second being the randomness $r \in \{0,1\}^{s(n)}$. By the definition of $\mathcal{G}^F_{s(n)}$, we get

$$\Pr_{\substack{x\sim\mathcal{U}_n, r\sim\mathcal{U}_{s(n)},\\ F\sim\{0,1\}^{\leq s(n)}\to\{0,1\}^{t(s(n))}}}\left[\mathsf{R}^{\mathcal{G}^F_{s(n)}(\cdot)}(x,r) \neq L_{\mathsf{TV}}(x)\right] \leq 2^{-n}. \tag{8}$$

By Lemma 10, we get

$$\Pr_{\substack{x\sim\mathcal{U}_n, r\sim\mathcal{U}_{s(n)},\\ F\sim\{0,1\}^{\leq s(n)}\to\{0,1\}^{t(s(n))}}}\left[\mathsf{R}^{\mathsf{Esti}^F(\cdot,1^{s(n)})}(x,r) \neq \mathsf{R}^{\mathcal{G}^F_{s(n)}(\cdot)}(x,r)\right] \leq \frac{1}{7p(n)}. \tag{9}$$

Applying a union bound over Equations (8) and (9), we have

$$\Pr_{\substack{x\sim\mathcal{U}_n, r\sim\mathcal{U}_{s(n)},\\ F\sim\{0,1\}^{\le s(n)}\to\{0,1\}^{t(s(n))}}}\left[\mathsf{R}^{\mathsf{Esti}^F(\cdot,1^{s(n)})}(x,r)\neq L_{\mathsf{TV}}(x)\right]\le\frac{1}{7p(n)}+2^{-n}\le\frac{1}{6p(n)}.$$

By Markov's inequality, we get

$$\Pr_{r\sim\mathcal{U}_{s(n)},\mathsf{Esti}}\left[\Pr_{x\sim\mathcal{U}_n}\left[\mathsf{R}^{\mathsf{Esti}(\cdot,1^{s(n)})}(x,r)\neq L_{\mathsf{TV}}(x)\right]\ge\frac{1}{p(n)}\right]\le\frac{1}{6}.$$

In other words, with probability at least $5/6$ over r and the internal randomness of Esti, $\mathsf{R}^{\mathsf{Esti}(\cdot,1^{s(n)})}(\cdot,r)$ decides L_{TV} correctly on a $1-1/p(n)$ fraction of inputs. Since L_{TV} is $\frac{1}{p(n)}$-random-self-reducible, we can use the corrector Cor to compute $L_{\mathsf{TV}}(x)$ for any $x\in\{0,1\}^n$ with error probability at most $1/6$. Hence we get

$$\forall x\in\{0,1\}^n,\Pr_{r\sim\mathcal{U}_{s(n)},\mathsf{Esti},\mathsf{Cor}}\left[\mathsf{Cor}^{\mathsf{R}^{\mathsf{Esti}(\cdot,1^{s(n)})}(\cdot,r)}(x)=L_{\mathsf{TV}}(x)\right]\ge\frac{5}{6}\cdot\frac{5}{6}>\frac{2}{3}.$$

That is, we can decide L_{TV} in probabilistic polynomial time. □

Now we are ready to reach contradiction. Note that for strings x,y of length at most n, deciding exactly whether $\mathsf{rKt}(x\mid y)<2c\log n$ is in PSPACE. Therefore, there exists an efficient deterministic algorithm N such that, given oracle access to a PSPACE complete language, it outputs $x_1,\dots,x_k\in\{0,1\}^{2c\log n}$ for $k=\left\lfloor\frac{n}{2c\log n}\right\rfloor$ such that for every $i<r$, we have

$$\mathsf{rKt}(x_i\mid x_{i+1}\circ\cdots\circ x_k)\ge 2c\log n.$$

This can be done using a brute force iteration over $x_k,\dots,x_1$ by computing randomized Kolmogorov complexity using the PSPACE oracle. By replacing each PSPACE oracle with calls to the BPP algorithm provided by Lemma 12 (and amplifying its success probability), we get that $x_1\circ\cdots\circ x_k$ can be computed in polynomial time, and therefore $\mathsf{rKt}(x_1\circ\cdots\circ x_k)\le O(\log n)$. On the other hand, given 1^n and uniformly random bits, we can efficiently generate the string pairs

$$(x_i, x_{i+1}\circ\cdots\circ x_k).$$

Therefore, we can apply SoI on this polynomial-time samplable distribution with error $\frac{1}{2n}$ and thus all SoI inequalities must hold. As a result, we get that

$$\begin{aligned}\mathsf{rKt}(x_1\circ\cdots\circ x_k)&\ge\sum\mathsf{rKt}(x_i\mid x_{i+1}\circ\cdots\circ x_k)-k\cdot c\log n\\&\ge k\cdot 2c\log n-k\cdot c\log n\\&=k\cdot c\log n.\end{aligned}$$

For sufficiently large n, due to our choice of k, the bound $k\cdot c\log n$ is larger than $\mathsf{rKt}(x_1\circ\cdots\circ x_k)\le O(\log n)$ and thus we get a contradiction.

Remark 1. The above proof also works with pKt. We first note that Lemma 10 does not depend at all on properties of rKt and would have essentially the same proof using $\mathsf{Gap\text{-}MpKtP}$ instead of $\mathsf{Gap\text{-}MrKtP}$. Lemma 12 only needs the fact that $\mathsf{Gap\text{-}MrKtP}$ can be used to break the appropriate PRG. It is easy to see that the same holds for pKt. Besides that, in Lemma 9 and at the end of the proof we rely on the following facts:

- Exactly computing $\mathsf{rKt}(x, y)$ for x of logarithmic and y of polynomial length is in PSPACE.
- Estimating $\mathsf{rKt}(x, y)$ up to additive error for x of logarithmic and y of polynomial length can be done in polynomial time.
- For every x that can be constructed by an efficient (randomized) algorithm it holds that $\mathsf{rKt}(x) \leq O(\log n)$.

It is easy to see that all properties hold for pKt as well.

7 Stronger Bounds for Conditional SoI: Proofs of Theorem 5 and Theorem 6

Before diving into the formal proof of Theorem 5, we will discuss the intuition behind the argument and also give a sketch of the proof.

Technical Overview. Assume that conditional SoI holds for rKt with error $e(n) = \varepsilon \cdot n$, where $\varepsilon > 0$ is a small constant to be determined later. Using an idea similar to the proof of Theorem 3, we can construct an algorithm A such that given $x \in \{0,1\}^n$ and $y \in \{0,1\}^m$, $A(x, y)$ decides the promise problem of whether $\mathsf{rKt}(x \mid y) \leq 0.1n$ or $\mathsf{rKt}(x \mid y) \geq 0.9n$ with high probability in time $2^{O(\varepsilon \cdot n)} \cdot m^{O(1)}$ (see Lemma 14). This does not contradict [Oli19], which only gives a lower bound on the running time of the form $n^{\mathsf{poly}(\log n)}$. Hence we will look more carefully into the proof of [Oli19].

In [Oli19], they used the PRG $G \colon \{0,1\}^n \to \{0,1\}^{n^c}$ from [IW01], initialized with the PSPACE-complete language L_{TV} of [TV07]. Here $G(\cdot)$ is computable in polynomial time with oracle access to L_{TV} on length at most n, which is computable in space n^d and thus time 2^{n^d}. We choose the parameters such that $c > d$. Hence $\mathsf{rKt}(G(z)) \leq n + n^d + O(\log n)$ for any $z \in \{0,1\}^n$, while from a counting argument, $\mathsf{rKt}(r) \geq n^c - O(1)$ for most of the strings $r \in \{0,1\}^{n^c}$. Therefore any algorithm deciding $\mathsf{Gap\text{-}MrKtP}[0.1n, 0.9n]$ is a distinguisher for G. Then, from the uniform polynomial-time reconstruction procedure of G, if $\mathsf{Gap\text{-}MrKtP}[0.1n, 0.9n] \in \mathsf{prBPTIME}[T(n)]$, we would get $L_{\mathsf{TV}} \in \mathsf{BPTIME}[T(O(n^c)) \cdot \mathsf{poly}(n)]$. Modulo polynomial overheads in parameters, this would eventually contradict $\mathsf{Gap\text{-}MrKtP}[0.1n, 0.9n] \in \mathsf{prBPTIME}[T(n)]$ when $T(n)$ is less than a half-exponential function, i.e., $T(T(n)) < 2^n$. But in our setting, $T(n) = 2^{\varepsilon \cdot n}$ is too large for this proof to work.

The key insight is that, once we get $L_{\mathsf{TV}} \in \mathsf{BPTIME}[2^{O(\varepsilon \cdot n^c)}]$, we can compute G in probabilistic time $2^{O(\varepsilon \cdot n^c)}$. This gives $\mathsf{rKt}(G(z) \mid z) \leq O(\varepsilon \cdot n^c)$. Hence to distinguish $(G(z), z)$ from (r, z) where r, z are sampled uniformly from $\{0,1\}^{n^c}$ and $\{0,1\}^n$, respectively[10], it suffices to run $A(u', z)$, where u' is the prefix of the first string of length $O(\varepsilon \cdot n^c)$. But this only takes time $2^{O(\varepsilon^2 \cdot n^c)}$, and the reconstruction procedure gives $L_{\mathsf{TV}} \in \mathsf{BPTIME}[2^{O(\varepsilon^2 \cdot n^c)}]$. When the constant ε is chosen to be small enough, this is faster than the $\mathsf{BPTIME}[2^{O(\varepsilon \cdot n^c)}]$ upper bound we obtained earlier!

Using the insight from above, we can iteratively apply the following "bootstrapping" argument:

1. Assume $L_{\mathsf{TV}} \in \mathsf{BPTIME}[2^{s(n)}]$.
2. Since G is polynomial-time computable with oracle queries to L_{TV} at length at most n, $\mathsf{rKt}(G(z) \mid z) \lesssim s(n)$.

[10] Different from the usual definition, here the distinguisher also gets the seed as part of its input. Nevertheless, we show that only a minor modification is required for the reconstruction procedure to work for this "weaker" distinguisher; see Lemma 16.

3. When the distinguisher gets (u, a), it only needs to invoke A on a and the length-$O(s(n))$ prefix of u, which runs in time $2^{O(\varepsilon \cdot s(n))}$.

4. Using the above distinguisher, the reconstruction procedure of G gives $L_{\mathsf{TV}} \in \mathsf{BPTIME}[2^{O(\varepsilon \cdot s(n))}]$.

Roughly speaking, if we start from $L_{\mathsf{TV}} \in \mathsf{BPTIME}[2^{n^c}]$ and apply the above argument i times, we would get $L_{\mathsf{TV}} \in \mathsf{BPTIME}[2^{(c'\varepsilon)^i \cdot n^c}]$ for some constant c'. Thus in principle, if ε is small enough and we set $i = O(\log n)$, we get $L_{\mathsf{TV}} \in \mathsf{BPP}$ and thus $\mathsf{PSPACE} \subseteq \mathsf{BPP}$!

By successively designing algorithms $P_1, \ldots, P_i$, where each P_i is faster than P_{i-1}, it is not hard to show that the iterative argument works for a fixed *constant* i. However, to obtain $L_{\mathsf{TV}} \in \mathsf{BPP}$, the number of times we bootstrap *depends* on the input length n. For this reason, in the proof of Lemma 17 given below, we will describe a *single* algorithm P that works as described above (i.e., using the reconstruction procedure for a distinguisher D that truncates the first input string to a certain length), such that $P(i, n)$ runs in time $2^{2^{-i} \cdot n^{O(1)}}$, and outputs a circuit deciding L_{TV} on length n. Due to the need for a value of i that depends on n, we must be particularly careful with the constants hidden by big $O(\cdot)$ notations, which might blow up after super-constantly many rounds, and with corner cases, like when i is too large, or when the first string that D gets is shorter than the length of the prefix we need to maintain to guarantee correctness of the distinguisher.

Once we obtain $\mathsf{PSPACE} \subseteq \mathsf{BPP}$, we can contradict the conditional SoI of rKt by explicitly constructing a sequence of strings whose concatenation should have large rKt complexity, in a short time.

We now provide the formal proof. We will rely on the relation between uniform computations and Boolean circuits. The following standard statement is sufficient for our purposes.

Lemma 13. *There exists a constant C_{circ} such that for every input length $n \geq 2$, the following holds. Let P be a program that runs in time $t(n)$. There exists a circuit C of size at most $(n + t(n))^{C_{circ}}$ that agrees with P over inputs of length n. Moreover, this circuit can be produced in time $(n + t(n))^{C_{circ}}$ given P and n. On the other hand, any circuit C of size k and of input size n can be simulated on a given input in time $(n + k)^{C_{circ}}$.*

First, we derive the following consequence from the validity of conditional SoI.

Lemma 14. *Suppose that conditional symmetry of information holds for rKt with error $\log n \leq e(n) \leq n/40$. Then there exists a constant C and an algorithm $A(x, y)$ that gets two input strings $x \in \{0,1\}^n$ and $y \in \{0,1\}^m$ (we may also write $A(x \mid y)$ to match the notation for conditional Kolmogorov complexity), where $n \leq m$, runs in time $2^{C \cdot e(n)} \cdot m^C$, and with probability at least $1 - 2^{-100(n+m)}$ over its internal randomness, $A(x, y)$ returns 1 if $\mathsf{rKt}(x \mid y) \geq 0.9|x|$ and 0 if $\mathsf{rKt}(x \mid y) \leq 0.1|x|$.*

Proof. By padding (see Lemma 5), from conditional symmetry of information for rKt, we get that for every $x, y, w \in \{0,1\}^*$ where $|x|, |y| \leq n$, it holds that $|\mathsf{rKt}(y \mid x, w) + \mathsf{rKt}(x \mid w) - \mathsf{rKt}(x, y \mid w)| \leq C_{\text{SoI}} \cdot e(n)$ for some constant C_{SoI}. Suppose the algorithm in Lemma 3 runs in time $2^{C_t \cdot n} \cdot m^{C_t}$ and makes at most C_t additive error when estimating conditional rKt, where C_t is a universal constant. The algorithm A splits x into $x = x_1 \circ x_2 \circ \cdots \circ x_\ell$, where $x_1, \ldots, x_{\ell-1}$ are of length $\lceil 40(C_{\text{SoI}} + C_t)e(n) \rceil$, and x_ℓ has length $\leq \lceil 40(C_{\text{SoI}} + C_t)e(n) \rceil$. Then, for each $i \in [\ell]$, let e_i be the estimation of $\mathsf{rKt}(x_i \mid x_1 \circ \cdots \circ x_{i-1} \circ y)$ by the algorithm from Lemma 3. Finally, A returns 1 if $\sum_{i=1}^{\ell} e_i \geq \frac{n}{2}$, and 0 otherwise.

To analyze the correctness of this algorithm, first note that applying ℓ times the conditional symmetry of

information for rKt, we get

$$\left|\mathsf{rKt}(x \mid y) - \sum_{i=1}^{\ell} \mathsf{rKt}(x_i \mid x_1 \circ \cdots \circ x_{i-1} \circ y)\right| \leq \ell \cdot e(n) \leq \frac{n}{20}.$$

Additionally, if all estimates e_i are correct within C_t additive error of $\mathsf{rKt}(x_i \mid x_1 \circ \cdots \circ x_{i-1} \circ y)$, which happens with high probability by standard repetition, we obtain that $\left|\mathsf{rKt}(x \mid y) - \sum_{i=1}^{\ell} e_i\right|$ is at most

$$\begin{aligned}
&\leq \left|\mathsf{rKt}(x \mid y) - \sum_{i=1}^{\ell} \mathsf{rKt}(x_i \mid x_1 \circ \cdots \circ x_{i-1} \circ y)\right| + \sum_{i=1}^{\ell} \left|\mathsf{rKt}(x_i \mid x_1 \circ \cdots \circ x_{i-1} \circ y) - e_i\right| \\
&\leq \frac{n}{20} + \ell \cdot C_t \\
&\leq \frac{n}{10}.
\end{aligned}$$

This establishes the correctness of the algorithm. Finally, the total running time is $2^{40 \cdot C_t (C_t + C_{\mathrm{SoI}}) \cdot e(n)} \cdot \mathsf{poly}(m)$, since we only need to run the algorithm from Lemma 3 on length $\lceil 40(C_{\mathrm{SoI}} + C_t)e(n) \rceil$ at most n times. □

We recall that L_{TV} denotes the PSPACE-complete language of Lemma 7. The following lemma is the key technical result of this section.

Lemma 15 (Collapse of PSPACE from Conditional SoI)**.** *There exist constants $\varepsilon_0 > 0$ and $c_0 \geq 1$ such that if conditional symmetry of information holds for rKt with error $e(n) \leq \varepsilon_0 \cdot n$, then $L_{\mathsf{TV}} \in \mathsf{BPTIME}[n^{c_0}]$.*

Roughly speaking, we prove Lemma 15 by designing a randomized algorithm P such that, for any $i \in \mathbb{N}$ and $x \in \{0,1\}^n$, $P(i,x)$ runs in time $2^{n \cdot 2^{-i}} \cdot \mathsf{poly}(n)$ and computes $P(i,x) = L_{\mathsf{TV}}(x)$ with probability at least $2/3$. By setting $i = O(\log n)$, we obtain the desired result.

To begin with, we will use the pseudorandom generator with uniform reconstruction from [IW01], but with a slight modification in the reconstruction algorithm. More precisely, we observe that its *reconstruction properties* remain valid when its seed is also provided to the distinguisher.

Lemma 16 (Extension of [IW01])**.** *Let L be a language that is both downward-self-reducible and random-self-reducible. Then for any large enough constant c, there exists a generator G and a randomized oracle algorithm $\mathsf{Rec}^{(\cdot,\cdot)}$ such that*

- *$G : \{0,1\}^{p_{in}(n)} \to \{0,1\}^{p_{out}(n)}$, where $p_{in}(n) \geq n$ and $p_{out}(n) = \Omega(n^c)$ are both polynomials.*
- *There exists a deterministic algorithm $B^{(\cdot)}$ and a polynomial $p_{run}(n)$, such that for any $s \in \{0,1\}^{p_{in}(n)}$, $B^{\mathcal{O}}(s)$ runs in time $p_{run}(n)$, makes queries to $\mathcal{O}$ only on inputs of length n, and outputs a string of length $p_{out}(n)$. Moreover, $B^{L}(s) = G(s)$.*
- *$\mathsf{Rec}^{D,\mathcal{O}}(1^n)$ runs in time $p_{run}(n)$, making queries to D only over input length $p_{in}(n)+p_{out}(n)$ and to $\mathcal{O}$ only over input length $n-1$, and outputs a (deterministic) oracle circuit $C^{(\cdot)} : \{0,1\}^n \to \{0,1\}$ which has oracle gates of input size exactly $p_{in}(n) + p_{out}(n)$.*
- *For any distinguisher $D : \{0,1\}^{p_{in}(n)+p_{out}(n)} \to \{0,1\}$ such that*

$$\Pr_{z \sim \mathcal{U}_{p_{in}(n)}}[D(z \circ G(z)) = 1] - \Pr_{z \sim \mathcal{U}_{p_{in}(n)}, r \sim \mathcal{U}_{p_{out}(n)}}[D(z \circ r) = 1] \geq \frac{1}{n},$$

with probability at least $1-2^{-100n}$, $\mathsf{Rec}^{D,L}(1^n)$ *outputs a circuit* C *as above such that* $C^D(x) = L(x)$ *for every* $x \in \{0,1\}^n$.

Proof Sketch. Our construction of G is the same as [IW01], initialized with the truth table of L over inputs of appropriate length. However, the distinguisher D we are assuming here is weaker than the standard definition, since it also gets the seed for G as part of its input. Nevertheless, we claim that only a slight modification is required for Rec, that is, when applying the hybrid argument and the predictor-to-distinguisher reduction in the first step. This will be explained in more detail in the following paragraphs.

Recall that in [IW01], they define a function $h : \{0,1\}^{p_h(n)} \to \{0,1\}$ which is essentially the hardness amplification of L, and they also used an explicitly constructible combinatorial design $\{S_i\}_{i\in[p_{\text{out}}(n)]}$ such that $S_i \subseteq [p_{\text{in}}(n)]$ and $|S_i| = p_h(n)$. They then defined $G(z) = h(z|_{S_1}) \circ h(z|_{S_2}) \circ \cdots \circ h(z|_{S_{p_{\text{out}}(n)}})$, where $z|_{S_i}$ is the restriction of z to the coordinates specified by S_i. Then for any distinguisher D, using a hybrid argument and the predictor-to-distinguisher reduction, Rec first obtains a D-oracle circuit that predicts $h_{p_h(n)}$ with $1/2 + 1/\mathsf{poly}(n)$ probability over uniform inputs. From this, using hardness amplification, it obtains a D-oracle circuit that predicts f_n with $1-1/\mathsf{poly}(n)$ probability over uniform inputs. Finally, using the random self-reducibility of L_{TV}, it obtains a D-oracle circuit that computes f_n correctly on every input with high probability. We observe that only the first step, i.e., obtaining a D-oracle circuit predicting $h_{p_h(n)}$ with $1/2 + 1/\mathsf{poly}(n)$ probability, requires modification. In fact, the second and third steps are black-box reductions between circuits, and they do not use D directly.

By definition of the distinguisher, we get

$$\Pr_{z\sim\mathcal{U}_{p_{\text{in}}(n)}}\left[D\Big(z \circ h(z|_{S_1}) \circ \cdots \circ h(z|_{S_{p_{\text{out}}(n)}})\Big) = 1\right]$$
$$- \Pr_{z\sim\mathcal{U}_{p_{\text{in}}(n)},r\sim\mathcal{U}_{p_{\text{out}}(n)}}\left[D\big(z \circ r_1 \circ \cdots \circ r_{p_{\text{out}}(n)}\big) = 1\right] \geq \frac{1}{n}.$$

By applying the hybrid argument on the last $p_{\text{out}}(n)$ bits, there exists some $i_0 \in [p_{\text{out}}(n)]$ such that

$$\Pr_{z\sim\mathcal{U}_{p_{\text{in}}(n)},r\sim\mathcal{U}_{p_{\text{out}}(n)}}\left[D\Big(z \circ h(z|_{S_1}) \circ \cdots \circ h(z|_{S_{i_0-1}}) \circ h(z|_{S_{i_0}}) \circ r_{i_0+1} \circ \cdots \circ r_{p_{\text{out}}(n)}\Big) = 1\right]$$
$$- \Pr_{z\sim\mathcal{U}_{p_{\text{in}}(n)},r\sim\mathcal{U}_{p_{\text{out}}(n)}}\left[D\Big(z \circ h(z|_{S_1}) \circ \cdots \circ h(z|_{S_{i_0-1}}) \circ r_{i_0} \circ r_{i_0+1} \circ \cdots \circ r_{p_{\text{out}}(n)}\Big) = 1\right] \geq \frac{1}{n \cdot p_{\text{out}}(n)}.$$

In fact, Rec can find such i_0 with high probability by enumerating over every $i \in [p_{\text{out}}(n)]$, and estimating the probabilities by sampling, while computing L using its oracle access to L at length $n-1$ and downward self reducibility. Next we describe an algorithm Pred for predicting $h(\cdot)$: on input $s \in \{0,1\}^{p_h(n)}$, it sets $z|_{S_{i_0}} = s$, samples $z|_{\overline{S_{i_0}}}$ uniformly from $\{0,1\}^{p_{\text{in}}(n)-p_h(n)}$, samples r uniformly from $\{0,1\}^{p_{\text{out}}(n)}$, and samples b uniformly from $\{0,1\}$. Then Pred outputs b if $D(z \circ h(z|_{S_1}) \circ \cdots \circ h(z|_{S_{i_0-1}}) \circ b \circ r_{i_0+1} \circ \cdots \circ r_{p_{\text{out}}(n)}) = 1$, and $1-b$ otherwise. Then by some calculations, we have

$$\Pr_{\substack{s\sim\mathcal{U}_{p_h(n)},z|_{\overline{S_{i_0}}}\sim\mathcal{U}_{p_{\text{in}}(n)-p_h(n)}\\ r\sim\mathcal{U}_{p_{\text{out}}(n)},b\sim\mathcal{U}_1}}\left[\mathsf{Pred}(s, z|_{\overline{S_{i_0}}}, r, b) = h(s)\right] \geq \frac{1}{2} + \frac{1}{n \cdot p_{\text{out}}(n)}.$$

By Markov's inequality, we get

$$\Pr_{\substack{z|_{\overline{S_{i_0}}}\sim\mathcal{U}_{p_{\text{in}}(n)-p_h(n)}\\ r\sim\mathcal{U}_{p_{\text{out}}(n)},b\sim\mathcal{U}_1}}\left[\Pr_{s\sim\mathcal{U}_{p_h(n)}}\left[\mathsf{Pred}(s, z|_{\overline{S_{i_0}}}, r, b) = h(s)\right] \geq \frac{1}{2} + \frac{1}{2n \cdot p_{\text{out}}(n)}\right] \geq \frac{1}{2n \cdot p_{\text{out}}(n)}.$$

Therefore if Rec samples $z|_{S_{i_0}}$, r and b uniformly for $\mathsf{poly}(n)$ times, with high probability there is one sample such that $\mathsf{Pred}(\cdot, z|_{\overline{S_{i_0}}}, r, b)$ has $\frac{1}{2} + \frac{1}{2n \cdot p_{\mathrm{out}}(n)}$ success probability for estimating $h(\cdot)$, and this can be checked with high probability by using oracle access to L and estimating the success probability of $\mathsf{Pred}(\cdot, z|_{\overline{S_{i_0}}}, r, b)$. Then for the good sample, $\mathsf{Pred}(\cdot, z|_{\overline{S_{i_0}}}, r, b)$ can be transformed into a D-oracle circuit, by saving $z|_{\overline{S_{i_0}}}, r, b$, as well as the truth table for $h(z|S_j)$ of every $j \neq i_0$ when $z|_{\overline{S_{i_0}}}$ is fixed, as advice. The rest of the proof is the same as [IW01]. □

Lemma 17 (Bootstrapping Lemma for L_{TV})**.** *Suppose there exists a constant $\varepsilon_0 > 0$ such that conditional symmetry of information holds for rKt with error $e(n) \leq \varepsilon_0 \cdot n$. Then there exists a constant C_P and a probabilistic algorithm $P(i, n)$ such that, for every i and n, the following conditions hold:*

- *With probability at least $1 - 2^{-10}(1 - 2^{-n})$, $P(i, n)$ outputs a circuit C deciding L_{TV} on length n.*
- *The algorithm $P(i, n)$ runs in time at most $2^{2^{-i} \cdot C_P \cdot n^{C_P}} \cdot (n + i)^{C_P}$.*
- *The output circuit C is of size at most $2^{2^{-i} \cdot C_P \cdot n^{C_P}} \cdot (n + i)^{C_P}$.*

Since L_{TV} is PSPACE-complete, it is easy to see that by plugging in $i = O(\log n)$ in Lemma 17 and running the corresponding circuit, we obtain Lemma 15.

Proof. Before we describe how algorithm P works, we give several definitions. For simplicity, we assume that $n \geq 2$. Let C_{circ} be the constant from Lemma 13. Let C_A be the constant from Lemma 14. We set $\varepsilon_0 = (1600 \cdot C_{\mathrm{circ}}^2 \cdot C_A)^{-1}$. Let C_{TV} be the constant such that L_{TV} from Lemma 7 is computable in time $2^{n^{C_{\mathsf{TV}}}}$ on input length $n \geq 2$. Let $G, p_{\mathrm{in}}, p_{\mathrm{out}}, p_{\mathrm{run}}$ be the generator and polynomials from Lemma 16 initialized with L_{TV} and constant $\tilde{c} = \lceil C_{\mathsf{TV}} \rceil + 1$. Let C_{run} be a constant such that $p_{\mathrm{in}}(n), p_{\mathrm{out}}(n), p_{\mathrm{run}}(n) \leq n^{C_{\mathrm{run}}}$ for $n \geq 2$. As $p_{\mathrm{out}} = \Omega(n^{\tilde{c}})$, there must exist some natural number n_0 such that for every $n \geq n_0$ it holds that $p_{\mathrm{out}}(n) \geq 800 \cdot (C_{\mathrm{circ}}^2 \cdot (n^{C_{\mathsf{TV}}} + \log n))$.

Next, we will define more refined bounds on the running time and output length of the program P. We define $\alpha(i, n) = \max(n^{C_{\mathsf{TV}}} \cdot 2^{-i}, 2) + \log n$. We define a constant $C^\dagger$ that is at least $(1000 C_{\mathrm{circ}}^3 \cdot C_A \cdot C_{\mathrm{run}} + 2C_{\mathrm{circ}} \cdot (n_0^{C_{\mathsf{TV}}} + \log n_0))$. We also require $C^\dagger$ to be larger than the description length of some programs (specifically, P and G, to be described later), which contain the description of $C^\dagger$ itself. However, since the description of $C^\dagger$ only takes $\log C^\dagger + O(1)$ bits, this will be satisfied by a large enough $C^\dagger$. We then define $C' = 4 \cdot C^\dagger \cdot C_{\mathrm{circ}}$. We will inductively prove that $P(i, n)$ actually runs in time at most $2^{C' \cdot \alpha(i,n)}$, and outputs a circuit of size at most $2^{C^\dagger \cdot \alpha(i,n)}$.

Description of $P(i, n)$. We now describe the algorithm $P(i, n)$. When $i = 0$, $n = 2$, or $C^\dagger \alpha(i, n) \geq C_{\mathrm{circ}} \cdot (n^{C_{\mathsf{TV}}} + \log n)$, the algorithm simply prints the circuit corresponding to the $2^{n^{C_{\mathsf{TV}}}}$-time brute-force algorithm for L_{TV}. For $i > \lceil C_{\mathsf{TV}} \cdot \log(n) \rceil$, the algorithm just runs $P(\lceil C_{\mathsf{TV}} \cdot \log(n) \rceil, n)$. For all other cases, we first describe a subroutine $D(i, \cdot, \cdot)$, which is supposed to work as a distinguisher as stated in Lemma 16. $D(i, \cdot, \cdot)$ gets two strings $u_1 \in \{0,1\}^{p_{\mathrm{in}}(n)}$ and $u_2 \in \{0,1\}^{p_{\mathrm{out}}(n)}$ as input, and defines u' as the prefix of u_2 of length $100C'\alpha(i - 1, n)$. (In fact, as we will show later, $|u_2|$ is at least $100C'\alpha(i - 1, n)$ in this case.) Then D just returns $A(u' \mid u_1)$, where A is the algorithm for estimating conditional rKt as defined in Lemma 14. After defining D, we now describe the algorithm $P(i, n)$. It first runs $P(i, n - 1)$ to get some circuit C_{n-1}, and also obtains a circuit C_D by randomly fixing the internal randomness of $D(i, \cdot, \cdot)$. Then it runs $\mathsf{Rec}^{\mathsf{C}_D, \mathsf{C}_{n-1}}(1^n)$ to get some circuit C' with oracle access to C_D, and finally replaces the oracle gate for C_D in C' with C_D.

To finish this proof, we first establish the bound on the running time and output length of P, then prove its correctness.

Bounds on the running time of $P(i,n)$. The bound on the running time is proven by induction on n. For the base case of the induction, when $n = 2$, we get by Lemma 13 that the running time is bounded by

$$(2 + 2^{2^{C_{\mathsf{TV}}}})^{C_{\text{circ}}} \leq 2^{2C_{\text{circ}} \cdot 2^{C_{\mathsf{TV}}}} \leq 2^{2C'} \leq 2^{C'\alpha(i,2)}.$$

For the induction step, we start by examining the case where $0 < i \leq \lceil C_{\mathsf{TV}} \cdot \log(n) \rceil$ and $C^\dagger \cdot \alpha(i,n) < C_{\text{circ}} \cdot (n^{C_{\mathsf{TV}}} + \log n)$. By Lemma 14, $D(i,\cdot,\cdot)$ runs in time

$$2^{C_A \cdot e(100 \cdot C' \cdot \alpha(i-1,n))} \cdot p_{\text{in}}(n)^{C_A} \leq 2^{100 C_A \cdot C' \cdot \alpha(i-1,n) \cdot \varepsilon_0} \cdot n^{C_A \cdot C_{\text{run}}}.$$

Hence by Lemma 13, C_D has size at most

$$2^{100 C_{\text{circ}} \cdot C_A \cdot C' \cdot \alpha(i-1,n) \cdot \varepsilon_0} \cdot n^{2C_{\text{circ}} \cdot C_A \cdot C_{\text{run}}}.$$

Recall that $P(i,n)$ first runs $P(i,n-1)$ to get the circuit C_{n-1} (of size at most $2^{C^\dagger \cdot \alpha(i,n-1)}$ by our induction hypothesis), and then runs $\mathsf{Rec}^{\mathsf{C}_D, \mathsf{C}_{n-1}}$ in time $p_{\text{run}}(n)$ with at most $p_{\text{run}}(n)$ oracle queries to C_D and C_{n-1}. Hence we can upper bound the running time of $P(i,n)$ by

$$2^{C'\alpha(i,n-1)} + p_{\text{run}}(n) \cdot 2^{C_{\text{circ}} \cdot C^\dagger \cdot \alpha(i,n-1)} \cdot n^{2C_{\text{circ}}} + p_{\text{run}}(n) \cdot 2^{100 C_{\text{circ}}^2 \cdot C_A \cdot C' \cdot \alpha(i-1,n) \cdot \varepsilon_0} \cdot n^{4C_{\text{circ}}^2 \cdot C_A \cdot C_{\text{run}}}.$$

We can upper bound the running time by the following calculation:

$$\begin{aligned}
& p_{\text{run}}(n) \cdot \left(2^{C_{\text{circ}} \cdot C^\dagger \cdot \alpha(i,n-1)} \cdot n^{2C_{\text{circ}}} + 2^{100 C_{\text{circ}}^2 \cdot C_A \cdot C' \cdot \alpha(i-1,n) \cdot \varepsilon_0} \cdot n^{4C_{\text{circ}}^2 \cdot C_A \cdot C_{\text{run}}}\right) + 2^{C'\alpha(i,n-1)} \\
\leq\ & n^{8C_{\text{circ}}^2 \cdot C_A \cdot C_{\text{run}}} \cdot \left(2^{C_{\text{circ}} \cdot C^\dagger \cdot \alpha(i,n-1)} + 2^{100 C_{\text{circ}}^2 \cdot C_A \cdot C' \cdot \alpha(i-1,n) \cdot \varepsilon_0}\right) + 2^{C'\alpha(i,n-1)} \\
\leq\ & n^{8C_{\text{circ}}^2 \cdot C_A \cdot C_{\text{run}}} \cdot \left(2^{200 C_{\text{circ}}^2 \cdot C_A \cdot C' \cdot \alpha(i,n) \cdot \varepsilon_0} + 2^{C_{\text{circ}} \cdot C^\dagger \cdot \alpha(i,n-1)}\right) + 2^{C'\alpha(i,n-1)} \\
\leq\ & 2^{(8C_{\text{circ}}^2 \cdot C_A \cdot C_{\text{run}} + 200 C_{\text{circ}}^2 \cdot C_A \cdot C' \cdot \varepsilon_0 + C_{\text{circ}} \cdot C^\dagger) \cdot \alpha(i,n)} + 2^{C'\alpha(i,n-1)} \\
\leq\ & 2^{(C'/8 + 200 C_{\text{circ}}^2 \cdot C_A \cdot C' \cdot \varepsilon_0 + C'/4) \cdot \alpha(i,n)} + 2^{C'\alpha(i,n-1)} \\
\leq\ & 2^{(\frac{3}{8}C' + C'/8) \cdot \alpha(i,n)} + 2^{C'\alpha(i,n-1)} \\
=\ & 2^{C'/2 \cdot \alpha(i,n)} + 2^{C'\alpha(i,n-1)} \\
=\ & 2^{\frac{C'}{2} \cdot \left(\max\left(n^{C_{\mathsf{TV}}} \cdot 2^{-i}, 2\right) + \log n\right)} + 2^{C'\left(\max\left((n-1)^{C_{\mathsf{TV}}} \cdot 2^{-i}, 2\right) + \log(n-1)\right)} \\
\leq\ & 2^{C' \cdot \max\left(n^{C_{\mathsf{TV}}} \cdot 2^{-i}, 2\right)} \cdot (n^{\frac{C'}{2}} + (n-1)^{C'}) \\
\leq\ & 2^{C' \cdot \max\left(n^{C_{\mathsf{TV}}} \cdot 2^{-i}, 2\right)} \cdot n^{C'} \\
=\ & 2^{C'\alpha(i,n)}.
\end{aligned}$$

For the case where $i > \lceil C_{\mathsf{TV}} \cdot \log n \rceil$, it holds that $n^{C_{\mathsf{TV}}} \cdot 2^{-i} \leq 2$. Therefore, $\alpha(i,n) = \alpha(\lceil C_{\mathsf{TV}} \cdot \log n \rceil, n)$. By the previous case analysis, we know that $P(\lceil C_{\mathsf{TV}} \cdot \log n \rceil, n)$ runs in time at most $2^{C'\alpha(\lceil C_{\mathsf{TV}} \cdot \log n \rceil, n)}$. Hence $P(i,n)$ runs in time at most $2^{C'\alpha(\lceil C_{\mathsf{TV}} \cdot \log n \rceil, n)} = 2^{C'\alpha(i,n)}$.

For $i = 0$, by Lemma 13, the running time of the algorithm is $2^{C_{\text{circ}} \cdot n^{C_{\mathsf{TV}}}} \cdot n^{C_{\text{circ}}}$, which can be upper bounded by

$$2^{C_{\text{circ}} \cdot n^{C_{\mathsf{TV}}}} \cdot n^{C_{\text{circ}}} = 2^{C_{\text{circ}} \cdot (n^{C_{\mathsf{TV}}} + \log n)} \leq 2^{C_{\text{circ}} \cdot \alpha(0,n)} \leq 2^{C' \cdot \alpha(0,n)}.$$

For the case where $C^\dagger \cdot \alpha(i,n) \geq C_{\text{circ}} \cdot (n^{C_{\mathsf{TV}}} + \log n)$, the running time can be upper bounded by

$$2^{C_{\text{circ}} \cdot n^{C_{\mathsf{TV}}}} \cdot n^{C_{\text{circ}}} \leq 2^{C_{\text{circ}} \cdot (n^{C_{\mathsf{TV}}} + \log n)} \leq 2^{C^\dagger \cdot \alpha(i,n)} \leq 2^{C' \cdot \alpha(i,n)},$$

where the second inequality comes from the condition for this case.

Bounds on the size of the output circuit of $P(i,n)$. We now bound the size of the circuit printed by $P(i,n)$. $P(i,n)$ either prints the circuit for the brute-force algorithm for L_{TV}, or outputs the circuit returned by $\mathsf{Rec}^{\mathsf{C}_D, \mathsf{C}_{n-1}}$. In the former case, by Lemma 13, the output length is at most $2^{C_{\text{circ}} \cdot n^{C_{\mathsf{TV}}}} \cdot n^{C_{\text{circ}}}$, which happens when $i = 0, n = 2$ or $C^\dagger \alpha(i,n) \geq C_{\text{circ}} \cdot (n^{C_{\mathsf{TV}}} + \log n)$. When $i = 0$ or $n = 2$, it is not hard to show that $2^{C_{\text{circ}} \cdot n^{C_{\mathsf{TV}}}} \cdot n^{C_{\text{circ}}} \leq 2^{C^\dagger \cdot \alpha(i,n)}$ by calculation. For the case where $C^\dagger \cdot \alpha(i,n) \geq C_{\text{circ}} \cdot (n^{C_{\mathsf{TV}}} + \log n)$, we have that $2^{C_{\text{circ}} \cdot n^{C_{\mathsf{TV}}}} \cdot n^{C_{\text{circ}}} \leq 2^{C^\dagger \cdot \alpha(i,n)}$ holds trivially.

For the case where $P(i,n)$ runs $\mathsf{Rec}^{\mathsf{C}_D, \mathsf{C}_{n-1}}(1^n)$, the circuit returned by $\mathsf{Rec}^{\mathsf{C}_D, \mathsf{C}_{n-1}}(1^n)$ is of size at most $p_{\text{run}}(n)$ with $p_{\text{run}}(n)$ oracle gates for C_D, and then the oracle gates are replaced by the actual circuit C_D, which is of size

$$2^{100 C_{\text{circ}} \cdot C_A \cdot C' \cdot \alpha(i-1,n) \cdot \varepsilon_0} \cdot n^{2 C_{\text{circ}} \cdot C_A \cdot C_{\text{run}}}.$$

Hence the size of the output circuit can be upper bounded by

$$\begin{aligned}
& p_{\text{run}}(n) \cdot 2^{100 C_{\text{circ}} \cdot C_A \cdot C' \cdot \alpha(i-1,n) \cdot \varepsilon_0} \cdot n^{2 C_{\text{circ}} \cdot C_A \cdot C_{\text{run}}} \\
\leq\ & 2^{100 C_{\text{circ}} \cdot C_A \cdot C' \cdot \alpha(i-1,n) \cdot \varepsilon_0} \cdot n^{3 C_{\text{circ}} \cdot C_A \cdot C_{\text{run}}} \\
\leq\ & 2^{200 C_{\text{circ}} \cdot C_A \cdot C' \cdot \alpha(i,n) \cdot \varepsilon_0} \cdot n^{C^\dagger/2} \\
\leq\ & 2^{(200 C_{\text{circ}} \cdot C_A \cdot C' \cdot \varepsilon_0 + C^\dagger/2) \cdot \alpha(i,n)} \\
\leq\ & 2^{(C^\dagger/2 + C^\dagger/2) \cdot \alpha(i,n)} \\
\leq\ & 2^{C^\dagger \cdot \alpha(i,n)}.
\end{aligned}$$

Correctness of $P(i,n)$. We prove the correctness by induction. The proposition we will show is that with probability at least $1 - 2^{-10}(1 - 2^{-n})$, the program $P(i,n)$ outputs a circuit that decides L_{TV} on length n. We will do the induction over i and n simultaneously, i.e., assuming the induction hypothesis holds for both $P(i-1,n)$ and $P(i,n-1)$, then the proposition is also correct for $P(i,n)$. The base cases are when $i = 0$ or $n = 2$, where with probability 1, $P(i,n)$ transforms the brute-force algorithm for L_{TV} into the output circuit.

In the induction step, we assume that for some $i > 0$ and $n > 2$, the proposition holds for both $P(i-1,n)$ and $P(i,n-1)$. We will prove that $P(i,n)$ outputs a correct circuit with probability $1 - 2^{-10}(1 - 2^{-n})$. For the case where $C^\dagger \alpha(i,n) \geq C_{\text{circ}} \cdot (n^{C_{\mathsf{TV}}} + \log n)$, with probability 1, $P(i,n)$ transforms the brute-force algorithm for L_{TV} into the output circuit. For the case where $i > \lceil \log(C_{\mathsf{TV}} \cdot n) \rceil$, the correctness reduces to $P(\lceil \log(C_{\mathsf{TV}} \cdot n) \rceil, n)$. Therefore we will focus on the case where $0 < i \leq \lceil \log(C_{\mathsf{TV}} \cdot n) \rceil$ and $C^\dagger \cdot \alpha(i,n) < C_{\text{circ}} \cdot (n^{C_{\mathsf{TV}}} + \log n)$ in the following paragraphs.

Notice that the conjunction of the following three events is sufficient for $P(i,n)$ to output a circuit deciding L_{TV} on length n:

1. C_D $1/n$-distinguishes (u_1, u_2) from $(u_1, G(u_1))$, where u_1 and u_2 are sampled uniformly from the sets $\{0,1\}^{p_{\text{in}}(n)}$ and $\{0,1\}^{p_{\text{out}}(n)}$, respectively.
2. $P(i,n-1)$ outputs a circuit C_{n-1} that decides L_{TV} on length $n-1$.

3. $\mathsf{Rec}^{\mathsf{C}_D,\mathsf{C}_{n-1}}(1^n)$ returns a circuit with oracle access to C_D that decides L_{TV} on length n.

By our inductive hypothesis, Item 2 happens with probability at least $1 - 2^{-10}(1 - 2^{-n+1})$. By Lemma 16, conditioning on Item 1 happening, Item 3 happens with probability at least $1 - 2^{-100n}$. Hence it remains to lower bound the probability of Item 1.

We start by claiming that $100C'\alpha(i-1,n) \le p_{\text{out}}(n)$, so that u_2 can have a prefix of the desired length. This follows from the following computation. First, we note that

$$C'\alpha(i-1,n) \le 2(C'\alpha(i,n)) = 8C_{\text{circ}} \cdot C^\dagger \cdot \alpha(i,n) \le (8C_{\text{circ}}^2 \cdot (n^{C_{\mathsf{TV}}} + \log n)),$$

where the last inequality follows from our assumption that $C^\dagger \cdot \alpha(i,n) < C_{\text{circ}} \cdot (n^{C_{\mathsf{TV}}} + \log n)$. Hence by definition of n_0, if $n \ge n_0$, then

$$p_{\text{out}}(n) \ge 100 \cdot (8C_{\text{circ}}^2 \cdot (n^{C_{\mathsf{TV}}} + \log n)) \ge 100C'\alpha(i-1,n).$$

On the other hand, for $n < n_0$ we have

$$C^\dagger \cdot \alpha(i,n) \ge C^\dagger \ge C_{\text{circ}} \cdot (n_0^{C_{\mathsf{TV}}} + \log n_0) \ge C_{\text{circ}} \cdot (n^{C_{\mathsf{TV}}} + \log n),$$

which contradicts the condition $C^\dagger \cdot \alpha(i,n) < C_{\text{circ}} \cdot (n^{C_{\mathsf{TV}}} + \log n)$ for this case.

By a standard counting argument, for uniformly random $u_1 \in \{0,1\}^{p_{\text{in}}(n)}$, $u' \in \{0,1\}^{100C'\alpha(i-1,n)}$, with high probability ($\ge 9/10$) it holds that

$$\mathsf{rKt}(u' \mid u_1) \ge 100C'\alpha(i-1,n) - 10 \ge 90C'\alpha(i-1,n). \tag{10}$$

Next we analyze the case where u_1 is sampled uniformly at random, but u' is the prefix of $G(u_1)$. By the induction hypothesis and Lemma 13, with probability at least $1 - 2^{-10} \ge 2/3$ over the internal randomness of $P(i-1,n)$, it outputs a circuit of size $2^{C^\dagger \cdot \alpha(i-1,n)}$ deciding L_{TV} in time $2^{C' \cdot \alpha(i-1,n)}$, which can then be evaluated in time $2^{C_{\text{circ}} \cdot C^\dagger \cdot \alpha(i-1,n)}$. Since $G(\cdot)$ can be computed in time $p_{\text{run}}(n)$ with oracle access to L_{TV}, we have

$$\begin{aligned}
\mathsf{rKt}(u' \mid u_1) &\le \mathsf{rKt}(G(u_1) \mid u_1) + 2\log i + 2\log n + O(1) \\
&\le C' \cdot \alpha(i-1,n) + C_{\text{circ}} \cdot C^\dagger \cdot \alpha(i-1,n) + 2\log p_{\text{run}}(n) + 2\log i + 2\log n + O(1) \\
&\le C' \cdot \alpha(i-1,n) + C_{\text{circ}} \cdot C^\dagger \cdot \alpha(i-1,n) + 6C_{\text{run}} \log n + O(1).
\end{aligned}$$

Here the $O(1)$ depends only on the description of P, G and the length of a small program that lets us cut the full output of G into u', so we may assume it is at most C' by setting $C^\dagger$ large enough from the beginning. As $C' \ge C_{\text{run}}$ and $C' \ge C_{\text{circ}} \cdot C^\dagger$, we get that

$$\mathsf{rKt}(u' \mid u_1) \le C' \cdot \alpha(i-1,n) + C_{\text{circ}} \cdot C^\dagger \cdot \alpha(i-1,n) + 6C_{\text{run}} \log n + O(1) \le 10C'\alpha(i-1,n). \tag{11}$$

Now by Lemma 14 and a union bound, with probability at least $1 - 2^{-99p_{\text{in}}(n)} \ge 1 - 2^{-20n}$ over the internal randomness of A, $A(u' \mid u_1)$ decides the promise problem of whether $\mathsf{rKt}(u' \mid u_1) \le 0.1|u'|$ or $\mathsf{rKt}(u' \mid u_1) \ge 0.9|u'|$ correctly for every $(u', u_1) \in \{0,1\}^{100C'\alpha(i-1,n)} \times \{0,1\}^{p_{\text{in}}(n)}$. Combined with Equations (10) and (11), with probability at least $1 - 2^{-20n}$, C_D will $1/2$-distinguish (u_1, u_2) from $(u_1, G(u_1))$.

Now by a union bound, the probability of Items 1 to 3 happening together is at least

$$\begin{aligned}
1 - (2^{-20n} + 2^{-10}(1 - 2^{-n+1}) + 2^{-100n}) &\ge 1 - (2^{-n-11} + 2^{-10}(1 - 2^{-n+1}) + 2^{-n-11}) \\
&= 1 - 2^{-10}(1 - 2^{-n}).
\end{aligned}$$

Hence with probability at least $1 - 2^{-10}(1 - 2^{-n})$, $P(i,n)$ outputs a circuit deciding L_{TV} on length n, which completes the proof of correctness. □

Using Lemma 15, we are now ready to prove Theorem 5.

Proof of Theorem 5. Let ε_0 and c_0 be the constants of Lemma 15. We consider two cases: either $L_{\mathsf{TV}} \notin \mathsf{BPTIME}[n^{c_0}]$, or $L_{\mathsf{TV}} \in \mathsf{BPTIME}[n^{c_0}]$. In the former case, by Lemma 15, conditional SoI fails for rKt with error $e(n) \leq \varepsilon_0 \cdot n$. In the latter case, note that given a string y and a length n, finding the lexicographically first string x of length n such that $\mathsf{rKt}(x \mid y) \geq 0.9|x|$ can be done by exhaustively testing every program of length at most $O(|x|)$ and every random string of length at most $2^{O(|x|)}$, in space $2^{O(|x|)} \cdot \mathsf{poly}(|y|)$. By our assumption that $L_{\mathsf{TV}} \in \mathsf{BPTIME}[n^{c_0}]$ and the PSPACE-completeness of L_{TV}, there exists some constant c_1 and a probabilistic algorithm running in time $2^{c_1 \cdot |x|} . |y|^{c_1}$ outputting such x with probability at least $1 - 2^{-|x|}$. Now let n be a sufficiently large integer. We set $k = 10c_1$, and define k strings $x_1, \ldots, x_k$ each of length n, such that for every i, x_i is the lexicographically first string in $\{0,1\}^n$ satisfying $\mathsf{rKt}(x_i \mid x_1 \circ \cdots \circ x_{i-1}) \geq 0.9n$. Then by using the probabilistic algorithm described above, given n as input, $x_1, \ldots, x_k$ can be found sequentially in time $2^{c_1 \cdot n} \cdot (kn)^{c_1} \cdot k$ with probability at least $2/3$. This gives us

$$\mathsf{rKt}(x_1 \circ \cdots \circ x_k) \leq c_1 \cdot n + O(\log n). \tag{12}$$

On the other hand, assuming conditional SoI for rKt with error $e(n)$, by applying SoI $(k-1)$ times we get

$$\mathsf{rKt}(x_1 \circ \cdots \circ x_k) \geq 0.9kn - k \cdot e(kn) = 9c_1 \cdot n - 10c_1 \cdot e(10c_1 \cdot n). \tag{13}$$

But this contradicts Equation (12) for $e(n) \leq n/(100c_1)$ and large enough n. In conclusion, if we set $\varepsilon \leq \min(\varepsilon_0, (100c_1)^{-1})$, then the conditional SoI for rKt does not hold with error $e(n) \leq \varepsilon \cdot n$. □

Next, we adapt the argument above and establish a corresponding lower bound on the complexity of estimating conditional rKt complexity (Theorem 6).

Proof of Theorem 6. In order to derive a contradiction, assume there is a probabilistic algorithm A with the properties in the statement. First, we observe that the proof of Lemma 15 still holds assuming the existence of A, after an appropriate modification of the parameters. Indeed, the crucial point is that conditional SoI is only employed in the proof to obtain an algorithm that distinguishes strings of bounded conditional rKt complexity from random, and the parameters of A are sufficient for this. Consequently, there exists some constant $c \geq 1$ and $\varepsilon_0 > 0$, such that for any $\varepsilon < \varepsilon_0$, if $A(x, w)$ runs in time $2^{\varepsilon \cdot |x|} \cdot \mathsf{poly}(|w|)$, then $L_{\mathsf{TV}} \in \mathsf{BPTIME}[n^c]$.

Now consider the following space-bounded deterministic procedure $B(1^n)$. It goes over all strings s of length n and exactly computes the acceptance probability of $A(s, \epsilon)$, where ϵ is the empty string, and returns the lexicographically first string x that is accepted with probability at least $2/3$. It is easy to see that $B(1^n)$ runs in space $2^{O(\varepsilon \cdot n)}$ for some constant C_B. Then, from $L_{\mathsf{TV}} \in \mathsf{BPTIME}[n^c]$ and the PSPACE-completeness of L_{TV}, it follows that x can also be constructed with high probability in time $2^{c' \cdot \varepsilon \cdot n}$ for some constant c'. The latter implies that $\mathsf{rKt}(x) \leq c' \cdot \varepsilon \cdot n + O(\log n)$.

By setting $\varepsilon \leq \min(\varepsilon_0, \alpha/(2c'))$, we have $\mathsf{rKt}(x) \leq (\alpha/2) \cdot n + O(\log n)$. However, by the correctness of A, it must hold that $\mathsf{rKt}(x) \geq \alpha \cdot n$. Hence we get a contradiction when n is sufficiently large. □

References

[CLO+23] Lijie Chen, Zhenjian Lu, Igor C. Oliveira, Hanlin Ren, and Rahul Santhanam. Polynomial-time pseudodeterministic construction of primes. In *Symposium on Foundations of Computer Science* (FOCS), pages 1261–1270, 2023.

[GHL+25] Halley Goldberg, Jinqiao Hu, Zhenjian Lu, Jingyi Lyu, and Igor C. Oliveira. Synergies between complexity theory and nondeterministic Kolmogorov complexity. *Preprint*, 2025.

[GK22] Halley Goldberg and Valentine Kabanets. A simpler proof of the worst-case to average-case reduction for polynomial hierarchy via symmetry of information. *Electron. Colloquium Comput. Complex.*, 7:1–14, 2022.

[GK23] Halley Goldberg and Valentine Kabanets. Improved learning from Kolmogorov complexity. In *Computational Complexity Conference* (CCC), pages 12:1–12:29, 2023.

[GK24] Halley Goldberg and Valentine Kabanets. Consequences of randomized reductions from SAT to time-bounded Kolmogorov complexity. In *Approximation, Randomization, and Combinatorial Optimization: Algorithms and Techniques* (APPROX/RANDOM), pages 51:1–51:19, 2024.

[GKLO22] Halley Goldberg, Valentine Kabanets, Zhenjian Lu, and Igor C. Oliveira. Probabilistic Kolmogorov complexity with applications to average-case complexity. In *Computational Complexity Conference* (CCC), pages 16:1–16:60, 2022.

[HIL+23] Shuichi Hirahara, Rahul Ilango, Zhenjian Lu, Mikito Nanashima, and Igor C. Oliveira. A duality between one-way functions and average-case symmetry of information. In *Symposium on Theory of Computing* (STOC), pages 1039–1050, 2023.

[Hir20] Shuichi Hirahara. Unexpected hardness results for Kolmogorov complexity under uniform reductions. In *Symposium on Theory of Computing* (STOC), pages 1038–1051, 2020.

[Hir21] Shuichi Hirahara. Average-case hardness of NP from exponential worst-case hardness assumptions. In *Symposium on Theory of Computing* (STOC), pages 292–302, 2021.

[Hir22] Shuichi Hirahara. Symmetry of information from meta-complexity. In *Computational Complexity Conference* (CCC), pages 26:1–26:41, 2022.

[Hir23] Shuichi Hirahara. Capturing one-way functions via NP-hardness of meta-complexity. In *Symposium on Theory of Computing* (STOC), pages 1027–1038, 2023.

[HKLO24] Shuichi Hirahara, Valentine Kabanets, Zhenjian Lu, and Igor C. Oliveira. Exact search-to-decision reductions for time-bounded Kolmogorov complexity. In *Computational Complexity Conference* (CCC), pages 29:1–29:56, 2024.

[HLN24] Shuichi Hirahara, Zhenjian Lu, and Mikito Nanashima. Optimal coding for randomized Kolmogorov complexity and its applications. In *Symposium on Foundations of Computer Science* (FOCS), pages 369–378, 2024.

[HLO24] Shuichi Hirahara, Zhenjian Lu, and Igor C. Oliveira. One-way functions and pKt complexity. In *Theory of Cryptography* (TCC), pages 253–286, 2024.

[HN23] Shuichi Hirahara and Mikito Nanashima. Learning in Pessiland via inductive inference. In *Symposium on Foundations of Computer Science,* (FOCS), pages 447–457, 2023.

[HN25] Shuichi Hirahara and Mikito Nanashima. Complexity-theoretic inductive inference. *Electron. Colloquium Comput. Complex.*, TR25-92, 2025.

[HW20] Shuichi Hirahara and Osamu Watanabe. On nonadaptive security reductions of hitting set generators. In *International Workshop on Randomization and Approximation Techniques* (RANDOM), pages 15:1–15:14, 2020.

[Ila23] Rahul Ilango. SAT reduces to the minimum circuit size problem with a random oracle. In *Symposium on Foundations of Computer Science* (FOCS), pages 733–742, 2023.

[IW01] Russell Impagliazzo and Avi Wigderson. Randomness vs time: Derandomization under a uniform assumption. *J. Comput. Syst. Sci.*, 63(4):672–688, 2001.

[KK25] Valentine Kabanets and Antonina Kolokolova. Chain rules for time-bounded Kolmogorov complexity. *Electron. Colloquium Comput. Complex.*, TR25-89, 2025.

[Lee06] Troy Lee. *Kolmogorov complexity and formula lower bounds*. PhD thesis, University of Amsterdam, 2006.

[Lev84] Leonid A. Levin. Randomness conservation inequalities; information and independence in mathematical theories. *Information and Control*, 61(1):15–37, 1984.

[Lev03] Leonid A. Levin. The tale of one-way functions. *Probl. Inf. Transm.*, 39(1):92–103, 2003.

[LM93] Luc Longpré and Sarah Mocas. Symmetry of information and one-way functions. *Inf. Process. Lett.*, 46(2):95–100, 1993.

[LO21] Zhenjian Lu and Igor C. Oliveira. An efficient coding theorem via probabilistic representations and its applications. In *International Colloquium on Automata, Languages, and Programming* (ICALP), pages 94:1–94:20, 2021.

[LORS24] Zhenjian Lu, Igor C. Oliveira, Hanlin Ren, and Rahul Santhanam. On the complexity of avoiding heavy elements. In *Symposium on Foundations of Computer Science* (FOCS), pages 2403–2412, 2024.

[LOS21] Zhenjian Lu, Igor C. Oliveira, and Rahul Santhanam. Pseudodeterministic algorithms and the structure of probabilistic time. In *ACM Symposium on Theory of Computing* (STOC), pages 303–316, 2021.

[LP20] Yanyi Liu and Rafael Pass. On one-way functions and Kolmogorov complexity. In *Symposium on Foundations of Computer Science* (FOCS), pages 1243–1254, 2020.

[LP22] Yanyi Liu and Rafael Pass. On one-way functions from NP-complete problems. In *Conference on Computational Complexity* (CCC), pages 36:1–36:24, 2022.

[LP23] Yanyi Liu and Rafael Pass. One-way functions and the hardness of (probabilistic) time-bounded Kolmogorov complexity w.r.t. samplable distributions. In *Annual Cryptology Conference* (CRYPTO), pages 645–673, 2023.

[LP25] Yanyi Liu and Rafael Pass. Hardness along the boundary towards one-way functions from the worst-case hardness of time-bounded Kolmogorov complexity. In *Annual Cryptology Conference* (CRYPTO), 2025.

[LR05] Troy Lee and Andrei E. Romashchenko. Resource bounded symmetry of information revisited. *Theor. Comput. Sci.*, 345(2-3):386–405, 2005.

[LS24] Zhenjian Lu and Rahul Santhanam. Impagliazzo's worlds through the lens of conditional Kolmogorov complexity. In *International Colloquium on Automata, Languages, and Programming* (ICALP), pages 110:1–110:17, 2024.

[LV19] Ming Li and Paul M. B. Vitányi. *An Introduction to Kolmogorov Complexity and Its Applications, 4th Edition*. Texts in Computer Science. Springer, 2019.

[LW95] Luc Longpré and Osamu Watanabe. On symmetry of information and polynomial time invertibility. *Inf. Comput.*, 121(1):14–22, 1995.

[Oli19] Igor C. Oliveira. Randomness and intractability in Kolmogorov complexity. In *International Colloquium on Automata, Languages, and Programming* (ICALP), pages 32:1–32:14, 2019.

[OS17a] Igor C. Oliveira and Rahul Santhanam. Conspiracies between learning algorithms, circuit lower bounds, and pseudorandomness. In *Computational Complexity Conference* (CCC), pages 18:1–18:49, 2017.

[OS17b] Igor C. Oliveira and Rahul Santhanam. Pseudodeterministic constructions in subexponential time. In *Symposium on Theory of Computing* (STOC), pages 665–677, 2017.

[Ron04] Detlef Ronneburger. *Kolmogorov Complexity and Derandomization*. PhD thesis, Rutgers University, 2004.

[San23] Rahul Santhanam. An algorithmic approach to uniform lower bounds. In *Computational Complexity Conference* (CCC), pages 35:1–35:26, 2023.

[SS22] Michael E. Saks and Rahul Santhanam. On randomized reductions to the random strings. In *Computational Complexity Conference* (CCC), pages 29:1–29:30, 2022.

[TV07] Luca Trevisan and Salil P. Vadhan. Pseudorandomness and average-case complexity via uniform reductions. *Computational Complexity*, 16(4):331–364, 2007.

[ZL70] Alexander K. Zvonkin and Leonid A. Levin. The complexity of finite objects and the algorithmic concepts of randomness and information. *UMN (Russian Math. Surveys)*, 25(6):83–124, 1970.