Optimal Coding Theorems in Time-Bounded Kolmogorov Complexity

Zhenjian Lu ✉
University of Warwick, Coventry, UK

Igor C. Oliveira ✉
University of Warwick, Coventry, UK

Marius Zimand ✉
Towson University, MD, USA

Abstract

The classical coding theorem in Kolmogorov complexity states that if an n-bit string x is sampled with probability δ by an algorithm with prefix-free domain then $\mathsf{K}(x) \leq \log(1/\delta) + O(1)$. In a recent work, Lu and Oliveira [31] established an unconditional time-bounded version of this result, by showing that if x can be efficiently sampled with probability δ then $\mathsf{rKt}(x) = O(\log(1/\delta)) + O(\log n)$, where rKt denotes the randomized analogue of Levin's Kt complexity. Unfortunately, this result is often insufficient when transferring applications of the classical coding theorem to the time-bounded setting, as it achieves a $O(\log(1/\delta))$ bound instead of the information-theoretic optimal $\log(1/\delta)$.

Motivated by this discrepancy, we investigate optimal coding theorems in the time-bounded setting. Our main contributions can be summarised as follows.

• **Efficient coding theorem for rKt with a factor of 2.** Addressing a question from [31], we show that if x can be efficiently sampled with probability at least δ then $\mathsf{rKt}(x) \leq (2 + o(1)) \cdot \log(1/\delta) + O(\log n)$. As in previous work, our coding theorem is *efficient* in the sense that it provides a polynomial-time probabilistic algorithm that, when given x, the code of the sampler, and δ, it outputs, with probability ≥ 0.99, a probabilistic representation of x that certifies this rKt complexity bound.

• **Optimality under a cryptographic assumption.** Under a hypothesis about the security of cryptographic pseudorandom generators, we show that no efficient coding theorem can achieve a bound of the form $\mathsf{rKt}(x) \leq (2 - o(1)) \cdot \log(1/\delta) + \mathsf{poly}(\log n)$. Under a weaker assumption, we exhibit a gap between *efficient* coding theorems and *existential* coding theorems with near-optimal parameters.

• **Optimal coding theorem for pK^t and unconditional Antunes-Fortnow.** We consider pK^t complexity [17], a variant of rKt where the randomness is public and the time bound is fixed. We observe the existence of an optimal coding theorem for pK^t, and employ this result to establish an *unconditional* version of a theorem of Antunes and Fortnow [5] which characterizes the worst-case running times of languages that are in average polynomial-time over all P-samplable distributions.

2012 ACM Subject Classification Theory of computation

Keywords and phrases computational complexity, randomized algorithms, Kolmogorov complexity

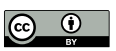
Digital Object Identifier 10.4230/LIPIcs.ICALP.2022.92

Category Track A: Algorithms, Complexity and Games

Related Version *Full Version*: `https://arxiv.org/abs/2204.08312` [33]

Funding M. Zimand was supported in part by the National Science Foundation through grant CCF 1811729. Z. Lu and I.C. Oliveira received support from the Royal Society University Research Fellowship URF\R1\191059 and from the EPSRC New Horizons Grant EP/V048201/1.

Acknowledgements We thank Bruno Bauwens for discussions and useful insights.



49th International Colloquium on Automata, Languages, and Programming (ICALP 2022).
Editors: Mikołaj Bojańczyk, Emanuela Merelli, and David P. Woodruff;
Article No. 92; pp. 92:1–92:14

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

1 Context and Background

A sampler is a probabilistic function that outputs Boolean strings. For any string $x \in \{0,1\}^*$ in its range, let $\mu(x)$ denote the probability with which x is generated. The Coding Theorem in Kolmogorov complexity states that if the sampler is computable and its domain is a prefix-free set, then for every x in its range

$$\mathsf{K}(x) \leq \log(1/\mu(x)) + O(1),$$

where $\mathsf{K}(\cdot)$ is the prefix-free Kolmogorov complexity. In other words, strings that are sampled with non-trivial probability have short representations. Note that the coding theorem achieves *optimal* expected length, since no uniquely decodable code can have expected length smaller than $\sum \mu(x) \log_2(1/\mu(x))$, the entropy of the sampler (the sum is over all x in the range of the sampler, assumed here to be finite).

The coding theorem is a central result in Kolmogorov complexity.[1] While it has found a number of applications in theoretical computer science (see, e.g., [28, 25, 1]), it comes with an important caveat: many aspects of the theory of Kolmogorov complexity are *non-constructive*. For instance, there is provably no algorithm that estimates $\mathsf{K}(x)$. Similarly, for arbitrary samplers, there is no effective compressor achieving the short representation provided by the coding theorem[2] and also no upper bound on the running time required to decompress x from it.

In order to translate results and techniques from Kolmogorov complexity to the setting of *efficient* algorithms and computations, several *time-bounded* variants of Kolmogorov complexity have been proposed. We refer to the book [29], thesis [25], and surveys [2, 3, 16, 4] for a comprehensive treatment of this area and its numerous applications to algorithms, complexity, cryptography, learning, and pseudorandomness, among other fields. We highlight that many exciting new results, which include worst-case to average-case reductions for NP problems [20, 21] and complexity-theoretic characterizations of one-way functions [30, 36], rely in a crucial way on time-bounded Kolmogorov complexity. These recent developments further motivate the investigation of key results from Kolmogorov complexity in the time-bounded setting.

In time-bounded Kolmogorov complexity we consider the minimum description length of a string x *with respect to machines that operate under a time constraint*. We informally review next two central notions in this area (see Section A for precise definitions). For a Turing machine $\mathcal{M}$, we let $|\mathcal{M}|$ denote its description length according to a fixed universal machine U. $\mathcal{M}(\varepsilon)$ denotes the computation of $\mathcal{M}$ over the empty string.

Kt Complexity. [26] This notion simultaneously considers description length and running time when measuring the complexity of a string x.

$$\mathsf{Kt}(x) \;=\; \min_{\text{TM } \mathcal{M},\, t \geq 1} \{|\mathcal{M}| + \log t \mid \mathcal{M}(\varepsilon) \text{ outputs } x \text{ in } t \text{ steps}\}\,.$$

K^t Complexity. [37] In contrast with Kt, here we fix the time bound $t\colon \mathbb{N} \to \mathbb{N}$, and consider the minimum description with respect to machines that run in time at most $t(|x|)$.

$$\mathsf{K}^t(x) \;=\; \min_{\text{TM } \mathcal{M}} \{|\mathcal{M}| \mid \mathcal{M}(\varepsilon) \text{ outputs } x \text{ in } t(|x|) \text{ steps}\}\,.$$

[1] For instance, [25] describes it as one of the four pillars of Kolmogorov complexity.

[2] However, there exists a probabilistic polynomial-time compressor that given x and an integer $m \geq \log(1/\mu(x))$ outputs a description of x of length m + small polylogarithmic overhead [6].

While Kt complexity is tightly related to optimal search algorithms (see [24] for a recent application), K^t is particularly useful in settings where maintaining a polynomial bound on the running time t is desired (see, e.g., [20]).

Antunes and Fortnow [5] introduced techniques that can be used to establish (conditional) coding theorems for K^t and Kt. In particular, if a sampler runs in *polynomial time* and outputs a string x with probability at least δ, then $\mathsf{Kt}(x) \leq \log(1/\delta) + O(\log n)$. Note that this coding theorem also achieves an optimal dependence on the probability parameter δ. However, the results of [5] rely on a strong derandomization assumption. For this reason, their application often lead to *conditional* results.

More recently, [31] established an *unconditional* coding theorem for a *randomized* analogue of Kt complexity. Before explaining their result, we review the definitions of rKt and rK^t.[3]

rKt Complexity. [35] In this definition, we consider randomized machines that output x with high probability.

$$\mathsf{rKt}(x) = \min_{\text{RTM } \mathcal{M},\, t \geq 1} \{|\mathcal{M}| + \log t \mid \mathcal{M}(\varepsilon) \text{ outputs } x \text{ in } t \text{ steps with probability} \geq 2/3\}.$$

rK^t Complexity. [10, 32][4] This is the randomized analogue of K^t, where the time bound t is fixed in advance.

$$\mathsf{rK}^t(x) = \min_{\text{RTM } \mathcal{M}} \{|\mathcal{M}| \mid \mathcal{M}(\varepsilon) \text{ outputs } x \text{ in } t(|x|) \text{ steps with probability} \geq 2/3\}.$$

In both cases, we can think of the randomized Turing machine $\mathcal{M}$ as a *probabilistic representation* of the input string x, in the sense that x can be recovered with high probability from its description. These measures allow us to employ methods from time-bounded Kolmogorov complexity in the setting of randomized computation, which is ubiquitous in modern computer science. For instance, [35, 32] employed rKt and rK^t to obtain bounds on the compressibility of prime numbers and other objects and to show that certain problems about time-bounded Kolmogorov complexity can be intractable. We note that, under derandomization assumptions (see [35]), for every string x, $\mathsf{rKt}(x) = \Theta(\mathsf{Kt}(x))$. Similarly, one can conditionally show that $\mathsf{K}^t(x)$ is essentially $\mathsf{rK}^t(x)$, up to a $O(\log |x|)$ additive term (see [17]). Consequently, insights obtained in the context of probabilistic notions of Kolmogorov complexity can often inform the study of more classical notions such as Kt and K^t.

Among other results, [31] established the following unconditional coding theorem in time-bounded Kolmogorov complexity: if a sampler runs in polynomial time and outputs a string x with probability at least δ, then $\mathsf{rKt}(x) = O(\log(1/\delta) + O(\log n)$. While this result can be used to port some applications of the coding theorem from Kolmogorov complexity to the time-bounded setting, in many cases it is still insufficient. This is because its dependence on the probability parameter δ is not optimal, which is often crucial in applications (see, e.g., [5, 1]).

2 Results

In this work, we investigate optimal coding theorems in time-bounded Kolmogorov complexity. We describe our results next.

[3] See Appendix A for a formal treatment.
[4] [10] refers to this notion as CBP^t complexity.

2.1 A Tighter Efficient Coding Theorem

Our first result addresses the question posed in [31, Problem 37].

▶ **Theorem 1.** *Suppose there is an efficient algorithm A for sampling strings such that $A(1^n)$ outputs a string $x \in \{0,1\}^n$ with probability at least δ. Then*

$$\mathsf{rKt}(x) \;\leq\; 2\log(1/\delta) + O\big(\log n + \log^2 \log(1/\delta)\big)\,,$$

where the constant behind the $O(\cdot)$ depends on A and is independent of the remaining parameters. Moreover, given x, the code of A, and δ, it is possible to compute in time $\mathsf{poly}(n, |A|)$, with probability ≥ 0.99, a probabilistic representation of x certifying this rKt-complexity bound.

In [9, Lemma 4], it was observed that by hashing modulo prime numbers one can obtain short descriptions of strings. As discussed in [31, Section A.2.1], for each efficient sampling algorithm, this technique implies that if some string x is produced with probability $\geq \delta$, then $\mathsf{rKt}(x) \leq 3\log(1/\delta) + O(\log n)$.[5] In contrast, Theorem 1 achieves a bound of the form $(2 + o(1)) \cdot \log(1/\delta) + O(\log n)$.

Theorem 1 readily improves some parameters in the applications of the coding theorem for rKt discussed in [31], such as the efficient instance-based search-to-decision reduction for rKt. We omit the details.

In [33, Section 3.1], we discuss extensions of this result. In particular, we describe precise bounds on the running time used in producing the corresponding probabilistic representation, and discuss computational aspects of the compression and decompression of x in detail. In [33, Appendix A], we discuss the computation of a probabilistic representation of the string x when one does not know a probability bound δ.

2.2 Matching Lower Bound Under a Cryptographic Assumption

It is possible to extend techniques from [5] to show the following conditional result (see [33, Section 3.2]).

▶ **Proposition 2.** *Assume there is a language $L \in \mathsf{BPTIME}\left[2^{O(n)}\right]$ that requires nondeterministic circuits of size $2^{\Omega(n)}$ for all but finitely many n. Suppose there is an efficient algorithm A for sampling strings such that $A(1^n)$ outputs a string $x \in \{0,1\}^n$ with probability at least $\delta > 0$. Then*

$$\mathsf{rKt}(x) \;\leq\; \log(1/\delta) + O(\log n).$$

While Proposition 2 provides a better bound than Theorem 1, the result is only *existential*, i.e., it does not provide an efficient algorithm that produces a probabilistic representation of x. In other words, Proposition 2 does not establish an *efficient* coding theorem. Our next result shows that the bound achieved by Theorem 1 is optimal for efficient coding theorems, under a cryptographic assumption.

[5] The bound from [31, Section A.2.1] is different because it does not take into account the running time, which incurs an additional overhead of $\log(1/\delta)$.

The Cryptographic Assumption. For a constant $\gamma \in (0,1)$, we introduce the γ-Crypto-ETH assumption, which can be seen as a cryptographic analogue of the well-known exponential time hypothesis about the complexity of k-CNF SAT [23]. Informally, we say that γ-Crypto-ETH holds if there is a pseudorandom generator $G\colon \{0,1\}^{\ell(n)} \to \{0,1\}^n$ computable in time $\mathrm{poly}(n)$ that fools *uniform algorithms* running in time $2^{\gamma \cdot \ell(n)}$. Any seed length $(\log n)^{\omega(1)} \leq \ell(n) \leq n/2$ is sufficient in our negative results.

In analogy with the well-known ETH and SETH hypotheses about the complexity of k-CNF SAT, we say that Crypto-ETH holds if γ-Crypto-ETH is true for some $\gamma > 0$, and that Crypto-SETH holds if γ-Crypto-ETH is true for every $\gamma \in (0,1)$. Since a candidate PRG of seed length $\ell(n)$ can be broken in time $2^{\ell(n)}\mathsf{poly}(n)$ by trying all possible seeds, these hypotheses postulate that for some PRGs one cannot have an attack that does sufficiently better than this naive brute-force approach.

We stress that these assumptions refer to uniform algorithms. In the case of non-uniform distinguishers, it is known that Crypto-SETH does not hold (see [15, 14, 13] and references therein). We provide a formal treatment of the cryptographic assumption in [33, Section 4].

▶ **Theorem 3** (Informal). *Let $\gamma \in (0,1)$ be any constant. If γ-Crypto-ETH holds, there is no efficient coding theorem for* rKt *that achieves bounds of the form* $(1+\gamma-o(1)) \cdot \log(1/\delta) + \mathsf{poly}(\log n)$.

Theorem 3 shows that if Crypto-ETH holds then the best parameter achieved by an *efficient* coding theorem for rKt is $(1+\Omega(1)) \cdot \log(1/\delta) + \mathsf{poly}(\log n)$. This exhibits an inherent gap in parameters between the efficient coding theorem (Theorem 1) and its existential analogue (Proposition 2). On the other hand, if the stronger Crypto-SETH hypothesis holds, then no efficient coding theorem for rKt achieves parameter $(2-o(1)) \cdot \log(1/\delta) + \mathsf{poly}(\log n)$. In this case, Theorem 1 is essentially optimal with respect to its dependence on δ.

Fine-grained complexity of coding algorithms for polynomial-time samplers. An rKt bound refers to the time necessary to *decompress* a string x from its probabilistic representation. On the other hand, an *efficient* coding theorem provides a routine that can *compress* x in polynomial time. More generally, a coding procedure for a sampler A consists of a pair of probabilistic algorithms (Compress, Decompress) that aim to produce a "good" codeword p for every string y sampled by A. The quality of p depends on three values: the length of p, the number of steps t_C used to produce p from y (the compression time), and the number of steps t_D used to produce y from p (the decompression time). It is interesting to understand the trade-off between these three values. Toward this goal, we aggregate them in a manner similar to rKt, by defining the 2-sided-rKt complexity of y to be, roughly, $|p| + \log(t_C + t_D)$ (the formal definition, see [33, Definition 25]), is more complicated because it takes into account that Compress and Decompress are probabilistic). Thus according to 2-sided-rKt, each bit gained by a shorter codeword is worth doubling the compression/decompression time. For instance, for simple samplers (say, having a finite range, or generating strings with the uniform distribution), there exist trivial polynomial time Compress and Decompress, which in case y is sampled with probability at least δ, produce a codeword p with $|p| = \log(1/\delta)$ (provided Compress and Decompress know δ). Such a coding procedure certifies for each sampled string a 2-sided-rKt complexity of $\log(1/\delta) + O(\log n)$. We say that the sampler admits coding with 2-sided-rKt complexity bounded by $\log(1/\delta) + O(\log n)$. In general, we have to include also the error probability of Compress and Decompress, which we omit in this informal discussion.

Similarly to Theorem 1 and Theorem 3 (and also with similar proofs), we establish the following theorem.

▶ **Theorem 4** (Informal). *The following results hold.*

(a) (Upper Bound) *Every polynomial-time sampler admits coding with* 2*-sided-*rKt *complexity* $2\log(1/\delta) + O(\log^2\log(1/\delta)) + O(\log n)$.

(b) (Conditional Lower Bound) *Let* $\gamma \in (0,1)$ *be any constant. If* γ*-*Crypto-ETH *holds, there exists a polynomial-time sampler that does not admit coding with* 2*-sided-*rKt *complexity bounded by* $(1+\gamma-o(1))\cdot\log(1/\delta) + \mathsf{poly}(\log n)$*, unless the error probability is greater than* $1/7$.

2.3 An Optimal Coding Theorem and Unconditional Antunes-Fortnow

While Theorem 1 improves the result from [31] to achieve a bound that is tight up to a factor of 2 and that is possibly optimal among efficient coding theorems, it is still insufficient in many applications. We consider next a variant of rKt that allows us to establish an *optimal* and *unconditional* coding theorem in time-bounded Kolmogorov complexity.

Fix a function $t\colon \mathbb{N}\to\mathbb{N}$. For a string $x\in\{0,1\}^*$, the probabilistic t-bounded Kolmogorov complexity of x (see [17]) is defined as

$$\mathsf{pK}^t(x) = \min\left\{ k \;\middle|\; \mathop{\mathbf{Pr}}_{w\sim\{0,1\}^{t(|x|)}}\left[\exists\, \mathcal{M}\in\{0,1\}^k,\ \mathcal{M}(w) \text{ outputs } x \text{ within } t(|x|) \text{ steps}\right] \geq \frac{2}{3} \right\}.$$

In other words, if $k = \mathsf{pK}^t(x)$, then with probability at least $2/3$ over the choice of the random string w, x admits a time-bounded encoding of length k. In particular, if two parties share a typical random string w, then x can be transmitted with k bits and decompressed in time $t = t(|x|)$. (Recall that here the time bound t is fixed, as opposed to rKt, where a $\log t$ term is added to the description length.)

It is possible to show that $\mathsf{K}^t(x)$, $\mathsf{rK}^t(x)$, and $\mathsf{pK}^t(x)$ correspond essentially to the same time-bounded measure, under standard derandomization assumptions [17].[6] One of the main benefits of pK^t is that it allows us to establish unconditional results that are currently unknown in the case of the other measures.[7]

▶ **Theorem 5.** *Suppose there is a randomized algorithm* A *for sampling strings such that* $A(1^n)$ *runs in time* $T(n)\geq n$ *and outputs a string* $x\in\{0,1\}^n$ *with probability at least* $\delta>0$. *Then*

$$\mathsf{pK}^t(x) \;=\; \log(1/\delta) + O(\log T(n))\,,$$

where $t(n) = \mathrm{poly}(T(n))$ *and the constant behind the* $O(\cdot)$ *depends on* $|A|$ *and is independent of the remaining parameters.*

Theorem 5 provides a time-bounded coding theorem that can be used in settings where the optimal dependence on δ is crucial. As an immediate application, it is possible to show an equivalence between efficiently sampling a fixed sequence $w_n\in\{0,1\}^n$ of objects (e.g., n-bit prime numbers) with probability at least $\delta_n/\mathsf{poly}(n)$ and the existence of bounds for the

---

[6] More precisely, under standard derandomization assumptions, $\mathsf{pK}^t(x)$ and $\mathsf{rK}^{t'}(x)$ coincide up to an additive term of $O(\log|x|)$, provided that $t' = \mathrm{poly}(t)$. A similar relation holds between K^t and rK^t.

[7] While in this work we focus on coding theorems, we stress that pK^t is a key notion introduced in [17] that enables the investigation of meta-complexity in the setting of probabilistic computations. It has applications in worst-case to average-case reductions and in learning theory.

corresponding objects of the form $\mathsf{pK}^{\mathrm{poly}}(w_n) = \log(1/\delta_n) + O(\log n)$.[8] This is the first tight equivalence of this form in time-bounded Kolmogorov complexity that does not rely on an unproven assumption.

As a more sophisticated application of Theorem 5, we establish an unconditional form of the main theorem from Antunes and Fortnow [5], which provides a characterization of the worst-case running times of languages that are in average polynomial-time over all P-samplable distributions.

We recall the following standard notion from average-case complexity (see, e.g., [8]). For an algorithm A that runs in time $T_A\colon \{0,1\}^* \to \mathbb{N}$ and for a distribution $\mathcal{D}$ supported over $\{0,1\}^*$, we say that A runs in polynomial-time on average with respect to $\mathcal{D}$ if there is some constant $\varepsilon > 0$ such that

$$\mathop{\mathbf{E}}_{x\sim\mathcal{D}}\left[\frac{T_A(x)^\varepsilon}{|x|}\right] < 1.$$

As usual, we say that a distribution $\mathcal{D}$ is P-samplable if it can be sampled in polynomial time.

▶ **Theorem 6.** *The following conditions are equivalent for any language $L \subseteq \{0,1\}^*$.*

(i) *For every P-samplable distribution $\mathcal{D}$, L can be solved in polynomial-time on average with respect to $\mathcal{D}$.*

(ii) *For every polynomial p, there exists a constant $b > 0$ such that the running time of some algorithm that computes L is bounded by $2^{O(\mathsf{pK}^p(x) - \mathsf{K}(x) + b\log(|x|))}$ for every input x.*

In contrast, [5] shows a *conditional* characterisation result that employs K^t complexity in the expression that appears in Item (*ii*).

3 Techniques

In this section, we provide an informal overview of our proofs and techniques.

Efficient Coding Theorem for rKt (Theorem 1). Breaking down the result into its components, Theorem 1 shows that for any polynomial-time sampler A, there exist a probabilistic polynomial-time algorithm Compress and an algorithm Decompress with the following properties: Compress on input an n-bit string x and δ (which estimates from below the probability with which A samples x), returns a codeword c_x of length $\log(1/\delta) + \mathrm{poly}(\log n)$ such that Decompress with probability ≥ 0.99 reconstructs x in time $1/\delta \cdot \exp(\mathrm{poly}(\log n))$. Note that the probabilistic representation of x certifying the rKt bound in Theorem 1 can be obtained from the codeword c_x and Decompress, and that obtaining a running time with a factor of $(1/\delta)^{1+o(1)}$ is crucial in order to get a final rKt bound of the form $(2 + o(1)) \cdot \log(1/\delta)$. (Actually, Compress does not have to depend on A, the 0.99 can be $1 - \varepsilon$ for arbitrary $\varepsilon > 0$, and the $\mathrm{poly}(\log n)$ term is $O(\log n + \log^2 \log(1/\delta))$, but we omit these details in our discussion). We explain what are the challenges in obtaining Compress and Decompress and how they are overcome. We remark that the construction is different from the approaches described in [31].

[8] An efficient sampler immediately implies the corresponding pK^t bounds via Theorem 5. On the other hand, objects of bounded pK^t complexity can be sampled by considering a random sequence of bits and a random program of appropriate length. We refer to [31, Theorem 6] for a weaker relation and its proof. Since the argument is essentially the same, we omit the precise details.

Decompress can run the sampler $K := O(1/\delta)$ times and obtain a list of elements S^* (the list of suspects) that with high probability contains x. Compress has to provide information that allows Decompress to prune S^* and find x. Since the algorithms do not share randomness, Compress does not know S^*, and so compression has to work for any $S \subseteq \{0,1\}^n$ of size K, only assuming that $x \in S$. Compress can use a bipartite lossless expander graph G, which is a graph with the property that any set S of left nodes with size $|S| \leq K$ has at least $(1-\varepsilon)D|S|$ neighbors, where D is the left degree. Such graphs are called $((1-\varepsilon)D, K)$ lossless expanders and they have numerous applications (see e.g., [11, 22]). An extension of Hall's matching theorem shows that for any set S of K left nodes, there is a matching that assigns to each $x \in S$, $(1-\varepsilon)D$ of its neighbors, so that no right node is assigned twice (i.e., the matching defines a subgraph with no collisions). Compress can just pick the codeword c_x to be one random neighbor of x. Then, Decompress can do the pruning of S^* as follows. Having S^* and c_x, it does the matching, and, since with probability $1-\varepsilon$, c_x is only assigned to x, Decompress can find x. There is one problem though. The algorithms for maximum matching in general bipartite graphs do not run in linear time (see [12, 34], and the references therein). Therefore, the decompression time would have a dependency on δ too large for us. Fortunately, lossless expanders can be used to do "almost" matching faster. [6] introduces *invertible functions* (see [33, Definition 12]) for the more demanding task in which the elements of S appear one-by-one and the matching has to be done in the online manner. We do not need online matching, but we take advantage of the construction in [6] to obtain a fast matching algorithm. It follows from [6], that in a lossless expander it is possible to do a greedy-type of "almost" matching, which means that every left node in S is matched to $(1-\varepsilon)D$ of its neighbors (exactly what we need), but with $\mathrm{poly}(\log n)$ collisions. The collisions can be eliminated with some additional standard hashing (see the discussion on [33, Page 14] for details). As we explain on [33, Page 14], this leads to decompression time $K \cdot D \cdot \mathrm{poly}(n)$ and the length of the codeword c_x is $\log |R| + |\text{hash-code}|$, where R is the right set of the lossless expander. To obtain our result, the degree D has to be $2^{\mathrm{poly}(\log n)}$ and $|R|$ has to be $K \cdot 2^{\mathrm{poly}(\log n)}$.

Building on results and techniques from [18], [6] constructs a $((1-\varepsilon)D, K)$ explicit lossless expander with left side $\{0,1\}^n$, degree $D = 2^d$ for $d = O(\log(n/\varepsilon) \cdot \log k)$, and right side R, with size verifying $\log |R| = k + \log(n/\varepsilon) \cdot \log k$ (where $k := \log K$). To obtain in Theorem 1 the dependency on n to be $O(\log n)$ (which is optimal up to the constant in $O(\cdot)$), we show the existence of a $((1-\varepsilon)D, K)$ explicit expander with $d = O(\log n + \log(k/\varepsilon) \cdot \log k)$ and $\log |R| = k + O(\log n + \log(k/\varepsilon) \cdot \log k)$. This lossless expander is constructed by a simple composition of the above lossless expander from [6] with a lossless expander from [18], with an appropriate choice of parameters (see [33, Section 3.1.1]).

Conditional Lower Bound for Efficient Coding Theorems (Theorem 3). Our goal is to show that there is no *efficient* coding theorem for rKt that achieves bounds of the form $(1+\gamma-o(1)) \cdot \log(1/\delta) + \mathrm{poly}(\log n)$, under the assumption that γ-Crypto-ETH holds for $\gamma \in (0,1)$. We build on an idea attributed to L. Levin (see e.g. [25, Section 5.3]). To provide an overview of the argument, let $G_n \colon \{0,1\}^{\ell(n)} \to \{0,1\}^n$ be a cryptographic generator of seed length $\ell(n) = n/2$ witnessing that γ-Crypto-ETH holds. In other words, G_n has security $2^{\gamma \cdot \ell(n)}$ against uniform adversaries. We define a sampler S_n as follows. On input $x \in \{0,1\}^n$, which we interpret as a random string, it outputs $G_n(x')$, where x' is the prefix of x of length $\ell(n)$. We argue that if an *efficient* algorithm F is able to compress every string y in the support of $\mathsf{Dist}(S_n)$, the distribution induced by the sampler S_n, to an rKt encoding of complexity $(1+\gamma-\varepsilon) \cdot \log(1/\delta'(y)) + C \cdot (\log n)^C$, where $\delta'(y)$ is a lower bound on $\delta(y)$ (the probability of y under $\mathsf{Dist}(S_n)$), we can use F to break G_n. (Note that F expects as input n, y, δ', and $\mathsf{code}(S)$.)

The (uniform) distinguisher D computes roughly as follows. Given a string $z \in \{0,1\}^n$, which might come from the uniform distribution U_n or from $G_n(U_{\ell(n)}) \equiv \mathsf{Dist}(S_n)$, D attempts to use F to *compress* z to a "succinct" representation, then checks if the computed representation *decompresses* to the original string z. If this is the case, it outputs 1, otherwise it outputs 0. (Note that we haven't specified what "succinct" means, and it is also not immediately clear how to run F, since it assumes knowledge of a probability bound δ'. For simplicity of the exposition, we omit this point here.) We need to argue that a test of this form can be implemented in time $2^{\gamma \cdot \ell(n)}$, and that it distinguishes the output of G from a random string.

To achieve these goals, first note that a typical random string cannot be compressed to representations of length, say, $n - \mathsf{poly}(\log n)$, even in the much stronger sense of (time-unbounded) Kolmogorov complexity. Therefore, with some flexibility with respect to our threshold for succinctness, the proposed distinguisher is likely to output 0 on a random string. On the other hand, if F implements an efficient coding theorem that achieves rKt encodings of complexity $(1+\gamma-\varepsilon)\cdot\log(1/\delta'(y))+\mathsf{poly}(\log n)$, the following must be true. Using that the expected *encoding length* of *any* (prefix-free) encoding scheme is at least $H(\mathsf{Dist}(S_n))$, where $\mathsf{Dist}(S_n)$ is the distribution of strings sampled by S_n and H is the entropy function, we get (via a slightly stronger version of this result) that a non-trivial measure of strings y in the support of $\mathsf{Dist}(S_n)$ have rKt *encoding length* at least $(1-\varepsilon/4)\cdot\log(1/\delta(y))$. Consequently, for such strings, an upper bound on rKt complexity of $(1+\gamma-\varepsilon)\cdot\log(1/\delta'(y))+\mathsf{poly}(\log n)$ when $\delta'(y)$ is sufficiently close to $\delta(y)$ implies that the running time t of the underlying machine satisfies $\log t \leq (\gamma-\varepsilon/2)\log(1/\delta(y))+\mathsf{poly}(\log n)$. Using that $\ell(n) = n/2$ and $\delta(y) \geq 2^{-\ell(n)}$ for any string y in the support of $\mathsf{Dist}(S_n)$, it is easy to check that (asymptotically) $t \leq 2^{(\gamma-\varepsilon/4)\cdot\ell(n)}$. For this reason, we can implement a (slightly modified) distinguisher D in time less than $2^{\gamma\cdot\ell(n)}$, by trying different approximations $\delta'(z)$ for an input string z and by running the decompressor on the produced representation for at most t steps on each guess for $\delta(z)$. By our previous discussion, a non-trivial measure of strings from $\mathsf{Dist}(S_n)$ will be accepted by D, while only a negligible fraction of the set of all strings (corresponding to the random case) will be accepted by D.

Implementing this strategy turns out to be more subtle than this. This happens because F is a *probabilistic* algorithm which does not need to commit to a *fixed* succinct encoding. We refer to the formal presentation in [33, Section 4] for details, where we also discuss the bound on the seed length $\ell(n)$.

Coding Theorem for pK^t (Theorem 5) and Unconditional [5] (Theorem 6). The proof of our optimal coding theorem for pK^t builds on that of the *conditional* coding theorem for K^t from [5], which can be viewed as a two-step argument. Roughly speaking, the first step is to show that if there is a polynomial-time sampler that outputs a string $x \in \{0,1\}^n$ with probability δ, then the polynomial-time-bounded Kolmogorov complexity of x is about $\log(1/\delta) + O(\log n)$ *if we are given a random string*. After this, they "derandomize" the use of random strings using a certain pseudorandom generator, which exists under a strong derandomization assumption. Our key observation is that the use of random strings arises naturally in probabilistic Kolmogorov complexity, and particularly in this case the random strings can be "embedded" into the definition of pK^t. As a result, we don't need to perform the afterward derandomization as in original proof of [5], and hence get rid of the derandomization assumption.

Next, we describe how to use Theorem 5, together with other useful properties of pK^t, to obtain an unconditional version of Antunes and Fortnow's main result. Let μ be a Kolmogorov complexity measure, such as $\mathsf{K}^{\mathsf{poly}}$, $\mathsf{rK}^{\mathsf{poly}}$ or $\mathsf{pK}^{\mathsf{poly}}$. The key notion in the proof

is the distribution (in fact, a class of semi-distributions) called m_μ, which is defined as $m_\mu(x) := 1/2^{\mu(x)}$. More specifically, following [5], it is not hard to show that, for every language L, L can be decided in polynomial-time on average with respect to m_μ if and only if its worst-case running time is $2^{O(\mu(x)-\mathsf{K}(x))}$ on input x (see [33, Lemma 25]). Then, essentially, to show our result we argue that L can be decided in polynomial time on average with respect to m_μ if and only if the same holds with respect to all P-samplable distributions.

Recall that if a distribution $\mathcal{D}$ *dominates* another distribution $\mathcal{D}'$ (i.e., $\mathcal{D}(x) \gtrsim \mathcal{D}'(x)$ for all x) and L is polynomial-time on average with respect to $\mathcal{D}$, then the same holds with respect to $\mathcal{D}'$ (see Definition 9 and Fact 10). Therefore, to replace m_μ above with P-samplable distributions, it suffices to show that m_μ is "universal" with respect to the class of P-samplable distributions, in the following sense.

1. m_μ dominates *every* P-samplable distribution. (This is essentially an optimal source coding theorem for the Kolmogorov measure μ.)
2. m_μ is dominated by *some* P-samplable distribution.

The above two conditions require two properties of the Kolmogorov measure μ that are somewhat conflicting: the first condition requires the notion of μ to be *general* enough so that m_μ can "simulate" *every* P-samplable distribution, while the second condition needs μ to be *restricted* enough so that m_μ can be "simulated" by *some* P-samplable (i.e., simple) distribution. For example, if μ is simply the time-unbounded Kolmogorov complexity K (or even the polynomial-space-bounded variant), then it is easy to establish an optimal source coding theorem for such a general Kolmogorov measure; however it is unclear how to sample in polynomial-time a string x with probability about $1/2^{\mathsf{K}(x)}$, so in this case μ does not satisfy the second condition. On the other hand, if μ is some restricted notion of time-bounded Kolmogrov complexity measure such as $\mathsf{K}^{\mathsf{poly}}$ or $\mathsf{rK}^{\mathsf{poly}}$, then one can obtain polynomial-time samplers that sample x with probability about $1/2^{\mathsf{K}^{\mathrm{poly}}(x)}$ or $1/2^{\mathsf{K}^{\mathrm{poly}}(x)}$ (up to a polynomial factor); however, as in [5], we only know how to show an *optimal* source coding theorem for $\mathsf{K}^{\mathsf{poly}}$ (or $\mathsf{rK}^{\mathsf{poly}}$) under a derandomization assumption. Therefore, in this case μ does not satisfies the first condition. Our key observation is that the notion $\mathsf{pK}^{\mathsf{poly}}$, which sits in between K and $\mathsf{K}^{\mathrm{poly}}$ (or $\mathsf{rK}^{\mathrm{poly}}$),[9] satisfies both conditions described above (see [33, Lemmas 36 and 37]).

4 Concluding Remarks and Open Problems

Our results indicate that Theorem 1 might be optimal among *efficient* coding theorems for rKt, i.e., those that efficiently produce representations matching the existential bounds. In the case of pK^t, the corresponding coding theorem (Theorem 5) is optimal. We have described a concrete application of Theorem 5 (Theorem 6). A second application appears in [17]. In both cases, achieving an optimal dependence on the probability parameter δ is critical, and for this reason, the result from [31] is not sufficient.

Naturally, we would like to understand the possibility of establishing an *unconditional* coding theorem for rKt with an optimal dependence on the probability parameter δ. While the validity of $\mathsf{Crypto\text{-}ETH}$ implies that no *efficient* coding theorem with this property exist, we have an *existential* coding theorem of this form under a derandomization assumption (Proposition 2). In the case of K^t complexity, it is known that an unconditional coding theorem with optimal dependence on δ implies that $\mathsf{EXP} \neq \mathsf{BPP}$ (see [25, Theorem 5.3.4]).

[9] We can show that for every $x \in \{0,1\}^*$ and every computable time bound $t\colon \mathbb{N} \to \mathbb{N}$, $\mathsf{K}(x) \lesssim \mathsf{pK}^t(x) \leq \mathsf{rK}^t(x) \leq \mathsf{K}^t(x)$.

However, the techniques behind this connection do not seem to lead to an interesting consequence in the case of rKt and rK^t. Consequently, an optimal coding theorem for rKt might be within the reach of existing techniques.

It would also be interesting to establish Theorem 3 under a weaker assumption, or to refute Crypto-SETH. A related question is the possibility of basing Crypto-ETH on the existence of one-way functions of exponential hardness. Existing reductions are not strong enough to provide an equivalence between one-way functions and cryptographic pseudorandomness in the exponential regime (see [38, 19]).

Finally, are there more applications of pK^t complexity and of Theorem 5? Since this coding theorem is both optimal and unconditional, we expect more applications to follow.

References

1 Scott Aaronson. The equivalence of sampling and searching. *Theory Comput. Syst.*, 55(2):281–298, 2014.

2 Eric Allender. Applications of time-bounded Kolmogorov complexity in complexity theory. In *Kolmogorov complexity and computational complexity*, pages 4–22. Springer, 1992.

3 Eric Allender. When worlds collide: Derandomization, lower bounds, and Kolmogorov complexity. In *International Conference on Foundations of Software Technology and Theoretical Computer Science* (FSTTCS), pages 1–15. Springer, 2001.

4 Eric Allender. The complexity of complexity. In *Computability and Complexity*, pages 79–94. Springer, 2017.

5 Luis Filipe Coelho Antunes and Lance Fortnow. Worst-case running times for average-case algorithms. In *Conference on Computational Complexity* (CCC), pages 298–303, 2009.

6 Bruno Bauwens and Marius Zimand. Universal almost optimal compression and Slepian-Wolf coding in probabilistic polynomial time. *CoRR*, abs/1911.04268, 2019. arXiv:1911.04268.

7 Shai Ben-David, Benny Chor, Oded Goldreich, and Michael Luby. On the theory of average case complexity. *J. Comput. Syst. Sci.*, 44(2):193–219, 1992. doi:10.1016/0022-0000(92)90019-F.

8 Andrej Bogdanov and Luca Trevisan. Average-case complexity. *Found. Trends Theor. Comput. Sci.*, 2(1), 2006.

9 Harry Buhrman, Lance Fortnow, and Sophie Laplante. Resource-bounded Kolmogorov complexity revisited. *SIAM J. Comput.*, 31(3):887–905, 2001.

10 Harry Buhrman, Troy Lee, and Dieter van Melkebeek. Language compression and pseudorandom generators. *Comput. Complex.*, 14(3):228–255, 2005.

11 M. R. Capalbo, O. Reingold, S. P. Vadhan, and A. Wigderson. Randomness conductors and constant-degree lossless expanders. In *STOC*, pages 659–668, 2002. doi:10.1145/509907.510003.

12 Li Chen, Rasmus Kyng, Yang P. Liu, Richard Peng, Maximilian Probst Gutenberg, and Sushant Sachdeva. Maximum flow and minimum-cost flow in almost-linear time. *CoRR*, abs/2203.00671, 2022. doi:10.48550/arXiv.2203.00671.

13 Kai-Min Chung, Siyao Guo, Qipeng Liu, and Luowen Qian. Tight quantum time-space tradeoffs for function inversion. In *Symposium on Foundations of Computer Science* (FOCS), pages 673–684, 2020.

14 Anindya De, Luca Trevisan, and Madhur Tulsiani. Time space tradeoffs for attacks against one-way functions and PRGs. In *Annual International Cryptology Conference* (CRYPTO), pages 649–665, 2010.

15 Amos Fiat and Moni Naor. Rigorous time/space trade-offs for inverting functions. *SIAM J. Comput.*, 29(3):790–803, 1999. doi:10.1137/S0097539795280512.

16 Lance Fortnow. Kolmogorov complexity and computational complexity. *Complexity of Computations and Proofs. Quaderni di Matematica*, 13, 2004.

17 Halley Goldberg, Valentine Kabanets, Zhenjian Lu, and Igor C. Oliveira. Probabilistic Kolmogorov complexity with applications to average-case complexity. Preprint, 2022.

18 Venkatesan Guruswami, Christopher Umans, and Salil P. Vadhan. Unbalanced expanders and randomness extractors from Parvaresh-Vardy codes. *J. ACM*, 56(4):20:1–20:34, 2009.

19 Iftach Haitner, Omer Reingold, and Salil P. Vadhan. Efficiency improvements in constructing pseudorandom generators from one-way functions. *SIAM J. Comput.*, 42(3):1405–1430, 2013.

20 Shuichi Hirahara. Non-black-box worst-case to average-case reductions within NP. In *Symposium on Foundations of Computer Science* (FOCS), pages 247–258, 2018.

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

22 S. Hoory, N. Linial, and A. Wigderson. Expander graphs and their applications. *Bull. Amer. Math. Soc.*, 43:439–561, 2006.

23 Russell Impagliazzo and Ramamohan Paturi. On the complexity of k-SAT. *J. Comput. Syst. Sci.*, 62(2):367–375, 2001. doi:10.1006/jcss.2000.1727.

24 Jan Krajíček. Information in propositional proofs and algorithmic proof search. *The Journal of Symbolic Logic*, pages 1–22, 2021.

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

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

27 Leonid A. Levin. Average case complete problems. *SIAM J. Comput.*, 15(1):285–286, 1986. doi:10.1137/0215020.

28 Ming Li and Paul M. B. Vitányi. Average case complexity under the universal distribution equals worst-case complexity. *Inf. Process. Lett.*, 42(3):145–149, 1992.

29 Ming Li and Paul M. B. Vitányi. *An introduction to Kolmogorov complexity and its applications.* Springer-Verlag, 2019. 4th edition (1st edition in 1993).

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

31 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.

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

33 Zhenjian Lu, Igor C. Oliveira, and Marius Zimand. Optimal coding theorems in time-bounded Kolmogorov complexity, 2022. doi:10.48550/ARXIV.2204.08312.

34 Aleksander Madry. Navigating central path with electrical flows: From flows to matchings, and back. In *Symposium on Foundations of Computer Science* (FOCS), pages 253–262, 2013. doi:10.1109/FOCS.2013.35.

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

36 Hanlin Ren and Rahul Santhanam. Hardness of KT characterizes parallel cryptography. In *Computational Complexity Conference* (CCC), pages 35:1–35:58, 2021.

37 Michael Sipser. A complexity theoretic approach to randomness. In *Symposium on Theory of Computing* (STOC), pages 330–335, 1983.

38 Salil P. Vadhan and Colin Jia Zheng. A uniform min-max theorem with applications in cryptography. In *Annual Cryptology Conference* (CRYPTO), pages 93–110, 2013.

A Definitions and Basic Results

Time-bounded Kolmogorov complexity. For a function $t\colon \mathbb{N} \to \mathbb{N}$, a string x, and a universal Turing machine U, let the time-bounded Kolmogorov complexity be defined as

$$\mathsf{K}^t_U(x) = \min_{p\in\{0,1\}^*} \{|p| \mid U(p) \text{ outputs } x \text{ in at most } t(|x|) \text{ steps}\}.$$

A machine U is said to be *time-optimal* if for every machine M there exists a constant c such that for all $x \in \{0,1\}^n$ and $t\colon \mathbb{N} \to \mathbb{N}$ satisfying $t(n) \geq n$,

$$\mathsf{K}_U^{ct \log t}(x) \leq \mathsf{K}_M^t(x) + c,$$

where for simplicity we write $t = t(n)$. It is well known that there exist time-optimal machines [29, Th. 7.1.1]. In this paper, we fix such a machine U, and drop the index U when referring to time-bounded Kolmogorov complexity measures. It is also possible to consider prefix-free notions of Kolmogorov complexity. However, since all our results hold up to additive $O(\log |x|)$ terms, we will not make an explicit distinction.

Henceforth we will not distinguish between a Turing machine $\mathcal{M}$ and its encoding p according to U. If p is a probabilistic Turing machine, we define $t_p \in \mathbb{N} \cup \{\infty\}$ to be the maximum number steps it takes p to halt on input λ (the empty string), where the maximum is over all branches of the probabilistic computation.

rKt complexity and probabilistic representations. A probabilistic representation of a string x is a probabilistic Turing machine p that on input λ halts with x on the output tape with probability at least $2/3$. The rKt-complexity of a string x is the minimum, over all probabilistic representations p of x, of $p + \log t_p$. A *probabilistic representation* p of x *certifies* rKt*-complexity bounded by* Γ if $|p| + \log t_p \leq \Gamma$.

Distributions and semi-distributions. We consider distributions over the set $\{0,1\}^*$. We will identify a distribution with its underlying probability density function of the form $\mathcal{D}\colon \{0,1\}^* \to [0,1]$. A distribution $\mathcal{D}$ is a *semi-distribution* if $\sum_{x \in \{0,1\}^*} \mathcal{D}(x) \leq 1$, and is simply called a *distribution* if the sum is exactly 1. In this subsection, we will use the word "distribution" to refer to both distribution and semi-distribution.

Samplers. A *sampler* is a probabilistic algorithm A with inputs in $\{1\}^n$ such that $A(1^n)$ outputs a string $x \in \{0,1\}^n$.[10] It defines a family of distributions $\{\mu_{A,n}\}_{n \in \mathbb{N}}$, where $\mu_{A,n}$ is the distribution on $\{0,1\}^n$ defined by $\mu_{A,n}(x) = \mathbf{Pr}_A[A(1^n) = x]$.

Average-case complexity. We now review some standard definitions and facts from average-case complexity. We refer to the survey [8] for more details.

▶ **Definition 7** (Polynomial-time Samplable [7])**.** *A distribution $\mathcal{D}$ is called* P*-samplable if there exists a polynomial p and a probabilistic algorithm M such that for every $x \in \{0,1\}^*$, M outputs x with probability $\mathcal{D}(x)$ within $p(|x|)$ steps.*

▶ **Definition 8** (Polynomial Time on Average [27])**.** *Let A be an algorithm and $\mathcal{D}$ be a distribution. We say that A runs in polynomial-time on average with respect to $\mathcal{D}$ if there exist constants ε and c such that,*

$$\sum_{x \in \{0,1\}^*} \frac{t_A(x)^\varepsilon}{|x|} \mathcal{D}(x) \leq c,$$

[10] For simplicity, we assume that $A(1^n)$ samples a string of length n. Our coding theorems also hold for algorithms used to define P-samplable distributions, see Definition 7, with obvious changes in the proofs. Also, as in [31], our results can be easily generalised to samplers that on 1^n output strings of arbitrary length. In this case, while the length of x might be significantly smaller than n, an additive overhead of $\log n + O(1)$ is necessary in our coding theorems, as we need to encode 1^n.

where $t_A(x)$ denotes the running time of A on input x. For a language L we say that L can be solved in polynomial time on average with respect to $\mathcal{D}$ if there is an algorithm that computes L and runs in polynomial-time on average with respect to $\mathcal{D}$.

▶ **Definition 9** (Domination). *Let $\mathcal{D}$ and $\mathcal{D}'$ be two distributions. We say that $\mathcal{D}$* dominates *$\mathcal{D}'$ if there is a constant $c > 0$ such that for every $x \in \{0,1\}^*$,*

$$\mathcal{D}(x) \geq \frac{\mathcal{D}'(x)}{|x|^c}.$$

▶ **Fact 10** (See e.g., [5, Lemma 3.3]). *Let $\mathcal{D}, \mathcal{D}'$ be two distributions, and let A be an algorithm. If*

- *A runs in polynomial time on average with respect to $\mathcal{D}$, and*
- *$\mathcal{D}$ dominates $\mathcal{D}'$*

Then A also runs in polynomial time on average with respect to $\mathcal{D}'$.