Electronic Colloquium on Computational Complexity, Report No. 82 (2021)

Hardness on Any Samplable Distribution Suffices: New Characterizations of One-Way Functions by Meta-Complexity

Rahul Ilango
MIT

Hanlin Ren
Tsinghua University

Rahul Santhanam
University of Oxford

June 16, 2021

Abstract

We show that one-way functions exist if and only if there is some samplable distribution $\mathcal{D}$ such that it is hard to approximate the Kolmogorov complexity of a string sampled from $\mathcal{D}$. Thus we characterize the existence of one-way functions by the average-case hardness of a natural *uncomputable* problem on samplable distributions, extending a recent line of work by Liu and Pass (FOCS'20, STOC'21) and Ren and Santhanam (CCC'21).

We also show that the average-case hardness of approximating Minimum Circuit Size on a locally samplable distribution (where the sampler runs in sub-linear time by using random access to its input) is equivalent to the existence of one-way functions. This is the first characterization of one-way functions by a natural average-case hardness assumption on the Minimum Circuit Size Problem. We present several other characterizations and connections between one-way functions and average-case hardness of meta-complexity problems (problems about complexity) on samplable distributions.

We give various applications of these results to the foundations of cryptography and the theory of meta-complexity. We show that the average-case hardness of deciding k-SAT or Clique on any samplable distribution of high enough entropy implies the existence of one-way functions. Thus one-way functions follow from general assumptions on the average-case hardness of NP-complete problems. We observe that our assumptions are implied by standard cryptographic assumptions such as the Planted Clique hypothesis and the pseudorandomness of Goldreich's local functions.

Our results imply a range of *equivalences* between various meta-complexity problems, showing that the theory of meta-complexity is very *robust* when considering average-case complexity. We use our results to unconditionally solve various meta-complexity problems in CZK (computational zero-knowledge) on average, and give implications of our results for the classic question of proving NP-hardness for the Minimum Circuit Size Problem.

1 Introduction

What is the most general complexity-theoretic assumption that implies the existence of one-way functions? This is a fundamental question in complexity theory and cryptography. The beautiful theory of cryptography that emerged in the 80s and 90s [BM84, Yao82, GM84, GGM86, HILL99, Gol01] uses one-way functions as a basic primitive. The relationship of this primitive to standard worst-case and average-case assumptions in complexity theory has yet to be properly understood. What is known is mostly in the form of *negative* results, indicating that we are unlikely to be able to base one-way functions on NP-hardness using "black-box" reductions [BT06, AGGM06, BB15].

For several well-studied computational problems, such as Factoring, Discrete Logarithm, the Shortest Vector Problem [AD97], Learning Parity with Noise [Ale11], Subset Sum [IN96] and Planted Clique [JP00], it is known that one-way functions (and in many of these cases, even public-key cryptography) can be based on hardness of the problem. However, it is not known for any of these problems whether a hardness assumption on the problem is *necessary*

ISSN 1433-8092

for the existence of one-way functions. In order to base cryptography on assumptions that are as *general* as possible, it would be helpful to have examples of natural computational problems whose hardness is *equivalent* to the existence of one-way functions.

If we relax the requirement that the computational problem needs to be natural, such examples are well-known. Levin [Lev03] showed the existence of a *universal* one-way function F: a fixed polynomial-time computable function that is one-way if one-way functions exist. It is straightforward to define a decision problem L_F corresponding to this universal one-way function such that L_F is average-case hard if and only if one-way functions exist.

We emphasize, however, that the naturalness of the computational problem is an important criterion when we are seeking to better understand the complexity-theoretic foundations of cryptography. As an analogy, consider the theory of NP-completeness. The canonical Bounded NTM Halting problem, which asks if a given non-deterministic Turing machine M halts on input x within t steps (for t given in unary), is easily seen to be NP-complete. What truly makes the notion of NP-completeness significant, though, is the work of Cook [Coo71], Levin [Lev73], and Karp [Kar72], showing that many natural problems (such as SAT, Clique, 3-Coloring, Integer Linear Programming, etc.) are NP-complete. This shows the deep relatedness of these seemingly different problems. The reductions among them can then be exploited in many different ways - in solving these problems in the real world, for example, or in understanding the approximability and average-case complexity of these problems.

One could ask, ambitiously: is there a similar Cook-Levin-Karp-style theory of completeness for one-way functions? Indeed, this question is explicitly posed by Levin in his survey [Lev03] on one-way functions.

In a recent breakthrough, Liu and Pass [LP20] showed that one-way functions exist if and only if the K^t problem (computing the t-time bounded Kolmogorov complexity[1]) is weakly hard[2] on average (in a bounded-error sense) over the uniform distribution for some polynomial t. The crucial point here is that unlike the problem L_F above, the problem K^t is *natural*: it is independently interesting and was heavily studied before in its own right [Tra84, Sip83, Har83, Ko86, Ko91, Hir18].

The Liu-Pass result suggests a deeper link between one-way functions and *meta-complexity*. Here "meta-complexity" refers to the complexity of a computational problem that is itself about complexity. The K^t problem is such an example, since determining the t-time bounded Kolmogorov complexity of a string is itself a problem about complexity. Other examples include the problem K of computing the Kolmogorov complexity of a string, and the Minimum Circuit Size Problem (MCSP) [KC00] of computing the minimum circuit size of a Boolean function given by its truth table. The theory of meta-complexity has found applications in several different contexts recently, including learning algorithms [CIKK16, OS17], worst-case to average-case reductions within NP and PH [Hir18, Hir20, Hir21], hardness magnification [OS18, OPS19, MMW19, CJW19, CJW20, CHO$^+$20, LP21a], pseudorandomness [Wil16, San20] and proof complexity [PS19].

Indeed, following [LP20], there have been several other results showing connections and equivalences between meta-complexity and one-way functions [LP21a, ACM$^+$21, RS21]. What these results all have in common is that they connect one-way functions to meta-complexity over the *uniform distribution*. While the uniform distribution is a natural one for meta-complexity problems, the theory of average-case complexity [Lev86] deals more generally with polynomial-time samplable distributions, and it is natural to ask if these connections and equivalences can be shown based on hardness over *samplable* distributions instead.

A further issue with considering hardness assumptions over the uniform distribution is that they are somewhat fragile in terms of parameters. For example, in [LP20], the assumption needs

[1]Given a universal Turing machine U, the t-time bounded Kolmogorov complexity of a string x is the size of the smallest program p such that $U(p)$ outputs x in at most $t(|x|)$ steps.

[2]We use "weakly hard" for the notion termed "mildly hard" in [LP20].

to be about weak average-case hardness rather than strong average-case hardness[3], even though one-wayness is robust in the sense that weak one-way functions exist if and only if strong one-way functions exist [Yao82]. Also, while the result in [LP20] does imply equivalence of computing K^t and $O(\log(n))$-additively approximating K^t, it is not capable of handling additive gaps that are $\omega(\log(n))$ (since K^t is in fact *not* weakly hard to $\omega(\log(n))$-additively approximate), or complexity thresholds much smaller than n (since only a negligible fraction of strings have K^t complexity significantly below n)[4].

In this paper, we characterize the existence of one-way functions by the average-case hardness of meta-complexity problems such as K and MCSP over *samplable* distributions. These characterizations are very robust, and lead to several applications in cryptography and meta-complexity.

1.1 Our Results

1.1.1 Equivalences

Our first main result shows an equivalence between the existence of one-way functions and the average-case hardness of a gap version of the problem of computing Kolmogorov complexity over samplable distributions. This might seem surprising at first, given that Kolmogorov complexity is *uncomputable*.

Below, $\mathsf{GapK}[s, c]$ denotes the promise problem of distinguishing between strings of Kolmogorov complexity at most s and Kolmogorov complexity at least c. We say that a problem is *weakly average-case hard* on a distribution $\mathcal{D}$ if every probabilistic polynomial-time algorithm fails to solve it with probability $1/n^{O(1)}$ (over $\mathcal{D}$ and the randomness of the algorithm) on almost all input lengths, and we say that a problem is *strongly average-case hard* on a distribution $\mathcal{D}$ if every probabilistic polynomial-time algorithm fails to solve it with probability $1/2 - 1/n^{\omega(1)}$ on almost all input lengths.

Theorem 1. *The following are equivalent:*

1. *One-way functions exist.*
2. *For some $s = n^{\Omega(1)}$ and $\Delta = \omega(\log(n))$, there is a samplable distribution $\mathcal{D}$ such that $\mathsf{GapK}[s, s + \Delta]$ is weakly average-case hard on $\mathcal{D}$.*
3. *For every $\epsilon > 0$, there is a samplable distribution $\mathcal{D}$ such that $\mathsf{GapK}[n^\epsilon, n - \omega(\log(n))]$ is strongly average-case hard on $\mathcal{D}$.*

Theorem 1 shows the *robustness* of approximating Kolmogorov complexity on average over samplable distributions, with respect to the complexity parameter s, the approximation gap Δ, and the notion of average-case hardness. If there is a samplable distribution on which approximating Kolmogorov complexity with some $\omega(\log(n))$ gap for some complexity parameter $s = n^{\Omega(1)}$ is weakly average-case hard, then for each $\epsilon > 0$ there is a samplable distribution on which approximating Kolmogorov complexity up to a multiplicative factor $n^{1-\epsilon}$ is strongly average-case hard.

The fact that the average-case hardness of approximating Kolmogorov complexity implies the existence of one-way functions might seem especially surprising, given that candidate one-way functions are usually defined based on problems in NP, while Kolmogorov complexity is uncomputable. While most constructions of one-way functions based on the average-case

[3]The result in [LP20] in fact shows that $O(\log(n))$-additively approximating K^t complexity is mildly hard on the uniform distribution if and only if one-way functions exist. K^t complexity can be $O(\log(n))$-additively approximated on all but an inverse polynomial fraction of the inputs on the uniform distribution simply by outputting n on every input, since most inputs have K^t complexity close to n.

[4]In a subsequent paper, [LP21a] address this issue, but at the cost of using a somewhat unnatural notion of average-case hardness (which they call "average-case* hardness").

hardness of some computational problem use the structure of the problem to define the one-way function and argue security based on the distributional average-case hardness, our construction does the reverse[5]: the one-way function is defined based on the distribution, while the proof of security exploits the structure of the problem assumed to be hard, i.e., Kolmogorov complexity.

By strengthening the samplability assumption in Theorem 1 and using the influential localization technique of [AIK06], we can also characterize one-way functions computable in NC^0.[6]

Theorem 2. *The following are equivalent:*

1. *There are one-way functions computable in* NC^0*.*
2. *For some* $s = n^{\Omega(1)}$ *and* $\Delta = \omega(\log(n))$*, there is a logspace-samplable distribution* $\mathcal{D}$ *such that* $\mathsf{GapK}[s, s + \Delta]$ *is weakly average-case hard on* $\mathcal{D}$*.*
3. *There is an* NC^0*-samplable distribution* $\mathcal{D}$ *such that* $\mathsf{GapK}[n - n^{0.99}, n - \omega(\log(n))]$ *is strongly average-case hard on* $\mathcal{D}$*.*

A natural question is whether there is a version of Theorem 1 or Theorem 2 where the equivalence involves the hardness of a meta-complexity problem known to be in NP, such as the K^t problem considered in [LP20][7] We are able to achieve this with a more complicated proof, but only under a complexity-theoretic derandomization assumption and for infinitely-often one-way functions. However, we get a much cleaner result for the Minimum Circuit Size Problem.

Let $\mathsf{GapMCSP}[s, c]$ denote the promise problem of distinguishing between truth tables of Boolean functions with circuit complexity at most s and Boolean functions with circuit complexity at least c. It is a major open problem to characterize one-way functions by the average-case hardness of MCSP. The celebrated Natural Proofs paper of Razborov and Rudich [RR97] shows that one-way functions imply the zero-error average-case hardness of MCSP, but no unconditional converse to this result is known[8].

We give the first equivalence between the average-case hardness of MCSP and the existence of one-way functions. In order to do this, we consider a samplability notion that we believe is interesting in its own right: *local samplability.* A t-local sampler is a sampler that, in order to compute a given bit of its output, runs in time t with random access to its input[9]. Here t is typically some sub-linear function. Several natural distributions, such as the uniform distribution and distributions induced by pseudorandom function generators [GGM86], are t-locally samplable for small t. Further motivating the notion is the fact that for most pairs L, L' of natural NP-complete problems, there is a *local* reduction from L to L', and hence the hardness of L with respect to some locally samplable distribution translates to the hardness of L' also with respect to some locally samplable distribution.

Theorem 3. *The following are equivalent:*

1. *One-way functions exist.*
2. *For some constant* $\delta > 0$ *and* $s = \Omega(n^{\delta})$*, there is an* (n^{δ})*-locally samplable distribution* $\mathcal{D}$ *such that* $\mathsf{GapMCSP}[s, sn^{5\delta}]$ *is weakly average-case hard on* $\mathcal{D}$*.*

[5]There are precedents for this, such as [IL90]. In [IL90], they prove that if NP is hard over some samplable distribution, then NP is also hard over the uniform distribution. One of the cases they consider is that the sampler itself already implements a one-way function; in this case, a hard NP language over the uniform distribution follows easily.

[6]NC^0 is the class of (multi-output) functions computable by uniform circuits such that each output bit is connected to a constant number of input bits.

[7]Intuitively K^t is an easier problem than K, therefore basing one-way functions on hardness of K^t should be *easier* than basing one-way functions on hardness of K. This is not the case, though, for the proof techniques we use.

[8]Such a converse is given in [San20] under an additional assumption about universal succinct pseudorandom distributions.

[9]Consequently, a t-local sampler can access at most t bits of its input while computing a specific output bit.

3. *For every constant $\delta > 0$, there is an (n^δ)-locally samplable distribution $\mathcal{D}$ such that* $\mathsf{GapMCSP}[n^\delta, o(\frac{n}{\log n})]$ *is strongly average-case hard on $\mathcal{D}$.*

One of the main research directions in the theory of meta-complexity is to show the robustness of meta-complexity problems with respect to the complexity parameter and the gap. Many results in the area are insensitive to differences in the complexity parameter and the gap, but there are few formal reductions between meta-complexity problems that justify this. Theorem 3 shows that in the setting of average-case complexity with respect to locally samplable distributions, MCSP is very robust with respect to the complexity parameter and the gap.

1.1.2 Applications

Our new characterizations of one-way functions by the hardness of meta-complexity problems on samplable distributions have several applications in cryptography and the theory of meta-complexity.

One-way functions from hardness of SAT and Clique. First, they allow us to show that one-way functions follow from the average-case hardness of NP-complete problems such as SAT and Clique under more general assumptions than were known before. Specifically, average-case hardness of SAT or Clique on *any* samplable distribution of *high enough entropy* implies the existence of one-way functions. Candidate one-way functions based on SAT and Clique are often based on very specific distributions, which lead to hardness assumptions that are not very robust. By showing that one-wayness can be derived from more general classes of distributions, we make progress towards basing one-way functions simply on the average-case hardness of NP.

Below, the entropy *deficiency* of a distribution on m bits is the difference between m and the entropy.

Theorem 4. *Given an integer k, let $\Delta \geq 2^{k+3}$ be a large enough integer, and let $t : \mathbb{N} \to \mathbb{N}$ be any function such that $t(n) = \omega(\log n)$. If*

- *k-SAT on Δn clauses is strongly average-case hard w.r.t. some samplable (resp. logspace-samplable) distribution $\mathcal{D}$ with entropy deficiency at most $\Delta n/2^{k+1}$, or*
- *t-Clique is strongly average-case hard w.r.t. some samplable (resp. logspace-samplable) distribution $\mathcal{D}$ with entropy deficiency at most $0.99\binom{t}{2}$,*

then one-way functions (resp. one-way functions computable in NC^0) exist.

It is natural to wonder if the hardness assumptions in Theorem 4 are reasonable. We show that in fact, the hardness assumption for Clique follows from the well-studied Planted Clique Hypothesis [Jer92, Kuc95, AKS98], while the hardness assumption for SAT follows from pseudorandomness of random local functions (often referred to as"Goldreich's PRG") [Gol00, App13, App16]. Thus our assumptions generalize hypotheses that have been intensively studied.

Unconditional CZK protocols for meta-complexity. Turning to the theory of meta-complexity, our characterizations imply *unconditional* average-case simulations of the corresponding meta-complexity problems in CZK (Computational Zero Knowledge) infinitely often. As far as we are aware, these are the first natural examples of approximation problems shown to be in CZK (on average) without also being shown to be in SZK (Statistical Zero Knowledge).

Below, we say that a problem is infinitely often in CZK on a distribution $\mathcal{D}$ if for each $k > 0$, there is a CZK protocol that is correct with probability at least $1 - 1/n^k$ on inputs sampled from $\mathcal{D}$, for infinitely many n.

Theorem 5. *For every $s : \mathbb{N} \to \mathbb{N}$ such that $s(n) = n^{\Omega(1)}$ and for every samplable distribution $\mathcal{D}$, $\mathsf{GapK}[s, s + \omega(\log(n))]$ is infinitely often in* CZK *on $\mathcal{D}$.*

For every $\delta > 0$, $s = \Omega(n^\delta)$, and (n^δ)-locally samplable distribution $\mathcal{D}$, $\mathsf{GapMCSP}[s, sn^{5\delta}]$ is infinitely often in CZK *on $\mathcal{D}$.*

Non-NP-hardness of GapMCSP under randomized local reductions. Finally, we use our results to shed some light on the long-standing open question of whether MCSP is NP-complete. Based on the assumption that one-way functions in NC^0 exist, we rule out NP-hardness of GapMCSP under randomized local reductions. To the best of our knowledge, this is the first piece of evidence against *randomized* reductions from SAT to GapMCSP. We note that Murray and Williams [MW17] unconditionally ruled out NP-hardness of MCSP under deterministic local reductions.

Theorem 6. *Suppose there are one-way functions computable in NC^0. Then for each $\delta > 0$ and $s = \Omega(n^\delta)$, there are no randomized (n^δ)-local reductions from* SAT *to $\mathsf{GapMCSP}[s, sn^{4\delta}]$.*

1.2 Techniques

Here we discuss the main ideas used in our proofs. We restrict ourselves to high-level arguments in this section, and do not delve too deeply into the choice of parameters.

1.2.1 Equivalences

The argument for one direction of Theorems 1 and 3 is straightforward given previous work. Suppose one-way functions exist, and we wish to show that GapK and GapMCSP are strongly hard on average. By [HILL99,GGM86], for each $\epsilon > 0$ there are pseudo-random generators with seed length n^ϵ computable in polynomial time such that each output of the generator, when interpreted as the truth table of a function, has circuit size $n^{O(\epsilon)}$. This also implies that every output of the generator has Kolmogorov complexity at most $n^{O(\epsilon)}$. On the other hand, a random string x has $\mathsf{K}(x)$ close to n and circuit size close to $n/\log(n)$ with high probability. Thus, we can consider the samplable distribution $\mathcal{D}$ that generates a uniformly random string with probability $1/2$ and a uniformly random output of the pseudo-random generator with probability $1/2$. Any algorithm for GapK or GapMCSP that has noticeable advantage over random could be used to distinguish the uniform distribution from the pseudo-random distribution, contradicting the pseudo-randomness assumption.

For the other direction of Theorem 1, suppose that there is a samplable distribution $\mathcal{D}$ such that GapK is hard on average with respect to $\mathcal{D}$. The crucial observation is the following:

- If a string y has a high probability of being sampled from $\mathcal{D}$, then the Kolmogorov complexity of y is small. The reason is that there are few strings sampled with high probability in $\mathcal{D}$, and we could simply specify y by giving its index in a lexicographically ordered list of high-probability strings.

- On the other hand, if a string x has a low probability of being sampled from $\mathcal{D}$, then the Kolmogorov complexity of y is large (on average over $\mathcal{D}$). To see this, note that by a simple counting argument there are few strings of small Kolmogorov complexity. Therefore the cumulative weight in $\mathcal{D}$ of such small Kolmogorov-complexity strings sampled with low probability is low.

Hence, in order to solve GapK on $\mathcal{D}$, it is sufficient to check whether a string is sampled with high or low probability on average over $\mathcal{D}$. Under the assumption that one-way functions don't exist, this can be done infinitely often in probabilistic polynomial time (on average) with standard hashing techniques.

We note that similar ideas have been used before in the literature on one-way functions and average-case complexity [IL89,IL90]. Our main contribution in proving Theorem 1 is conceptual. We use these ideas to illuminate the fundamental role of meta-complexity in this context.

In order to show Theorem 2, we need to adapt the proofs of both directions of Theorem 1. In order to get average-case hardness on logspace-samplable distributions from NC^0 one-way functions, we use [HRV13] rather than [HILL99]. In order to get NC^0 one-way functions from average-case hardness on logspace-samplable distributions, we observe that since there exist logspace-computable hash functions, when starting from a logspace-samplable distribution, the proof of Theorem 1 also gives logspace-computable one-way functions. We then apply the main result of [AIK06] to get one-way functions in NC^0.

To show Theorem 3, we adopt the same template as in Theorem 1, but need to work harder and use some new ideas. Suppose that we have hardness of $\mathsf{GapMCSP}$ on some locally samplable distribution $\mathcal{D}$, and wish to derive one-way functions from this hardness assumption. When we were analyzing Kolmogorov complexity, the basic observation that a string y sampled from $\mathcal{D}$ with high probability has low complexity was easy to show. The corresponding result is harder when we work with circuit complexity.

We show that if y is the truth table of a function with high circuit complexity, then p_y is low, where p_y is the probability that y is sampled from $\mathcal{D}$. The key idea is to "reveal" bits of the input to the sampler used to compute y in stages. Each stage reveals a small number of bits of the input to the sampler, by its locality. If after a small number of stages, all bits of y can be correctly computed by an approximate majority over random choices of the unrevealed bits, then we can argue that we get a small circuit for y. Suppose this is not the case. Then there is some bit of y for which random choices to the unrevealed bits give the wrong answer with probability $\geq 1/3$. In this case, we can argue that p_y must decrease by a factor of $2/3$. If y has large circuit complexity, the number of stages in this process must be high, and hence p_y must be low.

Once we show this claim, we can use the non-existence of one-way functions to approximate p_y on average as in the proof of Theorem 1.

Remark 7. The first bullet in our discussion of GapK (i.e. any string sampled from an (efficient) sampler with noticeable probability has low Kolmogorov complexity) is called a *coding theorem* in Kolmogorov complexity. Our results for $\mathsf{GapMCSP}$ can also be interpreted as a coding theorem for circuit complexity (and KT complexity), but over *locally-samplable* distributions. We remark that Lu and Oliveira [LO21] recently showed a coding theorem for rKt (a randomized version of Levin's Kt complexity [Lev84, Oli19]).

1.2.2 Applications

Theorem 4: One-way functions from hardness of SAT and Clique. Our proof of Theorem 4 is inspired by a zero-error average-case reduction from SAT to computing KT complexity[10] in [HS17]. The idea is that random k-CNF formulas are incompressible, while k-CNFs with satisfying assignments can be compressed if they are long enough. A similar idea gives a zero-error reduction from Clique to computing KT.

Here we adapt these ideas to the bounded-error average-case setting. When considering bounded-error average-case complexity, it is no longer the case that the uniform distribution is a reasonable one to consider for k-SAT, since answering "Unsatisfiable" works with overwhelmingly high probability. However, it is still reasonable to expect average-case hardness on distributions with high entropy. We show that if the distribution has high enough entropy, then there are bounded-error reductions from k-SAT and Clique to approximating Kolmogorov complexity. The reductions themselves are the simplest possible, namely the identity reduction! However, the proof that they work requires the compressibility argument from [HS17] as well

[10] KT complexity is a meta-complexity notion defined in [All01] that is closely related to circuit complexity.

as the fact that high entropy distributions must place noticeable probability on strings of high Kolmogorov complexity. We thereby get a reduction from computing SAT or Clique on average with noticeable advantage over random on a samplable distribution $\mathcal{D}$ with high enough entropy to computing GapK with all but inverse polynomial probability on $\mathcal{D}$. The robustness of GapK on average is crucial to our argument, as the reduction needs the algorithm for GapK to be correct w.p. $1 - 1/\text{poly}(n)$.

To show that our average-case assumptions are reasonable, we show that they are implied by well-studied hardness assumptions in average-case complexity and cryptography, namely the Planted Clique Hypothesis for Clique and the pseudorandomness of random local functions for k-SAT.

Theorem 5: Unconditional CZK protocols for meta-complexity. The proof of Theorem 5 uses a win-win argument:

- It is well known that if one-way functions exist, then CZK = IP = PSPACE [BGG+88, Sha92]. In this case, since p_y can be computed in polynomial space for any string y sampled from the distribution $\mathcal{D}$, we have that GapK is in CZK on average.
- Suppose, on the other hand, that one-way functions don't exist. Then by Theorem 1, GapK is infinitely often in probabilistic polynomial time on $\mathcal{D}$. Since CZK trivially contains probabilistic polynomial time, GapK is infinitely often in CZK on $\mathcal{D}$ in this case as well.

A similar argument works for GapMCSP on locally samplable distributions, using Theorem 3 instead of Theorem 1.

Theorem 6: Non-NP-hardness of GapMCSP under randomized local reductions. Finally, to prove Theorem 6, we first show that if a language L has randomized local reductions to GapMCSP, then L is easy on average over locally samplable distributions. The main ingredient of this proof is showing that GapMCSP can be is easy on average over a locally samplable distribution when given the randomness of the sampler, rather than just its output. This argument is similar to the argument that p_y is low for strings y of high circuit complexity sampled by a local sampler $\mathcal{D}$. We observe that under the assumption that there are one-way functions in NC^0, k-SAT is average-case hard on some locally samplable distribution, and combining this with the lemma about randomized local reductions concludes the proof.

1.3 Related Work

There have been several works relating one-way functions to non-cryptographic notions. Impagliazzo and Levin [IL90] show that one-way functions exist if and only if "universal extrapolation" does not, where universal extrapolation is a generic procedure to sample from continuations of the output of some samplable process. Some of the ideas we use are similar to theirs, though there does not seem to be a formal connection between the results. Blum, Furst, Kearns, and Lipton [BFKL93] relate the existence of one-way functions to an average-case notion of learning. Oliveira and Santhanam [OS17] show that exponentially hard (non-uniform) one-way functions exist if and only if non-trivial (non-uniform) learning is hard.

More recently, there have been a number of works considering the average-case hardness of meta-complexity problems on the uniform distribution and relating it to one-way functions. Santhanam [San20] showed that under an assumption on universal succinct pseudorandom distributions, MCSP is zero-error hard on average on the uniform distribution if and only if one-way functions exist. By considering K^t rather than MCSP and bounded-error hardness rather than zero-error hardness, Liu and Pass [LP20] gave an unconditional equivalence. Characterizations of NC^0 cryptography by meta-complexity over the uniform distribution are given in [LP21c, RS21]. An implication for one-way functions from the average-case hardness of the conditional KT-complexity problem is given in [ACM+21]. [LP21b] give a natural NP-complete problem whose average-case hardness on the uniform distribution is equivalent to the existence of one-way functions.

2 Preliminaries

For a positive integer n, we let $[n] = \{1, \ldots, n\}$. Throughout this paper, we assume $s, \Delta : \mathbb{N} \to \mathbb{N}$ are polynomial-time computable functions. If S is a set, we write $x \leftarrow S$ to denote sampling an element of S uniformly at random.

An n-bit *partial string* is a $\rho \in \{0, 1, \star\}^n$. We say a (total string) $x \in \{0,1\}^n$ *agrees* with ρ if the i'th bit of x equals the i'th bit of ρ whenever $\rho(i)$ is Boolean. We write $y \leftarrow \rho$ to denote sampling a uniformly random binary string $y \in \{0,1\}^n$ that agrees with ρ.

A PPT algorithm is a probabilistic polynomial time algorithm. We let $\mathsf{negl}(n)$ denote a negligible function (i.e. $\mathsf{negl}(n) = 1/n^{\omega(1)}$).

2.1 Samplable Distributions

For each positive integer n, let D_n be a probability distribution on $\{0,1\}^n$. We say $\mathcal{D} = \{D_n\}$ is an *ensemble.* We say that an ensemble $\mathcal{D}$ is a *samplable distribution* if there is a polynomial-time sampling algorithm Samp such that $\mathsf{Samp}(1^n, r)$ samples from D_n when r is a uniformly random string of length $m = \text{poly}(n)$. For any Boolean string $y \in \{0,1\}^n$, we let p_y denote the probability that y is sampled from D_n (the underlying samplable distribution $\mathcal{D}$ is implicit and will always be clear from context).

We say that a probabilistic algorithm $\mathcal{A}$ computes a language L on $\mathcal{D}$ in time $t(n)$ with error $\delta(n)$ for all n in some set $N \subseteq \mathbb{N}$ if for all $n \in N$

$$\Pr_{x \leftarrow D_n, \mathcal{A}}[\mathcal{A}(x) \neq L(x)] \leq \delta(n)$$

and $\mathcal{A}(x)$ runs in time $t(n)$ for all $x \in \{0,1\}^n$. If this holds with $N = \mathbb{N}$, we omit specifying a subset N, and if this holds for an infinite set N, we say $\mathcal{A}$ computes L on $\mathcal{D}$ in time $t(n)$ with error $\delta(n)$ *infinitely-often.*

We say an ensemble $\mathcal{D} = \{D_n\}$ is a $(t(n))$-*locally samplable distribution* if there is a local sampling algorithm LSamp that gets random access to three inputs (a length parameter n in binary[11], an index $i \in [n]$, and some randomness $r \in \{0,1\}^{\text{poly}(n)}$) and has the following property. For any $i \in [n]$, if $r \in \{0,1\}^{\text{poly}(n)}$ is chosen uniformly at random, then $\mathsf{LSamp}(n, i, r)$ outputs the i'th bit of a sample from D_n in time $t(n)$. In particular, the n-bit string given by

$$\mathsf{LSamp}(n, 1, r) \ldots \mathsf{LSamp}(n, n, r)$$

is a sample from D_n if r is chosen uniformly at random.

We will abuse notation and let $\mathsf{LSamp}(n, r)$ with no i provided as input denote the n-bit string:

$$\mathsf{LSamp}(n, r) = \mathsf{LSamp}(n, 1, r) \ldots \mathsf{LSamp}(n, n, r).$$

2.2 One-Way Functions

Throughout this paper, we consider one-way functions secure against *uniform* probabilistic adversaries. More precisely, we say a (deterministic) polynomial-time computable function

[11]The reason why n is provided to LSamp in binary rather than unary is because we will want LSamp to run in $t(n)$ time and $t(n)$ is usually less than n.

$f : \{0,1\}^n \to \{0,1\}^n$ is *one-way* if every probabilistic polynomial-time algorithm I succeeds at finding pre-image of f with negligible probability. That is, for all n,

$$\Pr_{x \leftarrow \{0,1\}^n}[f(I(f(x))) = f(x)] \le \mathsf{negl}(n).$$

(Note that we assumed f is *length-preserving*, i.e. for every x, $|f(x)| = |x|$. This is without loss of generality, see e.g. [Gol01, Proposition 2.2.5].)

We say f is *one-way infinitely often* if for every probabilistic polynomial-time algorithm I, there exists an infinite set N such that for all $n \in N$,

$$\Pr_{x \leftarrow \{0,1\}^n}[f(I(f(x))) = f(x)] \le \mathsf{negl}(n).$$

Finally, we say a polynomial-time computable function $f : \{0,1\}^n \to \{0,1\}$ is *weakly one-way* if there is a polynomial p such that for every probabilistic polynomial-time algorithm I and every n,

$$\Pr_{x \leftarrow \{0,1\}^n}[f(I(f(x))) = f(x)] \le 1 - 1/p(n).$$

Yao [Yao82] showed that weak one-way functions exist if and only if one-way functions exist. As a result, if one-way functions do not exist, then for every polynomial computable function $f : \{0,1\}^n \to \{0,1\}^n$ and every constant $q \ge 1$, there exists a randomized polynomial-time algorithm I such that for infinitely many n

$$\Pr_{x \leftarrow \{0,1\}^n}[f(I(f(x))) = f(x)] \ge 1 - O(1/n^{-q}).$$

Remark 8. In this paper we assume one-way functions are secure against *uniform* adversaries. Most of our results should also hold for non-uniform adversaries, but there is a potential counter-example, namely Theorem 28. The reason is that we assume there are no one-way functions and invert two candidate one-way functions f_1, f_2 in the proof. The function f_2 *depends on the inverter of* f_1, so if the inverter of f_1 is non-uniform, then f_2 itself is only computable with non-uniformity, and it is unclear whether we can still invert f_2.

2.3 Circuits

Throughout this paper, we work with *DeMorgan circuits* (i.e. circuits with fan-in two AND and OR gates and NOT gates), although this choice is not crucial to our results. The *size* of a circuit C, denoted $|C|$, is the number of AND and OR gates in the circuit. Given a truth table T of a Boolean function, the *circuit complexity* of T, denoted $\mathsf{CC}(T)$, is the size of the smallest circuit computing T.

We will also make use of *randomized circuits*. A randomized circuit D is a circuit C that has two inputs, a (regular) input $x \in \{0,1\}^n$ and a probabilistic input $r \in \{0,1\}^m$. We say that a randomized circuit computes D computes a Boolean function f if for all $x \in \{0,1\}^n$, the probability $C(x,r) = f(x)$ is at least $2/3$ when r is chosen uniformly at random. The size of D is the size of C.

We will also use Adleman's construction for converting a randomized circuit into a deterministic circuit.

Theorem 9 (Adleman [Adl78]). *Suppose D is a randomized circuit of size s that computes $f : \{0,1\}^n \to \{0,1\}$. Then there is a (deterministic) circuit C of size at most $s \cdot \mathrm{poly}(n)$ that computes f.*

2.4 Kolmogorov Complexity and Its Meta-Complexity

We introduce the variants of Kolmogorov complexity that we consider.

Definition 10. Fix a universal Turing machine U. For a string $x \in \{0,1\}^\star$:

- The *Kolmogorov complexity* of x, denoted as $\mathsf{K}_U(x)$, is the length of the shortest program d such that $U(d)$ outputs x in finite time.
- Let $t(\cdot)$ be a function. The *t-time-bounded Kolmogorov complexity* of x, denoted as $\mathsf{K}^t_U(x)$, is the length of the shortest program d such that $U(d)$ outputs x in at most $t(|x|)$ steps [Ko91].
- The KT-*complexity* of x, denoted as $\mathsf{KT}_U(x)$, is the minimum of $|d| + t$ over all programs d and time bounds t such that for every $1 \le i \le |x| + 1$, $U^d(i) = x_i$ [All01]. (Here we assume $x_{|x|+1} = \star$. Also note that U is given *random* access to the program d.)

Our results hold for every efficient enough universal Turing machine U, therefore we will omit the subscript U and simply write $\mathsf{K}(x)$, $\mathsf{K}^t(x)$, and $\mathsf{KT}(x)$.

We need the standard fact that the number of strings with low Kolmogorov complexity is small (which is proved by a counting argument):

Fact 11. *Let s, n be integers. The number of strings $x \in \{0,1\}^n$ such that $\mathsf{K}(x) \le s$ is at most $\sum_{i=0}^{s} 2^i = 2^{s+1} - 1$.*

We also define the meta-complexity problems associated with each complexity measure (K, K^t, KT, and circuit complexity).

Definition 12. Let $0 < s_1(n) < s_2(n) < n$ be two functions.

- $\mathsf{GapK}[s_1, s_2]$ is the promise problem whose YES instances are strings x such that $\mathsf{K}(x) \le s_1(|x|)$, and NO instances are strings x such that $\mathsf{K}(x) \ge s_2(|x|)$.
- Let t be a polynomial. $\mathsf{GapK}^t[s_1, s_2]$ is the promise problem whose YES instances are strings x such that $\mathsf{K}^t(x) \le s_1(|x|)$, and NO instances are strings x such that $\mathsf{K}^t(x) \ge s_2(|x|)$.
- $\mathsf{GapKT}[s_1, s_2]$ is the promise problem whose YES instances are strings x such that $\mathsf{KT}(x) \le s_1(|x|)$, and NO instances are strings x such that $\mathsf{KT}(x) \ge s_2(|x|)$.

Let $0 < s_1(n) < s_2(n) < n/\log n$ be two functions.

- $\mathsf{GapMCSP}[s_1, s_2]$ is the promise problem whose YES instances are truth tables tt with circuit complexity at most $s_1(|tt|)$, and NO instances are truth tables tt with circuit complexity at least $s_2(|tt|)$.

2.5 Hash Functions

We will make use of the existence of explicit, pairwise independent hash functions.

Theorem 13 (See e.g. [Vad12, Problem 3.3])**.** *Let $m \le n \in \mathbb{N}$. There is a family of pairwise independent hash functions $\mathcal{H}_{n,m}$ where each element of $\mathcal{H}_{n,m}$ is a function $h_w : \{0,1\}^n \to \{0,1\}^m$ indexed by a $w \in \{0,1\}^{n+m}$. Moreover, given n, m, w, and $x \in \{0,1\}^n$, one compute $h_w(x)$ in time $\mathrm{poly}(n)$ and space $O(\log n)$.*

The Leftover Hash Lemma, first stated by Impagliazzo, Levin, and Luby [ILL89], will be crucial for us.

Lemma 14 (Leftover Hash Lemma)**.** *Let $X \in \{0,1\}^n$ be a random variable with min-entropy at least k. Let $H_{n,m} = \{h_w : w \in \{0,1\}^{n+m}\}$ be a family of pairwise independent hash functions from n bits to m bits. Then the statistical distance between the following two distributions is at most $\sqrt{2^{m-k}}$:*

- *sample $w \leftarrow \{0,1\}^{n+m}$, $x \leftarrow X$, and output $(w, h_w(x))$;*
- *sample $w \leftarrow \{0,1\}^{n+m}$, $v \leftarrow \{0,1\}^m$, and output (w, v).*

3 Average-Case Algorithms for Variants of Kolmogorov Complexity

In this section, we give polynomial-time algorithms for that solving various versions of Kolmogorov complexity on average, under some assumptions (such as the non-existence of one-way functions).

3.1 Algorithms for GapK

3.1.1 An Unconditional Exponential-Time Algorithm

While GapK is uncomputable in the *worst-case*, things are different in the *average-case* setting with two-sided error. For example, if one wants to compute $\mathsf{GapK}[n-\Delta, n]$ on the uniform distribution, there is a very simple algorithm: just always output NO. This algorithm errs with probability at most $2^{-\Delta+1}$ since the number of strings with K-complexity at most $n-\Delta$ is at most $2^{n-\Delta+1}$ by Fact 11.

A natural question is whether one can give an average-case algorithm for computing GapK on *any* polynomial-time samplable distribution. It turns out that (if Δ is large enough) one can give an average-case algorithm that runs in exponential-time unconditionally.

Recall that throughout this paper $s, \Delta : \mathbb{N} \to \mathbb{N}$ denote polynomial-time computable functions.

Theorem 15. *Let $\mathcal{D} = \{D_n\}$ be a samplable distribution and $\Delta = \omega(\log n)$. Then there is a deterministic algorithm $\mathcal{A}$ running in time $2^{\mathrm{poly}(n)}$ that solves $\mathsf{GapK}[s-\Delta, s]$ on $\mathcal{D}$ with error at most $2^{-\Delta/3}$.*

We actually prove a stronger version of Theorem 15 that will be useful later. Recall p_y denotes the probability that y is sampled from D_n, where $y \in \{0,1\}^n$.

Theorem 16. *Let $\mathcal{D} = \{D_n\}$ be a samplable distribution and $\Delta = \omega(\log n)$. Let $\mathcal{O}$ be an oracle, $c \geq 1$ be an arbitrary constant, and $0 < \epsilon < 1$ be such that*

$$\Pr_{y \leftarrow D_n}[p_y/c < \mathcal{O}(y) \leq p_y] \geq 1-\epsilon$$

for all n in some set $N \subseteq \mathbb{N}$.

Then given oracle access to $\mathcal{O}$, one can solve $\mathsf{GapK}[s-\Delta, s]$ on $\mathcal{D}$ in deterministic[12] polynomial time with error at most $\epsilon + 2^{-\Delta/3}$ for all $n \in N$.

Assuming Theorem 16, we can prove Theorem 15.

Proof of Theorem 15. For any samplable distribution $\mathcal{D}$, there is an exponential time algorithm computing p_y exactly (in the worst-case) given $y \in \{0,1\}^n$: we simply compute

$$\sum_{r \in \{0,1\}^m} \mathbb{1}_{\mathsf{Samp}(1^n, r) = y}$$

where $m = \mathrm{poly}(n)$. Therefore, Theorem 15 follows from Theorem 16 with $c = 1$ and $\epsilon = 0$. $\square$

[12] If $\mathcal{O}$ is randomized, then the algorithm will be randomized.

We now prove Theorem 16.

Proof of Theorem 16. The algorithm $\mathcal{A}$ is very simple. Given $y \in \{0,1\}^n$, $\mathcal{A}$ outputs YES if $\mathcal{O}(y)$ is large, in particular greater than

$$\alpha = 2^{-s+\Delta/2},$$

and outputs NO otherwise. This completes our description of the algorithm $\mathcal{A}$.

Clearly this algorithm runs in polynomial time (with oracle access to $\mathcal{O}$), so we just need to argue for correctness when n is sufficiently large. The key to proving the correctness of this algorithm are the following two claims. The first claim says (informally) that low complexity y rarely have $p_y < c\alpha$.

Claim 17. *If n is sufficiently large, then*

$$\Pr_{y \leftarrow D_n}[\mathsf{K}(y) \leq s - \Delta \textit{ and } p_y < c\alpha] \leq 2^{-\Delta/3}.$$

The second claim says (informally) that high complexity strings y always have $p_y \leq \alpha$.

Claim 18. *If n is sufficiently large, then*

$$\Pr_{y \leftarrow D_n}[\mathsf{K}(y) \geq s \textit{ and } p_y \geq \alpha] = 0.$$

Combining Claim 17 and Claim 18 and union bounding over the ϵ error probability of $\mathcal{O}$, we get that for all sufficiently large $n \in N$ that

$$\Pr_{y \leftarrow D_n}[\mathcal{A}(y) \text{ errs in computing } \mathsf{GapK}[s-\Delta, s]] \leq \epsilon + 2^{-\Delta/3}$$

which proves the theorem. It remains to prove the two claims, which we do below.

Proof of Claim 17. By Fact 11, the number of strings with K-complexity at most $s - \Delta$ is at most $2^{s-\Delta+1}$. On the other hand, the probability that y is output by D_n is p_y, by definition. Thus, by a union bound, we get

$$\Pr_{y \leftarrow D_n}[\mathsf{K}(y) \leq s - \Delta \text{ and } p_y < \alpha \cdot c] \leq 2^{s-\Delta+1} \cdot \alpha \cdot c \leq 2c \cdot 2^{-\Delta/2} \leq 2^{-\Delta/3}$$

when n is sufficiently large since $c = O(1)$ and $\Delta = \omega(\log n)$. ◇

Proof of Claim 18. Observe that the set of strings $H = \{y \in \{0,1\}^n : p_y \geq \alpha\}$ has size at most $1/\alpha + 1$, since the p_y quantities are non-negative and together sum to one. Thus, we can describe any element y of H by specifying the code for Samp, n, α and the index of y in H. Therefore, we get that the Kolmogorov complexity of any element of H is at most

$$O(\log n) + \log(1/\alpha) + O(1) \leq O(\log n) + s - \Delta/2 + O(1) \leq s - \omega(\log n).$$

Thus, if n is sufficiently large, no string with K-complexity at least s can be an element of H, as desired. ◇

□

3.1.2 A Conditional Polynomial-Time Algorithm

Theorem 15 gives an *exponential-time* algorithm for solving GapK on average. In this subsection, we show that if one-way functions do not exist, then one can get a randomized *polynomial-time* algorithm for solving GapK on average infinitely often.

We begin by showing that p_y can be efficiently approximated on average (infinitely often) if one-way functions do not exist.

Theorem 19 ([IL90,IL89]). *Assume no one-way functions exist. Let $\mathcal{D} = \{D_n\}$ be a samplable distribution and let $q \geq 1$ be an arbitrary constant. Then there exists a randomized polynomial-time algorithm $\mathcal{A}$ such that for infinitely many n,*

$$\Pr_{y \leftarrow D_n}[\Omega(p_y) \leq \mathcal{A}(y) \leq p_y] \geq 1 - O(1/n^q).$$

Proof. Let Samp be the sampling algorithm for $\mathcal{D}$. Let $m = \mathrm{poly}(n)$ be the number of random bits the sampler Samp uses to sample from D_n. Let $H_{m,k} = \{h_w : w \in \{0,1\}^{m+k}\}$ denote the family of efficient pairwise independent hash functions mapping m-bits to k-bits given by Theorem 13.

Consider the polynomial-time computable function f that given an input (n, k, w, r), where $k \in [m]$, $w \in \{0,1\}^{m+k}$, and $r \in \{0,1\}^m$, acts as follows. Set $y = \mathsf{Samp}(1^n, r)$. Select the hash function h_w from $\mathcal{H}_{m,k}$ and compute $h_w(r)$. Finally, output $(n, y, k, w, h_w(r))$.

Now, since one-way functions do not exist (and therefore weak one-way functions do not exist), there exists a randomized polynomial-time algorithm I such that for all n in an infinite set N we have

$$\Pr_{\substack{k \leftarrow [m],\\ w \leftarrow \{0,1\}^{m+k},\\ r \leftarrow \{0,1\}^m,\\ z = f(n,k,w,r)}}[f(I(z)) = z] \geq 1 - 1/(mn^q).$$

We will use I to estimate p_y by seeing "how often there is an r that Samp maps to y in some random hash bucket." In more detail, we define a $\{0,1\}$-valued random variable $E(k, y)$ for each $k \in [m]$ and $y \in \{0,1\}^n$ as follows (E stands for "experiment"). $E(k, y)$ is the indicator random variable for the event that I succeeds at finding a pre-image for (n, y, k, w, v) when $v \leftarrow \{0,1\}^k$ and $w \leftarrow \{0,1\}^{m+k}$.

Our randomized polynomial-time algorithm $\mathcal{A}$ works as follows. Given y, the algorithm repeats the experiment $E(k, y)$ m^2 times independently for each $k \in [m]$. It then outputs $2^{k'-m}$ where k' is the smallest value of k such that the experiment succeeded (i.e. output 1) for at most half of the repetitions.

It is easy to see that this algorithm runs in randomized polynomial-time. We now argue for correctness. Let $n \in N$. We will show that the expectation of $E(k, y)$ is small (at most 1/4) when $2^{k-m} \geq 4p_y$ and usually (with probability $1 - O(1/n^q)$ over $y \leftarrow D_n$) large (at least 3/4) when $2^{k-m} \leq p_y/64$. The correctness of $\mathcal{A}$ will then follow from a Chernoff bound.[13]

First, we show that the expectation of $E(k, y)$ is small if p_y is small relative to k. Fix y and k. The total number of $r \in \{0,1\}^m$ that map to y under Samp is exactly $p_y 2^m$ (by definition of p_y). Therefore, for any fixed w, the total number of $v \in \{0,1\}^k$ for which there exists an $r \in \{0,1\}^m$ such that $y = \mathsf{Samp}(1^n, r)$ and $h_w(r) = v$ is at most $p_y 2^m$. Thus, $E(k, y)$ can succeed with probability at most $p_y 2^{m-k}$. So if $p_y \leq 2^{k-m}/4$, then the expectation of $E(k, y)$ is at most 1/4.

Next, we show that the expectation of $E(k, y)$ is usually large if p_y is large relative to k. Fix some $k \in [m]$, and fix some $y \in \{0,1\}^n$. Let R_y be the set of r such that $\mathsf{Samp}(1^n, r) = y$. Again note that by definition of p_y we have $|R_y| = p_y 2^m$. The leftover hash lemma (Lemma 14) implies that the following two distributions have statistical distance at most $\sqrt{2^{k-m}/p_y}$

[13]To get the guarantee stated in the theorem that $\Omega(p_y) \leq \mathcal{A}(y) \leq p_y$, one needs to multiply the output of $\mathcal{A}$ by an appropriate constant.

- sample $w \leftarrow \{0,1\}^{m+k}$ and $v \leftarrow \{0,1\}^k$ and output (n, y, k, w, v), and
- sample $w \leftarrow \{0,1\}^{m+k}$, $r \leftarrow R_y$, and $v \leftarrow \{0,1\}^k$ and output $(n, y, k, w, h_w(r))$.

Thus, we have that

$$\Pr[E(k,y)=0] \le \sqrt{2^{k-m}/p_y} + \Pr_{r\leftarrow R_y, w\leftarrow\{0,1\}^{m+k}}[I(n,y,k,w,h_w(r)) \text{ fails to invert}]$$

We now show that this quantity is small for most y. Say that y is *good* if for all $k \in [m]$

$$\Pr_{r\leftarrow R_y, w\leftarrow\{0,1\}^{m+k}}[I(n,y,k,w,h_w(r)) \text{ fails to invert}] \le 1/8.$$

If y is good, then for all k satisfying $2^{k-m} \ge p_y/64$, we have that

$$\begin{aligned}\Pr[E(k,y)=0] &\le \sqrt{2^{k-m}/p_y} + \Pr_{r\leftarrow R_y, w\leftarrow\{0,1\}^{m+k}}[I(n,y,k,w,h_w(r)) \text{ fails to invert }]\\ &\le 1/8 + 1/8\\ &= 1/4,\end{aligned}$$

so the expectation of $E(k,y)$ is at least $3/4$. Finally, because the failure probability of I is at most $1/(mn^q)$, by union bounding over all $k \in [m]$, we know that a y sampled from D_n is good with probability $1 - O(1/n^q)$. □

We also note that the same proof shows a stronger consequence if we assume infinitely-often one-way functions do not exist, and that the proof produces an $\mathcal{A}$ that gives an estimate on p_y with one-sided error. (This is because the upper bound of $p_y 2^{m-k}$ on the expectation $E(k,y)$ holds for all y.) These modifications will be useful later in Section 3.2.

Theorem 20. *Assume no infinitely-often one-way functions exist. Let $\mathcal{D} = \{D_n\}$ be a samplable distribution and let $q \ge 1$ be an arbitrary constant. Then there exists a randomized polynomial-time algorithm $\mathcal{A}$ such that for all n*

$$\Pr_{y\leftarrow D_n}[\mathcal{A}(y) = \Omega(p_y)] \ge 1 - O(1/n^q),$$

and for all $y \in \{0,1\}^n$

$$\Pr[\mathcal{A}(y) \le p_y] \ge 1 - O(1/n^q).$$

Combining Theorem 19 and Theorem 16, we get the following corollary.

Corollary 21. *Assume one-way functions do not exist. Let $\mathcal{D}$ be a samplable distribution, $\Delta = \omega(\log n)$, and $q \ge 1$ be an arbitrary constant. Then there is a randomized polynomial-time algorithm $\mathcal{A}$ that infinitely-often solves $\mathsf{GapK}[s-\Delta, s]$ on $\mathcal{D}$ with error at most $O(n^{-q})$.*

3.1.3 Algorithms for Logspace-Samplable Distributions

Recall that [AIK06] showed that the existence of one-way functions computable in NC^0 is equivalent to the existence of one-way functions computable in logspace. To characterize NC^0-computable one-way functions, we prove the following variant of Theorem 19:

Theorem 22. *Assume that no NC^0-computable one-way functions exist. Let $\mathcal{D} = \{D_n\}$ be a logspace-samplable distribution and let $q \ge 1$ be an arbitrary constant. Then there exists a randomized polynomial-time algorithm $\mathcal{A}$ such that for infinitely many n,*

$$\Pr_{y\leftarrow D_n}[\mathcal{A}(y) \ge \Omega(p_y)] \ge 1 - O(1/n^q)$$

and for all $y \in \{0,1\}^\star$,

$$\Pr[\mathcal{A}(y) \le p_y] \ge 1 - O(1/|y|^q).$$

Proof Sketch. The theorem essentially follows from the proof of Theorem 19. Let Samp be the logspace sampling algorithm for $\mathcal{D}$. Let $H_{m,k} = \{h_w : w \in \{0,1\}^{m+k}\}$ denote the family of efficient pairwise independent hash functions given by Theorem 13. Recall that the candidate one-way function f (that we will invert) is defined as follows: on input (n, k, w, r), set $y = \mathsf{Samp}(1^n, r)$ and output $(n, y, k, w, h_w(r))$. As both Samp and the hash function are computable in logarithmic space, it follows that f is also computable in logarithmic space. Since logspace-computable one-way functions imply NC^0-computable one-way functions, which does not exist by our assumption, we know that f is not one-way. As in the proof of Theorem 19, we can use the inverter I for f to estimate p_y. (Note that $\mathcal{A}(y) \le p_y$ holds w.h.p. for *every* string y regardless of how the inverter I behaves.) □

Combining Theorem 22 and Theorem 16, we obtain the following corollary:

Corollary 23. *Assume that no NC^0-computable one-way functions exist. Let $\mathcal{D}$ be a logspace-samplable distribution, $q \ge 1$ be an arbitrary constant, and $\Delta = \omega(\log n)$. Then there is a randomized polynomial-time algorithm that infinitely-often solves $\mathsf{GapK}[s - \Delta, s]$ over $\mathcal{D}$ with error at most $O(n^{-q})$.*

3.2 An Algorithm for GapK^t

When moving from GapK to GapK^t the main thing that breaks is Claim 18, which shows that high complexity strings must have low values of p_y. Unfortunately, we do not know how to get around this *unconditionally*. Luckily, assuming *infinitely-often*[14] *one-way functions do not exist* and a derandomization assumption, we can prove an analogous claim.

Various derandomization assumptions suffice for our purpose but we will use [IW97] which shows that if $\mathsf{E} \not\subseteq \mathsf{ioSIZE}[2^{.01n}]$, then there are complexity-theoretic pseudorandom generators secure against $\mathsf{P/Poly}$ with seed length $O(\log n)$ computable in time $\mathrm{poly}(n)$. More precisely, if $\mathsf{E} \not\subseteq \mathsf{ioSIZE}[2^{.01n}]$, then for every polynomial p and every constant $0 < \epsilon < 1$, there exists a multi-set $G_{n,p,\epsilon} \subseteq \{0,1\}^n$ such that

- $G_{n,p,\epsilon}$ has $\mathrm{poly}(n)$ elements,
- the i'th element of $G_{n,p,\epsilon}$ is computable in deterministic time $\mathrm{poly}(n)$, and
- if C is a circuit that takes n-inputs and has size at most $p(n)$, then

$$\left| \Pr_{r \leftarrow \{0,1\}^n}[C(r) = 1] - \Pr_{r \leftarrow G_{n,p,\epsilon}}[C(r) = 1] \right| \le \epsilon.$$

Lemma 24. *Assume infinitely-often one-way functions do not exist and that $\mathsf{E} \not\subseteq \mathsf{ioSIZE}[2^{.01n}]$. Let $\mathcal{D} = \{D_n\}$ be a samplable distribution, $q \ge 1$, and $\Delta = \omega(\log n)$. Let $\alpha = 2^{-s+\Delta/2}$. Then for every large enough polynomial τ and for all n,*

$$\Pr_{y \leftarrow D_n}[\mathsf{K}^\tau(y) \ge s \text{ and } p_y \ge \alpha] \le O(n^{-q}).$$

Proof. At a high level, the idea is this: suppose y is a high complexity string and p_y is large. To achieve a contradiction, we want to come up with a small, efficient description of y. We will do this by specifying a small hash v of y such that this hash is unique among the (not too many) strings z such that p_z is large. We will then use the non-existence of one-way functions (in two different ways) to show that given v, one can recover y (on average). The full details are below.

[14] At a high level, the reason why we assume that *infinitely often* one-way functions do not exist (instead of just the non-existence of one-way functions) is that we use the ability to invert candidate one-way functions twice, in a recursive manner. As a result, we need to be careful to make sure the input lengths where the two inverters succeed match.

Let k be the ceiling of $s - \Delta/3$. We can assume that $k \leq n$ (since otherwise $s = n + \omega(\log n)$ and the statement is trivial). Let $\mathcal{H}_{n,k} = \{h_w : w \in \{0,1\}^{n+k}\}$ be the family of explicit pairwise independent hash functions mapping n bits to k bits given by Theorem 13. Let $m = \mathrm{poly}(n)$ be the number of random bits used by Samp to sample from D_n. Since infinitely often one-way functions do not exist, Theorem 20 implies there is a randomized polynomial time algorithm $\mathcal{A}$ which computes a c-factor approximation of p_y with error at most $O(n^{-2q})$ over $\mathcal{D}$. We can assume that $\mathcal{A}$ is deterministic using our derandomization assumption. Again using our derandomization assumption, let $\mathsf{HitSet} = G_{n+k,p,\epsilon} \subseteq \{0,1\}^{n+k}$ where p is some fixed sufficiently large polynomial and $\epsilon = 1/3$. Let τ be some sufficiently large polynomial.

Consider the polynomial-time computable function f that given $n \in \mathbb{N}, r \in \{0,1\}^m$, and $w \in \{0,1\}^{n+k}$ works as follows. Let $y = \mathsf{Samp}(1^n, r)$. If $\mathcal{A}(y) < \alpha/c$, then f outputs $\perp$. Otherwise, f outputs $(n, w, h_w(y))$ where $h_w \in \mathcal{H}_{n,k}$. This completes the description of f.

Since infinitely-often one-way functions do not exist, there exists a deterministic[15] polynomial time algorithm I such that for all n

$$\Pr_{r \leftarrow \{0,1\}^m, w \leftarrow \{0,1\}^{n+k}, y = \mathsf{Samp}(1^n, r)}[\mathcal{A}(y) < \alpha/c \text{ or } h_w(I(n, w, h_w(y))) = h_w(y)] \geq 1 - 1/n^{4q}.$$

Now fix some sufficiently large n and let Bad be the set of all $y \in \{0,1\}^n$ satisfying $\mathsf{K}^\tau(y) \geq s$ and $p_y \geq \alpha$. For contradiction, assume that

$$\Pr_{y \leftarrow D_n}[y \in \mathit{Bad}] > n^{-q}.$$

We make the following claim.

Claim 25. *If n is sufficiently large, there exists a $y \in \mathit{Bad}$, a $w \in \mathsf{HitSet}$, and a $v \in \{0,1\}^k$ such that $I(n, w, v)$ outputs y.*

If Claim 25 is true, then this yields an efficient description of y. In particular, if τ is a sufficiently large polynomial

$$\mathsf{K}^\tau(y) \leq \log n + \log |\mathsf{HitSet}| + k + O(1) \leq k + O(\log(n + k)) \leq s - \omega(\log n),$$

which contradicts the s lower bound on the K^τ complexity of y when n is sufficiently large. This proves the theorem (assuming Claim 25).

It remains to prove Claim 25.

Proof of Claim 25. We say that $w \in \{0,1\}^{n+k}$ is *nice* if

$$\Pr_{y \leftarrow D_n}[I(n, w, h_w(y)) \neq y \mid \mathcal{A}(y) \geq \alpha/c] \leq 4n^{-2q}.$$

We also say w is *almost nice* if the RHS is replaced by $5n^{-2q}$, that is

$$\Pr_{y \leftarrow D_n}[I(n, w, h_w(y)) \neq y \mid \mathcal{A}(y) \geq \alpha/c] \leq 5n^{-2q}.$$

We note that the event being conditioned on above occurs somewhat often, that is

$$\Pr_{y \leftarrow D_n}[\mathcal{A}(y) \geq \alpha/c] \geq \Pr_{y \leftarrow D_n}[y \in \mathit{Bad}] - O(n^{-2q}) = \Omega(n^{-q}),$$

using the fact that $\mathcal{A}$ gives a c-approximation of p_y with error probability $O(n^{-2q})$.

We make another claim.

[15] We can assume this algorithm is deterministic because of our derandomization assumption and the fact we can efficiently check whether the output of the inverter is indeed a pre-image of f.

Claim 26. *Assume n is sufficiently large. If $w \leftarrow \{0,1\}^{n+k}$, then w is nice with probability at least $1/2$.*

Assuming Claim 26 is true, there must exist an element w of HitSet that is almost nice. This is because given w, one can estimate $\Pr_{y \leftarrow D_n}[I(n, w, h_w(y)) \neq y \mid \mathcal{A}(y) \geq \alpha/c]$ up to an arbitrarily small constant factor in polynomial time by sampling from D_n repeatedly and outputting the empirical frequency of the event that $I(n, w, h_w(y)) \neq y$ when $\mathcal{A}(y) \geq \alpha/c$. (Recall the event that $\mathcal{A}(y) \geq \alpha/c$ occurs with probability at least $\Omega(n^{-q})$, so a polynomial number of samples suffice.) Using our derandomization assumption, there is a deterministic polynomial-time algorithm that accepts at least $1/2$ fraction of w's (namely, the nice ones) and rejects every w that is not almost nice. Since p is a sufficiently large polynomial, the security of the pseudorandom generator $\mathsf{HitSet} = G_{n+k,p,1/3}$ implies the existence of an almost nice $w \in \mathsf{HitSet}$.

Now fix an almost nice $w \in \mathsf{HitSet}$. Since every element $y \in \mathit{Bad}$ satisfies that $p_y \geq \alpha$ (by construction) and since $\mathcal{A}$ has failure probability $O(n^{-2q})$, we know that

$$\begin{aligned}
\Pr_{y \leftarrow D_n}[y \in \mathit{Bad} \mid \mathcal{A}(y) \geq \alpha/c] &\geq \Pr_{y \leftarrow D_n}[y \in \mathit{Bad} \text{ and } \mathcal{A}(y) \geq \alpha/c] \\
&\geq \Pr_{y \leftarrow D_n}[y \in \mathit{Bad}] - O(n^{-2q}) \\
&\geq \Omega(n^{-q})
\end{aligned}$$

Combining this bound on $\Pr_{y \leftarrow D_n}[y \in \mathit{Bad} \mid \mathcal{A}(y) \geq \alpha/c]$ with the fact that w is almost nice, we get via a union bound that

$$\Pr_{y \leftarrow D_n}[y \notin \mathit{Bad} \text{ or } I(n, w, h_w(y)) \neq y \mid \mathcal{A}(y) \geq \alpha/c] \leq 1 - \Omega(n^{-q}) + O(n^{-2q}),$$

which is less than 1 when n is sufficiently large. Therefore, there must exist a $y \in \mathit{Bad}$ such that $I(n, w, h_w(y)) = y$ when n is sufficiently large. $\diamond$

Finally, we prove Claim 26.

Proof of Claim 26. When n is sufficiently large we have that

$$\begin{aligned}
&\Pr_{y \leftarrow D_n, w \leftarrow \{0,1\}^{n+k}}[I(n, w, h_w(y)) \neq y \mid \mathcal{A}(y) \geq \alpha/c] \\
\leq\, & O(n^{-3q}) + \Pr_{y \leftarrow D_n, w \leftarrow \{0,1\}^{n+k}}[\exists y' \in \{0,1\}^n \setminus \{y\} \text{ with } \mathcal{A}(y') \geq \alpha/c \text{ and } h_w(y) = h_w(y') \mid \mathcal{A}(y) \geq \alpha/c] \\
\leq\, & n^{-2q} + (c/\alpha + 1)2^{-k} \\
\leq\, & n^{-2q} + 2^{s-\Delta/2+O(1)} 2^{\Delta/3-s-1} \\
\leq\, & 2n^{-2q},
\end{aligned}$$

where the first inequality comes from the n^{-4q} bound on the failure probability of I and the fact that $\Pr_{y \leftarrow D_n}[\mathcal{A}(y) \geq \alpha/c] = \Omega(n^{-q})$, the second inequality comes from two-wise independence and the fact that the number of y with $\mathcal{A}(y) \geq \alpha$ is at most[16] the number of y with $p_y \geq \alpha$ which is at most $1/\alpha + 1$.

Therefore, by Markov's inequality, for all least half of the $w \in \{0,1\}^{n+k}$ we have that

$$\Pr_{y \leftarrow D_n}[I(n, w, h_w(y)) \neq y \mid \mathcal{A}(y) \geq \alpha/c] \leq 4n^{-2q},$$

and so w is nice. $\diamond$

$\square$

[16]This uses that $\mathcal{A}$ has one-sided error and never overestimates p_y.

The analog of Claim 17 also holds for GapK^t.

Proposition 27. *Let τ be any polynomial, let $c \geq 1$ be an arbitrary constant, and let $\Delta = \omega(\log n)$. Let $\alpha = 2^{-s+\Delta/2}$. Then when n is sufficiently large*

$$\Pr_{y \leftarrow D_n}[\mathsf{K}^\tau(y) \leq s - \Delta \text{ and } p_y < c\alpha] \leq 2^{-\Delta/3}.$$

Proof. The proof is similar to Claim 17. Union bound over the $2^{s-\Delta}$ low complexity strings that each show up with probability $c\alpha$. This gives a bound of

$$c\alpha 2^{s-\Delta} = c2^{-\Delta/2} = o(2^{-\Delta/3}). \qquad \square$$

Combining Proposition 27, Lemma 24, and Theorem 20, we get the following theorem.

Theorem 28. *Assume infinitely-often one-way functions do not exist and $\mathsf{E} \not\subseteq \mathsf{ioSIZE}[2^{.01n}]$. Let $\mathcal{D}$ be a samplable distribution, let $q \geq 1$ be an arbitrary constant, and let $\Delta = \omega(\log n)$. Then there exists a polynomial τ and randomized polynomial-time algorithm $\mathcal{A}$ such that $\mathcal{A}$ solves $\mathsf{GapK}^\tau[s - \Delta, s]$ on $\mathcal{D}$ with error probability at most $O(n^{-q})$.*

3.3 An Algorithm for MCSP

When moving from GapK to MCSP, again the part of the proof that breaks is Claim 18, showing that high complexity strings must have low values of p_y. We show that the analogous claim holds for locally samplable distributions (with certain parameters).

Throughout this section, let $0 < \delta < 1$, and let LSamp be the local sampling algorithm for a (n^δ)-locally samplable distribution $\mathcal{D} = \{D_n\}$. Assume LSamp takes as input a random seed of length $m = \mathrm{poly}(n)$ to sample from D_n.

Before we prove the analogous claim, we prove a useful lemma. Recall our notation that if $\rho \in \{0, 1, *\}^m$, then $z \leftarrow \rho$ denotes sampling z uniformly at random from the set of strings in $\{0, 1\}^m$ that agree ρ.

Lemma 29. *Let $\rho \in \{0, 1, *\}^m$ be a string with at most d Boolean values. If $y \in \{0, 1\}^n$ requires circuits of size at least $dn^{3\delta}$ and n is sufficiently large, then there exists an $i \in [n]$ such that*

$$\Pr_{\tilde{r} \leftarrow \rho}[\mathsf{LSamp}(1^n, i, \tilde{r}) \neq y(i)] \geq 1/3.$$

Proof. For contradiction, assume that for all $i \in [n]$ that

$$\Pr_{\tilde{r} \leftarrow \rho}[\mathsf{LSamp}(1^n, i, \tilde{r}) = y(i)] \geq 2/3.$$

We will use this to construct a small circuit C for y and get a contradiction.

To begin, we construct a small *randomized* circuit C' for computing y. C' will take as input an index $i \in [n]$, will use n^δ bits of randomness, and will have size $O(dn^{2\delta})$.

The circuit C' on input i will work as follows. It will simulate running the local sampling algorithm LSamp on inputs 1^n, i, and randomness $\tilde{r}$, where $\tilde{r}$ is constructed "on the fly" via the following method (recall, LSamp is a local sampling algorithm that makes n^δ oracle queries to $\tilde{r}$). Whenever LSamp makes an oracle query to the j'th bit of $\tilde{r}$, respond as follows. If LSamp has previously queried the j'th bit of $\tilde{r}$, then give the same answer as before. Otherwise, if $\rho(j) \in \{0, 1\}$, then answer with $\rho(j)$; if $\rho(j) = \star$, answer with a (fresh) uniformly random bit. This completes our description of C'.

Observe that for all $i \in [n]$

$$\Pr[C'(i) = y(i)] = \Pr_{\tilde{r} \leftarrow \rho}[\mathsf{LSamp}(1^n, i, \tilde{r}) = y(i)] \geq 2/3,$$

so the randomized circuit C' computes y. This is because our method for answering oracle queries to $\tilde{r}$ is equivalent to picking a uniformly random $\tilde{r}$ that agrees with ρ and answering according to that.

Now we bound the size of C'. One can convert the oracle algorithm LSamp running in time n^δ into an oracle circuit D of size $O(n^\delta \mathrm{poly}\log n)$. To build C' from D, one just needs to replace each of the at most n^δ oracle gates in D with a circuit that can query a bit of the $\tilde{r}$ we are building "on the fly." Checking if an index of $\tilde{r}$ was previously queried and repeating that answer if so can be done with $O(n^\delta \log n)$ gates, checking if ρ is defined on an index and responding with that value if so can be done with $O(d \cdot \mathrm{poly}\log n)$ gates (since ρ has a Boolean value on d indices), and answering with a fresh random bit can be done with $O(1)$ gates. Thus, each oracle gate can be replaced with a circuit of size at most $O(dn^\delta \mathrm{poly}\log n)$. Thus C' has size at most $O(dn^{2\delta} \mathrm{poly}\log n)$.

Finally, we can convert the randomized circuit C' computing y into a deterministic circuit C computing y of size $O(dn^{2\delta} \mathrm{poly}\log n)$ by using Adleman's construction (Theorem 9). Note that the input $i \in [n]$ to the circuit C' has length $O(\log n)$ as a bit string, which is why the blowup is only a multiplicative $\mathrm{poly}\log n$. $\square$

We can now show that high complexity strings have small values of p_y.

Lemma 30. *Let $y \in \{0,1\}^n$ and $s = \mathsf{CC}(y)$. Then if n is sufficiently large*

$$p_y \leq (2/3)^{s/n^{3\delta}}.$$

Proof. We will show that the set of all r such that $\mathsf{LSamp}(1^n, r) = y$ is small. We will do this by constructing a sequence of sets Rest_k that will contain m-bit restrictions (i.e. elements of $\{0,1,*\}^m$) and will satisfy the following two properties for all k:

1. If $r \in \{0,1\}^m$ and $\mathsf{LSamp}(1^n, r) = y$, then r agrees with exactly one element of Rest_k.
2. The number of 0/1-values in ρ is at most $k \cdot n^\delta$ for every $\rho \in \mathit{Rest}_k$.

We set Rest_0 to just contain the empty restriction, that is,

$$\mathit{Rest}_0 = \{*^m\}.$$

Observe that Rest_0 satisfies properties (1) and (2).

We will construct the remaining Rest_k inductively. To do this, we will need to introduce some notation. Given a restriction $\rho \in \{0,1,*\}^m$, let $i(\rho)$ denote the (lexicographically first) value of $i \in [n]$ that minimizes the quantity

$$p_i(\rho) = \Pr_{\tilde{r} \leftarrow \rho}[\mathsf{LSamp}(1^n, i, \tilde{r}) = y(i)].$$

Let $p(\rho)$ denote $p_{i(\rho)}(\rho)$.

Given a $\tilde{r} \in \{0,1\}^m$ and an $i \in [n]$, we say the *query sequence of i for $\tilde{r}$* is the sequence $(j_1, v_1), \ldots, (j_t, v_t) \in [m] \times \{0,1\}$ of oracle queries and responses that LSamp makes and receives on input $(1^n, i, \tilde{r})$. That is, LSamp first queries the j_1-th bit of $\tilde{r}$, obtains that $\tilde{r}_{j_1} = v_1$, then queries the j_2-th bit of $\tilde{r}$ and obtains v_2, and so on. Note that $t \leq n^\delta$ since LSamp runs in time n^δ.

Given a restriction $\rho \in \{0,1,\star\}^m$, we say a *consistent query sequence of i for ρ* is a query sequence of i for some $\tilde{r}$ that agrees with ρ. Note that there are only 2^{n^δ} consistent query sequences of i for ρ since LSamp makes at most n^δ queries and each query has at most two possible outcomes.

If S is a consistent query sequence for i and ρ, then S and ρ determine a refinement $\rho'[\rho, S]$ of ρ as follows. Set $\rho'[\rho, S] = \rho$ and then further set $\rho'[\rho, S](j_k) = v_k$ for all $k \in [t]$, where $S = ((j_1, v_1), \ldots, (j_t, v_t))$.

Next, we define $Next(\rho)$. Given a restriction ρ, let $Next(\rho)$ be the set of all $\rho'[\rho, S]$ where S is a consistent query sequence of $i(\rho)$ for ρ.

We can now finish our inductive construction. We set $Rest_{k+1} = \bigcup_{\rho \in Rest_k} Next(\rho)$. Observe that $Rest_{k+1}$ will satisfy properties (1) and (2) assuming that $Rest_k$ does.

Now, we will bound the number of strings that agree with an element of $Rest_{n^\delta}$. First, observe that the total number of strings that agree with an element of $Next(\rho)$ is at most $p(\rho)$ times the number of strings that agree with ρ. (This is essentially by definition.)

Next, assume n is sufficiently large so that we can use Lemma 29. Then if $\rho \in Rest_k$ and $k \leq s/n^{3\delta}$, then combining (2) and Lemma 29, we get that $p(\rho) \leq 2/3$, and thus that the number of strings that agree with an element of $Next(\rho)$ is at most 2/3rds the number of strings that agree with ρ.

Thus, using (1) we get that the number of strings that agree with an element of $Rest_{k+1}$ is at most 2/3rds the number of strings that agree with an element of $Rest_k$. Applying this repeatedly, we get that the fraction of m-bit strings that agree with $Rest_{n^\delta}$ is at most

$$(2/3)^{s/n^{3\delta}}. \qquad \square$$

On the other hand, a union bound shows that low complexity strings rarely have small p_y values.

Proposition 31. *Let $c \geq 1$ be an arbitrary constant and $s = \Omega(n^{5\delta})$. Then*

$$\Pr_{y \leftarrow D_n}[\mathsf{CC}(y) \leq s/n^{4\delta} \text{ and } p_y < (2/3)^{s/n^{3\delta}}] = \mathsf{negl}(n).$$

Proof. The number of strings with circuit complexity at most $s/n^{4\delta}$ is at most $O(2^{s/n^{3.5\delta}})$. By a union bound, we have that

$$\Pr_{y \leftarrow D_n}[\mathsf{CC}(y) \leq s - \Delta \text{ and } p_y < \alpha \cdot c] \leq O(2^{s/n^{3.5\delta}}(2/3)^{s/n^{3\delta}}) = \mathsf{negl}(n). \qquad \square$$

Putting it all together, we get an average-case algorithm for approximating $\mathsf{GapMCSP}$ if one-way functions do not exist.

Theorem 32. *Assume one-way functions do not exist. Let $0 < \delta < 1$ and $q \geq 1$. Let $\mathcal{D}$ be a (n^δ)-locally samplable distribution. Let $s = \Omega(n^\delta)$. Then there exists a randomized polynomial-time algorithm $\mathcal{A}$ that infinitely-often solves $\mathsf{GapMCSP}[s, sn^{5\delta}]$ on $\mathcal{D}$ with error $O(n^{-q})$.*

Proof. Combine Proposition 31, Lemma 30, and Theorem 19. $\square$

3.3.1 An Algorithm for GapKT

In this subsection, we prove analogous results on one-way functions computable in NC^0. Here, we consider the KT complexity instead of the circuit complexity. Also, instead of (n^δ)-locally samplable distributions, here we consider $O(1)$-locally samplable distributions (i.e. those samplable in NC^0) that are closely related to the existence of one-way functions in NC^0. In particular, we consider t-locally samplable distributions in this section for some fixed constant t.

It is without loss of generality to assume that the sampler is non-adaptive, since any adaptive algorithm with query complexity $t = O(1)$ can be simulated by a non-adaptive algorithm with query complexity $2^t = O(1)$.

We first prove an analog of Lemma 29.

Lemma 33. *Let C be a large enough absolute constant. Let $\rho \in \{0, 1, *\}^m$ be a string with at most d Boolean values. If n is sufficiently large, $y \in \{0, 1\}^n$ satisfies that $\mathsf{KT}(y) > C(d + 2^t) \log m$, then there exists an $i \in [n]$ such that*

$$\Pr_{\tilde{r} \leftarrow \rho}[\mathsf{LSamp}(1^n, i, \tilde{r}) \neq y(i)] \geq 1/2.$$

Proof Sketch. Suppose not. Then we have the following upper bound for $\mathsf{KT}(y)$.

First, we use perfect hashing [FKS82] to preprocess ρ into a dictionary Dict_ρ of bit-length $O(d \log m)$ such that, given an index $i \in [m]$, we can query ρ_i in $O(\log m)$ time. Now, given an index i, we know that y_i is the majority of $\mathsf{LSamp}(1^n, i, \tilde{r})$ over all $\tilde{r}$ consistent with ρ. Since LSamp is a t-local sampler, $\mathsf{LSamp}(1^n, i, \tilde{r})$ only queries t bits of $\tilde{r}$, and we can compute this majority value in $O(2^t \cdot \log m)$ time. It follows that $\mathsf{KT}(y) \le |\mathsf{Dict}_\rho| + O(2^t \cdot \log m) \le O((d + 2^t) \log m)$. □

Now we prove an analog of Lemma 30.

Theorem 34. *Let $y \in \{0,1\}^n$ and $s = \mathsf{KT}(y)$. For some constant $\epsilon > 0$ (that depends on t), if n is sufficiently large, then*

$$p_y \le 2^{-\epsilon s/\log m}.$$

Proof Sketch. We follow the proof of Lemma 30. In particular, we consider the same construction of $Rest_k$ for $0 \le k \le k_0$ where $k_0 = \epsilon s/\log m$. Note that every restriction in $Rest_k$ has at most $k \cdot t$ 0/1 elements, and $C(k \cdot t + 2^t) \log m \le s$. By Lemma 33, for every $k \le k_0$, the number of random strings $r \in \{0,1\}^m$ that agrees with some element in $Rest_k$ is at most half of the number of $r \in \{0,1\}^m$ that agrees with some element in $Rest_{k-1}$. It follows that there are at most 2^{m-k_0} random strings $r \in \{0,1\}^m$ that agrees with some element in $Rest_{k_0}$. Since only such random strings r may satisfy that $\mathsf{LSamp}(1^n, r) = y$, we have

$$p_y \le 2^{-k_0} = 2^{-\epsilon s/\log m}.$$

□

Theorem 35. *Suppose that there are no one-way functions computable in NC^0. Let $q \ge 1$ be a constant, $\mathcal{D}$ be a t-locally samplable distribution, and $s(n) = \omega(\log^2 n)$ be an efficiently computable function. Then there exists a randomized PPT algorithm that infinitely-often solves $\mathsf{GapKT}[o(s/\log n), s]$ on $\mathcal{D}$ with error $O(1/n^q)$.*

Proof. Let $\mathcal{D}$ be a t-locally samplable distribution. By Theorem 22, there exists a randomized polynomial-time algorithm $\mathcal{A}$ and an infinite set of "good" input lengths $N \subseteq \mathbb{N}$ such that for every $n \in N$,

$$\Pr_{y \leftarrow D_n}[\mathcal{A}(y) \ge \Omega(p_y)] \ge 1 - O(1/n^q),$$

and for all $y \in \{0,1\}^n$,

$$\Pr[\mathcal{A}(y) \le p_y] \ge 1 - O(1/n^q).$$

Let $\alpha = 2^{-\epsilon s(n)/\log m}$. On input y, we output YES if $\mathcal{A}(y) \ge \alpha$, and output NO otherwise. Fix a "good" input length $n \in N$. For every $y \in \{0,1\}^n$ such that $\mathsf{KT}(y) \ge s(n)$, w.p. $1 - O(1/n^q)$ we have $\mathcal{A}(y) \le p_y \le \alpha$. On the other hand, the probability over a random input $y \leftarrow D_n$ that $\mathsf{KT}(y) \le o(s/\log n)$ and our algorithm errs on y is at most

$$O(\alpha) \cdot 2^{o(s/\log n)} \le \mathsf{negl}(n).$$

It follows that the error probability of our algorithm is at most $O(1/n^q)$ over $\mathcal{D}$. □

4 New Characterizations of One-Way Functions

In this section, we characterize the existence of one-way functions by the average-case hardness of Kolmogorov complexity and their variants. It turns out that our characterizations are extremely *robust*: under samplable distributions, mildly average-case hardness (i.e. $1 - 1/\mathrm{poly}(n)$) of GapK with very small gap (e.g. $\mathsf{GapK}[\sqrt{n}, \sqrt{n} + \log^2 n]$) is equivalent to strong average-case hardness (i.e. $1/2 + \mathsf{negl}(n)$) of GapK with very large gap (e.g. $\mathsf{GapK}[n^{0.01}, n - \log^2 n]$), and they are all equivalent to the existence of one-way functions.

Section 3 already presents average-case algorithms for GapK under the assumption that one-way functions do not exist. Therefore, to complete the equivalence, we need to construct hard samplable distributions for GapK from the existence of one-way functions. This is done in Section 4.1 using the known constructions of PRGs and PRFs from one-way functions [HILL99, GGM86, HRV13]. Then in Section 4.2, we present our new characterizations of one-way functions.

4.1 Hardness of Kolmogorov Complexity from One-Way Functions

We begin by showing that computing GapK and GapK^τ are hard on average if one-way functions exist.

Theorem 36. *Assume one-way functions exist. Let* $0 < \epsilon < 1$. *Then there exists a samplable distribution* $\mathcal{D}$ *such that no PPT algorithm infinitely-often solves any of the following problems with probability greater than* $1/2 + \mathsf{negl}(n)$[17]*:*

- $\mathsf{GapK}[n^\epsilon, n - \omega(\log n)]$
- $\mathsf{GapK}^\tau[n^\epsilon, n - \omega(\log n)]$ *for every large enough polynomial* τ.

Proof. Suppose one-way functions exist, then pseudorandom generators also exist [HILL99]. Also, we may assume that the stretch of the PRG is an arbitrarily large polynomial (see e.g. [Gol01, Theorem 3.3.4]).

Let $\ell = n^{\epsilon/2}$, $G : \{0,1\}^\ell \to \{0,1\}^n$ be a secure PRG. Let U_n be the uniform distribution over $\{0,1\}^n$, and $G(U_\ell)$ be the output of G on a uniformly random seed. Let

$$\mathcal{D} = \frac{1}{2}(U_n + G(U_\ell)),$$

i.e. to sample from $\mathcal{D}$, w.p. $1/2$ we sample from U_n and w.p. $1/2$ we sample from $G(U_\ell)$.

Suppose there is an algorithm $\mathcal{A}$ that infinitely-often solves $\mathsf{GapK}[n^\epsilon, n - \omega(\log n)]$ over $\mathcal{D}$ w.p. $\geq 1/2 + 1/p(n)$, where p is a polynomial. Note that for every large enough n and every seed $s \in \{0,1\}^\ell$, we have $\mathsf{K}(G(s)) \leq \ell + O(1) \leq n^\epsilon$. On the other hand, w.p. $\leq \mathsf{negl}(n)$, a random string of length n has Kolmogorov complexity at most $n - \omega(\log n)$. Therefore the correctness of $\mathcal{A}$ implies

$$\frac{1}{2}\left(\Pr_{s\leftarrow U_\ell}[\mathcal{A}(G(s)) = 1] + (1 - \Pr_{x\leftarrow U_n}[\mathcal{A}(x) = 1]) + \mathsf{negl}(n)\right) \geq 1/2 + 1/p(n).$$

That is,

$$\left|\Pr_{s\leftarrow U_\ell}[\mathcal{A}(G(s)) = 1] - \Pr_{x\leftarrow U_n}[\mathcal{A}(x) = 1]\right| \geq 1/q(n),$$

for some polynomial q. This contradicts the security of G. Therefore, no PPT algorithm infinitely-often solves $\mathsf{GapK}[n^\epsilon, n - \omega(\log n)]$ over $\mathcal{D}$ with probability greater than $1/2 + \mathsf{negl}(n)$.

Let τ be any large enough polynomial. Note that:

- For every large enough n and every seed $s \in \{0,1\}^\ell$, we also have $\mathsf{K}^\tau(G(s)) \leq \ell + O(1) \leq n^\epsilon$.
- W.p. $\leq \mathsf{negl}(n)$, a random string of length n has K^τ complexity at most $n - \omega(\log n)$.

Thus the above argument also shows that $\mathsf{GapK}^\tau[n^\epsilon, n - \omega(\log n)]$ is average-case hard on $\mathcal{D}$. $\square$

Next, we show that $\mathsf{GapMCSP}$ is hard on average if one-way functions exist.

[17] Here, "with probability greater than $1/2 + \mathsf{negl}(n)$" means "with probability greater than $1/2 + 1/p(n)$ for some polynomial p".

Theorem 37. *Assume one-way functions exist. For every constant $0 < \delta < 1$, there is an (n^δ)-locally samplable distribution $\mathcal{D}$ such that no PPT algorithm infinitely-often solves* $\mathsf{GapMCSP}[n^\delta, o(\frac{n}{\log n})]$ *with probability greater than* $1/2 + \mathsf{negl}(n)$.

Proof. Suppose one-way functions exist, then pseudorandom function generators (PRFs) also exist [GGM86, HILL99].

More precisely, let $f : \{0,1\}^n \to \{0,1\}^n$ be a one-way function, and $k = (C/\delta)\log n$ where C is a large enough absolute constant. Let $N = 2^k$, then we can construct a PRF $F : \{0,1\}^n \to \{0,1\}^N$ according to [GGM86, HILL99] that satisfies the following properties:

1. For every $x \in \{0,1\}^n$, the circuit complexity of $F(x)$ (viewed as a truth table) is at most $n^C \le N^\delta$.
2. Any PPT algorithm $\mathcal{A}$ could distinguish $F(U_n)$ from U_N with advantage at most $\mathsf{negl}(n)$.

Consider the distribution
$$\mathcal{D} = \frac{1}{2}(U_N + F(U_n)).$$

First, it is easy to see that $\mathcal{D}$ is (N^δ)-local: it is $O(1)$-local to sample from U_N, and it is n-local to sample from $F(U_n)$. Since $n \le N^\delta$, it is (N^δ)-local to sample from $\mathcal{D}$.

Any output from $F(U_n)$ is a YES instance of $\mathsf{GapMCSP}[N^\delta, o(\frac{N}{\log N})]$. On the other hand, a random truth table of length N is a NO instance of $\mathsf{GapMCSP}[N^\delta, o(\frac{N}{\log N})]$ w.p. $\ge 1 - \mathsf{negl}(N)$. It follows that no PPT algorithm solves $\mathsf{GapMCSP}[N^\delta, o(\frac{N}{\log N})]$ over $\mathcal{D}$ with probability greater than $1/2 + \mathsf{negl}(N)$. □

We also can show hardness under the assumption that there are one-way functions computable in NC^0.

Theorem 38. *Assume there exist one-way functions in* NC^0. *Then there exists an $O(1)$-locally samplable distribution $\mathcal{D}$ such that no PPT algorithm infinitely-often solves any of the following problems with probability greater than* $1/2 + \mathsf{negl}(n)$:

- $\mathsf{GapK}[n - n^{0.99}, n - \omega(\log n)]$;
- $\mathsf{GapKT}[n - n^{0.99}, n - \omega(\log n)]$.

Proof. Suppose there is a one-way function in NC^0, then there is also a PRG in NC^0 [HRV13]. Moreover, we may assume that the PRG has stretch $n^{0.999}$, i.e., it stretches $n - n^{0.999}$ random bits into n pseudorandom bits. (See e.g., [AIK06, Remark 6.2].)

Let G be a PRG in NC^0 with stretch $n^{0.999}$. Consider the distribution
$$\mathcal{D} = \frac{1}{2}(U_n + G(U_{n-n^{0.999}})).$$

Since G is a PRG in NC^0, $\mathcal{D}$ is $O(1)$-locally samplable.

Any output from $G(U_{n-n^{0.999}})$ is a YES instance of $\mathsf{GapK}[n - n^{0.99}, n - \omega(\log n)]$, since it can be described by a seed of length $n - n^{0.999}$ and the code of G which has length $O(1)$. On the other hand, the probability that a random string of length n has Kolmogorov complexity $\le n - \omega(\log n)$ is $\le \mathsf{negl}(n)$. It follows that $\mathsf{GapK}[n - n^{0.99}, n - \omega(\log n)]$ is hard on average under $\mathcal{D}$.

Since G is a PRG in NC^0, its outputs also have small KT complexity. Therefore the above hardness result also holds for $\mathsf{GapKT}[n - n^{0.99}, n - \omega(\log n)]$. □

4.2 Equivalences

We begin by proving an equivalence between the existence of one-way functions and the average-case hardness of GapK and $\mathsf{GapMCSP}$.

Theorem 39. *Let $q \geq 1$ and $0 < \delta < 1$ be constants. Let $n^{\Omega(1)} \leq s \leq n$ and $\omega(\log n) \leq \Delta \leq s/2$. Let $n^{5\delta} < s' < n^{1-6\delta}$. Then the following are equivalent.*

1. *One-way functions exist.*
2. *There is a samplable distribution $\mathcal{D}$ such that no PPT algorithm infinitely-often solves $\mathsf{GapK}[n^\delta, n - \omega(\log n)]$ on $\mathcal{D}$ with probability greater than $1/2 + \mathsf{negl}(n)$.*
3. *There is a samplable distribution $\mathcal{D}$ such that no PPT algorithm infinitely-often solves $\mathsf{GapK}[s - \Delta, s]$ on $\mathcal{D}$ with probability greater than $1 - 1/n^q$.*
4. *There is an (n^δ)-locally samplable distribution $\mathcal{D}$ such that no PPT algorithm infinitely-often solves $\mathsf{GapMCSP}[n^\delta, o(\frac{n}{\log n})]$ on $\mathcal{D}$ with probability greater than $1/2 + \mathsf{negl}(n)$.*
5. *There is an (n^δ)-locally samplable distribution $\mathcal{D}$ such that no PPT algorithm infinitely-often solves $\mathsf{GapMCSP}[s', s'n^{5\delta}]$ on $\mathcal{D}$ with probability greater than $1 - 1/n^q$.*

Proof. (2) $\implies$ (3) and (4) $\implies$ (5) are trivial.
(1) $\implies$ (2) follows from Theorem 36.
(3) $\implies$ (1) follows from Corollary 21.
(1) $\implies$ (4) follows from Theorem 37.
(5) $\implies$ (1) follows from Theorem 32. $\square$

We also get an equivalence between the average-case hardness of GapK^t and the existence of infinitely-often one-way functions under a derandomization assumption.

Theorem 40. *Assume that $\mathsf{E} \not\subseteq \mathsf{ioSIZE}[2^{.01n}]$. Then the following are equivalent:*

1. *Infinitely-often one-way functions exist.*
2. *For every constant $0 < \epsilon < 1$, there is a samplable distribution $\mathcal{D}$ such that for every large enough polynomial τ, no PPT algorithm solves $\mathsf{GapK}^\tau[n^\epsilon, n - \omega(\log n)]$ on $\mathcal{D}$ with probability greater than $1/2 + \mathsf{negl}(n)$.*
3. *There are functions $s \in [n^{\Omega(1)}, n]$, $\Delta = \omega(\log n)$, a constant $q \geq 1$, and a samplable distribution $\mathcal{D}$, such that for every large enough polynomial τ, no PPT algorithm infinitely-often solves $\mathsf{GapK}^\tau[s - \Delta, s]$ on $\mathcal{D}$ with probability greater than $1 - 1/n^q$.*

Proof. (2) $\implies$ (3) is trivial.
(1) $\implies$ (2) follows from (an infinitely-often version of) Theorem 36.
(3) $\implies$ (1) follows from Theorem 28. $\square$

We can also characterize the existence of one-way functions in NC^0 by the average-case hardness of GapK over $O(1)$*-locally samplable distributions*:

Theorem 41. *The following are equivalent:*

1. *There are one-way functions computable in logspace.*
2. *There are one-way functions computable in NC^0.*
3. *There is an NC^0-samplable distribution $\mathcal{D}$ such that no PPT algorithm infinitely-often solves $\mathsf{GapK}[n - n^{0.99}, n - \omega(\log n)]$ on $\mathcal{D}$ with probability greater than $1/2 + \mathsf{negl}(n)$.*

4. *There are a constant $c > 0$, a logspace-samplable distribution $\mathcal{D}$, and efficiently computable functions $s_1(n), s_2(n) < n$, such that $s_2 - s_1 = \omega(\log n)$, and no PPT algorithm infinitely-often solves $\mathsf{GapK}[s_1, s_2]$ on $\mathcal{D}$ with probability greater than $1 - 1/n^c$.*

Proof. (1) $\iff$ (2) is the main result of [AIK06].
(2) $\implies$ (3) follows from Theorem 38.
(3) $\implies$ (4) is trivial.
(4) $\implies$ (2) follows from Corollary 23. □

Finally, we attempt to use the average-case hardness of GapKT to characterize the existence of one-way functions in NC^0. We could not obtain an equivalence result with current techniques; we could only show that "one-way functions in NC^0" is between an "easier" version and a "harder" version of GapKT. The two versions have different parameters — the "easier" version is $\mathsf{GapKT}[o(\frac{n}{\log n}), n - \omega(\log n)]$ while the "harder" version is $\mathsf{GapKT}[n - n^{0.99}, n - \omega(\log n)]$.

Theorem 42. *The following items hold:*

1. *Suppose there exist one-way functions in NC^0. Then there exists an $O(1)$-locally samplable distribution $\mathcal{D}$ such that no PPT algorithm infinitely-often solves $\mathsf{GapKT}[n - n^{0.99}, n - \omega(\log n)]$ with probability greater than $1/2 + \mathsf{negl}(n)$.*

2. *Suppose there exist no one-way functions in NC^0. Then for every constant $q \geq 1$ and every $O(1)$-locally samplable distribution $\mathcal{D}$, there is a randomized PPT algorithm that infinitely-often solves $\mathsf{GapKT}[o(\frac{n}{\log n}), n - \omega(\log n)]$ with probability $\geq 1 - 1/n^q$.*

Proof. The first item follows from Theorem 38. The second item follows from Theorem 35. □

5 Applications

5.1 One-Way Functions from Hardness of SAT and Clique

We will be considering average-case hardness assumptions on k-SAT and Clique. In particular, we show that if these problems are hard on average under samplable distributions with large enough entropy, then one-way functions exist.

Definition 43. Given positive integers k and $\Delta > 0$, k-$\mathsf{SAT}(n, \Delta n)$ is the set of satisfiable k-CNF formulas on n variables with Δn clauses. Given a function $t : \mathbb{N} \to \mathbb{N}$, t-$\mathsf{Clique}(n)$ is the set of graphs on n vertices that have a clique of size at most $t(n)$.

We need suitable encodings of k-$\mathsf{SAT}(n, \Delta n)$ and t-$\mathsf{Clique}(n)$ such that the problems are defined on all input lengths. It is important for us that the problems are defined on all input lengths because one-way functions naturally correspond to *almost everywhere* hardness.

We choose the following natural encoding for k-SAT: for k-CNF formulas on n variables and Δn clauses, let $m(n, k, \Delta) = \Delta n(k\lceil\log(n)\rceil + k)$. We can represent a k-CNF formula on n variables and Δn clauses with $m(n, k, \Delta)$ bits by specifying, for each clause, the sequence of k literals in the clause. A literal can be specified with $\lceil\log(n)\rceil + 1$ bits by giving the index of a variable as well as its polarity. For lengths m that are not of the form $m(n, k, \Delta)$, we interpret an input of length m as a padded version of an input of length m', where m' is the largest $m(n, k, \Delta) < m$. In other words, we ignore the last $m - m'$ bits of the input.

Similarly we choose the following natural encoding for t-Clique: for a graph G on n vertices, let $m(n) = \binom{n}{2}$. We interpret a string x of bit-length $m(n)$ as a graph G_x on n vertices, where the i'th bit of x is 1 iff the i'th edge (out of the $\binom{n}{2}$ possible edges, lexicographically ordered) is present in G. If m is not of the form $m(n)$, then we interpret a string x of length m as a padded version of an input of length m', where m' is the largest $m(n) < m$.

We emphasize that the specific choice of these encodings isn't crucial in our results — as mentioned before, what is important is that the problem is non-trivial on all input lengths.

Definition 44. Let D_n be a distribution on n-bit strings. We say that D_n has *entropy deficiency* at most ℓ if the entropy of D_n is at least $n - \ell$.

We first need a lemma stating that a string sampled from a high entropy distribution must have high Kolmogorov complexity with noticeable probability.

Lemma 45. *Let D_n be a distribution on n-bit strings with entropy k. With probability at least $1/n$ over x sampled from D_n, $\mathsf{K}(x) > k - 3$.*

Proof. Suppose, for the sake of contradiction, that with probability greater than $1 - 1/n$ over x sampled from D_n, $\mathsf{K}(x) \leq k - 3$. Let Low be the set of n-bit strings x with $\mathsf{K}(x) \leq k - 3$ and High be the set of n-bit strings with $\mathsf{K}(x) > k - 3$. Consider the random variable S defined to be "high" if x sampled from D_n is in High and "low" if x sampled from D_n is in Low. Let p_{high} be the probability that S is "high" and p_{low} the probability that S is "low". By the chain rule for entropy, we have that

$$H(D_n) \leq H(D_n, S) = H(S) + p_{\mathsf{low}} H(D_n \mid S \text{ is "low"}) + p_{\mathsf{high}} H(D_n \mid S \text{ is "high"}).$$

Since S is defined on a 2-element set, the first term in the RHS is at most 1. Since Low has size at most 2^{k-2}, the second term is at most $k - 2$. Since $p_{\mathsf{high}} < 1/n$ and $H(D_n \mid S \text{ is "high"}) \leq H(D_n) \leq n$, the third term is less than 1. Hence $H(D_n) < k$, in contradiction to the assumption on D_n. □

A key ingredient in our proofs is that satisfiable formulas and graphs with large cliques can be compressed.

Lemma 46. *If $\phi \in k\text{-}\mathsf{SAT}(n, \Delta n)$ is satisfiable, then $\mathsf{K}(\phi) \leq n + \Delta n \log((2^k - 1)\binom{n}{k}) + O(\log(n))$.*

Proof. Let ϕ be a satisfiable k-CNF on n variables with Δn clauses, and let $z \in \{0,1\}^n$ be a satisfying assignment to ϕ. We will describe ϕ using z and the index i of ϕ in the lexicographic ordering of all sequences of Δn k-clauses for which each clause is satisfied by z. Note that there are at most $(2^k - 1)\binom{n}{k}$ k-clauses satisfied by z - given a choice of k variables, satisfiability of z excludes one out of the 2^k polarities for these variables in a clause. Hence the index of ϕ can be described with at most $\Delta n \log((2^k - 1)\binom{n}{k}) + O(1)$ bits. To describe ϕ, we specify n, z, the index i and a constant-size program that recovers ϕ from n, z, i. This takes $n + \Delta n \log((2^k - 1)\binom{n}{k}) + O(\log(n))$ bits overall. □

Lemma 46 gives a saving over the encoding length of arbitrary k-CNFs with Δn clauses when $\Delta \gg 2^k$. Note that an arbitrary k-CNF with Δn clauses requires encoding length at least $\Delta n \log(2^k \binom{n}{k})$.

Lemma 47. *Let $t : \mathbb{N} \to \mathbb{N}$ be a function. If an n-vertex graph G has a clique of size at least $t(n)$, then $\mathsf{K}(G) \leq t \log(n) + \binom{n}{2} - \binom{t}{2} + O(\log(n))$.*

Proof. Suppose that G has a clique S of size $t(n)$. We will describe G using the vertices in S together with all edges such that at least one endpoint of the edge is not in S. The vertices in S can be described with $t \log(n)$ bits. The edges that are not contained in S can be described with $\binom{n}{2} - \binom{t}{2}$ bits given knowledge of S, by a bitstring specifying for each pair of vertices (u, v) in lexicographic order, where at least one of u and v is not in S, whether (u, v) is an edge of G. Thus G can be described by specifying n, S and edges not in S, together with a program for reconstructing G from these information. This takes $t \log(n) + \binom{n}{2} - \binom{t}{2} + O(\log(n))$ bits overall. □

Lemma 47 gives a saving over the encoding length of arbitrary n-vertex graphs when $t(n) = \omega(\log(n))$. Note that an arbitrary n-vertex graph requires encoding length at least $\binom{n}{2}$.

Theorem 48. *Let $\Delta \geq 2^{k+3}$ be an integer. Suppose that there is a samplable distribution $\mathcal{D}$ (resp. logspace-samplable distribution $\mathcal{D}$) such that every probabilistic polynomial-time algorithm solving k-$\mathsf{SAT}(n, \Delta n)$ on $\mathcal{D}$ has error at least $1/2 - 1/m^2$ for all but finitely many m (where m is the bit-length of the encoding of the k-SAT formula), and moreover $\mathcal{D}$ has entropy deficiency at most $\Delta n/(2^{k+1})$. Then one-way functions (resp. one-way functions computable in NC^0) exist.*

Proof. We give an average-case reduction implying that k-$\mathsf{SAT}(n, \Delta n)$ can be solved with error at most $1/2 - 1/100m$ on $\mathcal{D} = \{D_m\}$ for infinitely many m if $\mathsf{GapK}[s - \lambda, s]$ can be solved with error at most $1/m^3$ on D_m for infinitely many m, for some efficiently computable $s = \Omega(m), \lambda = \omega(\log(m))$. The theorem follows by combining this reduction with Corollary 21 in case $\mathcal{D}$ is samplable, and with Corollary 23 in case $\mathcal{D}$ is logspace-samplable.

For a given m, let n be the largest integer such that $m(n, k, \Delta) \leq m$. We choose $s(m) = 2n + \Delta n \log((2^k - 1)\binom{n}{k})$ and $\lambda(m) = \log^2(n)$. Suppose that $\mathsf{GapK}[s - \lambda, s]$ can be solved with error at most $1/m^3$ on D_m for infinitely many m. Let $I \subseteq \mathbb{N}$ be the set of such good input lengths m. By a simple Markov argument, there is a PPT algorithm $\mathcal{A}$ such that for every $m \in I$, with probability at least $1 - 1/m^2$ on $x \leftarrow D_m$, the following holds:

- If $\mathsf{K}(x) \leq s(m) - \lambda(m)$ then $\mathcal{A}(x)$ outputs 1 with probability $\geq 2/3$, and if $\mathsf{K}(x) \geq s(m)$ then $\mathcal{A}(x)$ outputs 0 with probability $\geq 2/3$.

If the above item holds for some $x \in \{0,1\}^m$, then we say x is *good*. (In particular, every x such that $s(m) - \lambda(m) < \mathsf{K}(x) < s(m)$ is good.) Then a random sample from D_m is good w.p. $\geq 1 - 1/m^2$.

We use $\mathcal{A}$ to define a probabilistic polynomial-time algorithm $\mathcal{A}'$ that solves k-$\mathsf{SAT}(n, \Delta n)$ on D_m with error $\leq 1/2 - 1/100m$ for infinitely many m. Given input x, $\mathcal{A}'$ simply simulates $\mathcal{A}$ on x. If $\mathcal{A}$ outputs 0, then $\mathcal{A}'$ outputs 0. If $\mathcal{A}$ outputs 1, $\mathcal{A}'$ outputs a fixed bit b hardcoded into the algorithm, which we specify later.

Clearly, $\mathcal{A}'$ is a probabilistic polynomial-time algorithm. We need to choose the bit b so that $\mathcal{A}'$ solves k-$\mathsf{SAT}(n, \Delta n)$ on D_m with error $\leq 1/2 - 1/100m$ infinitely often. We say that $\mathcal{A}$ *0-classifies* x if $\mathsf{K}(x) > s(m) - \lambda(m)$ and $\mathcal{A}(x)$ outputs 0 with probability $> 2/3$. We choose b to be 1 if for infinitely many $m \in I$, $x \leftarrow D_m$ is the encoding of a satisfiable formula with probability at least $1/2$ conditioned on $\mathcal{A}$ not 0-classifying x; and $b = 0$ otherwise. Let $I' \subseteq I$ be an infinite set of input lengths such that for each $m \in I'$, b is the correct answer for satisfiability of $x \leftarrow D_m$ with probability at least $1/2$ conditioned on $\mathcal{A}$ not 0-classifying x. We show that $\mathcal{A}'$ correctly solves satisfiability on x with error at most $1/2 - 1/100m$ on D_m for each $m \in I'$.

The key point is that $\mathcal{A}'$ always solves k-SAT correctly on inputs x for which $\mathcal{A}$ 0-classifies x, and this event happens with noticeable probability over D_m when $m \in I'$. Indeed, if $\mathcal{A}$ 0-classifies x, then $\mathsf{K}(x) > s(m) - \lambda(m)$, and x would be an unsatisfiable formula by Lemma 46 and our choice of Δ. Also, by Lemma 45 and the upper bound on entropy deficiency of $\mathcal{D}$, with probability at least $1/m$ over $x \leftarrow D_m$, $\mathsf{K}(x) > m - \Delta n/2^{k+1} - 3 > s(m)$. Let p be the probability that $\mathcal{A}$ 0-classifies x for a random $x \leftarrow D_m$, then

$$p \geq \Pr_{x \leftarrow D_m}[x \text{ is good and } \mathsf{K}(x) > s(m)] \geq 1/m - 1/m^2,$$

which is $\geq 1/(2m)$ for $m \geq 2$. Since b is the correct value for k-SAT on $x \leftarrow D_m$ with probability at least $1/2$ conditioned on $\mathcal{A}$ not 0-classifying x, we have that

$$\begin{aligned}
\Pr_{x \leftarrow D_m, \mathcal{A}}[\mathcal{A}(x) \text{ is correct}] = & \Pr_{x \leftarrow D_m, \mathcal{A}}[\mathcal{A}(x) \text{ is correct} \mid \mathcal{A} \text{ 0-classifies } x] \cdot p \\
& + \Pr_{x \leftarrow D_m, \mathcal{A}}[\mathcal{A}(x) \text{ is correct} \mid \mathcal{A} \text{ does not 0-classify } x] \cdot (1 - p) \\
& \geq (2/3) \cdot p + (1/2) \cdot (1 - p) \\
& = 1/2 + p/6 \\
& \geq 1/2 + 1/100m.
\end{aligned}$$

□

We note that any reasonable encoding of a k-SAT formula on n variables with a linear number of clauses has bit-length $O(n\log(n))$, while the entropy deficiency upper bound for Theorem 48 to apply is $O(n)$.

Next we show an analogous result to Theorem 48 for Clique.

Theorem 49. *Let $t : \mathbb{N} \to \mathbb{N}$ be a function such that $t(n) = \omega(\log(n))$. Suppose that there is a samplable distribution $\mathcal{D}$ (resp. logspace-samplable distribution $\mathcal{D}$) such that every probabilistic polynomial-time algorithm solving t-Clique on $\mathcal{D}$ has error at least $1/2 - 1/m^2$ for all but finitely many m, and moreover $\mathcal{D}$ has entropy deficiency at most $(1-\epsilon)\binom{t}{2}$ for some constant $\epsilon > 0$. Then one-way functions (resp. one-way functions computable in NC^0) exist.*

Proof Sketch. The proof is very similar to that of Theorem 48, so we just provide a sketch. We give an average-case reduction implying that t-Clique(n) can be solved with error at most $1/2 - 1/n^{\Omega(1)}$ on $\mathcal{D} = \{D_m\}$ for infinitely many m if GapK$[s-\lambda, s]$ can be solved with error at most $1/m^q$ on D_m for infinitely many m, for some constant q and efficiently computable $s = \Omega(m), \lambda = \omega(\log(m))$. The theorem follows by combining this reduction with Corollary 21 in case $\mathcal{D}$ is samplable, and with Corollary 23 in case $\mathcal{D}$ is logspace-samplable.

For a given m, let n be the largest integer such that $m(n) \le m$. Let $\epsilon > 0$ be a constant such that $\mathcal{D}$ has entropy deficiency at most $(1-\epsilon)t(n)$. We choose $s(m) = \binom{n}{2} - (1-\epsilon/2)\binom{t}{2}$ and $\lambda = \log(n)^2$. The reduction is exactly the same as in the proof of Theorem 48, except that we use Lemma 47 instead of Lemma 46 and the parameters defined above to argue correctness. □

While the upper bound on entropy deficiency is inherently sub-linear in Theorem 48, it can be chosen to be linear in Theorem 49 by choosing $t = \Omega(n)$. As far as we are aware, there is no evidence against $\Omega(n)$-Clique being hard on average on some samplable distribution.

A natural question is whether the assumption of Theorem 49 is believable. We justify this assumption by showing that it follows from the well-known Planted Clique Hypothesis (thus generalizes the latter).

Given a function $t : \mathbb{N} \to \mathbb{N}$, the t-Planted Clique Hypothesis [Jer92, Kuc95, AKS98, JP00] states that for almost all n, no probabilistic polynomial-time algorithm can distinguish with probability $1/n^{O(1)}$ between a random graph on n vertices where each edge is included independently with probability $1/2$ and the union of a random graph and a clique on a uniformly chosen subset of t vertices. It is consistent with our current knowledge of algorithms for Clique that the t-Planted Clique Hypothesis holds for $t(n) = n^{0.49}$[Jer92, FK03, MPW15].

Since for $t = \omega(\log(n))$, a random graph has a t-Clique with negligible probability, the t-Planted Clique Hypothesis also implies that t-Clique is hard on average with respect to the distribution on graphs obtained by choosing a random graph with probability $1/2$ and the union of a random graph and a random planted clique of size t with probability $1/2$.

Proposition 50. *Let $t : \mathbb{N} \to \mathbb{N}$ be a function. If the t-Planted Clique Hypothesis holds, then there is a samplable distribution $\mathcal{D}$ with entropy deficiency at most $\binom{t}{2}/2$ such that every probabilistic polynomial-time algorithm solving t-Clique(n) has error at least $1/2 - 1/n^{\omega(1)}$ on all but finitely many n.*

Proof. Consider the distribution $\mathcal{D} = \{D_m\}$ obtained as follows. Choose a random size-$t(n)$ subset S of $[n]$ and a uniformly random G on n vertices where each edge is included independently with probability $1/2$. Let $C[S]$ be the clique on S. With probability $1/2$, output G, and with probability $1/2$ output the union of G and $C[S]$. By the t-Planted Clique Hypothesis, t-Clique(n) is strongly hard on this distribution for all but finitely many n. The entropy of D_m is at least $\binom{m}{2}/2 + (\binom{m}{2} - \binom{t}{2})/2 = \binom{m}{2} - \binom{t}{2}/2$, and hence the entropy deficiency is at most $\binom{t}{2}/2$. □

In order to similarly justify the assumption of Theorem 48, we consider the following hypothesis on pseudorandomness of random local functions, which is inspired by the candidate OWF proposed by Goldreich [Gol00, App13, App16].

The Random Local Function Pseudorandomness Hypothesis states that for any $c > 0$, there is an integer k and a k-ary predicate P such that for every $\Delta > 0$ no probabilistic poly-time algorithm can distinguish the following 2 distributions with probability $\geq 1/n^c$:

1. Δn uniformly chosen k-tuples $S_1, \ldots S_{\Delta n}$ of $[n]$ together with uniformly and independently chosen Δn-bit string z;

2. Δn uniformly chosen k-tuples $S_1, \ldots S_{\Delta n}$ of $[n]$ together with the evaluations of P on $x|_{S_1}, \ldots x|_{S_{\Delta n}}$ on a uniformly and independently chosen n-bit string x.

The following proposition is closely analogous to Proposition 50, and we omit the proof.

Proposition 51. *Suppose the Random Local Function Pseudorandomness Hypothesis holds. Then there are integers k and Δ and a samplable distribution $\mathcal{D}$ with entropy deficiency at most $\Delta n/2^{k+1}$ such that any probabilistic polynomial-time algorithm solving k-$\mathsf{SAT}(n, \Delta n)$ on $\mathcal{D}$ has error at least $1/2 - 1/m^3$ for all but finitely many m.*

5.2 Unconditional Average-Case CZK Protocols

In this subsection, we show unconditionally that $\mathsf{GapK}[s - \omega(\log n), s]$ admits an average-case computational zero-knowledge protocol on infinitely many input lengths.

We first define the properties that our interactive proof system will have: namely, our proof system is polynomially-bounded and computationally zero-knowledge. (For a rigorous definition of *interactive protocols*, we refer the reader to Section 2.5 of [Vad06].)

Definition 52. Let (P, V) be an interactive protocol where P stands for "prover" and V stands for "verifier". We use $\langle P, V\rangle(x)$ to denote V's view of the interaction on input x.

- We say (P, V) is *polynomially-bounded* if the total length of messages exchanged between P and V are polynomially-bounded, and V is a PPT algorithm.

- Let x be an input. We say (P, V) *accepts* x if V accepts in $\langle P, V\rangle(x)$ with probability 1, and (P, V) *rejects* x if for every prover $P^\star$, V accepts in $\langle P^\star, V\rangle(x)$ with probability at most 1/2.[18]

- We say (P, V) is *(computationally) zero-knowledge* if there is an oracle PPT S (called a "simulator") such that for every non-uniform PPT $V^\star$ and every x such that (P, V) accepts x, the distributions $\langle P, V^\star\rangle(x)$ and $S^{V^\star}$ are computationally indistinguishable.

We show unconditionally that $\mathsf{GapK}[s - \omega(\log n), s]$ admits an average-case computational zero-knowledge protocol.

Theorem 53. *Let $\mathcal{D} = \{D_n\}$ be a samplable distribution, $\Delta = \omega(\log n)$, and $q \geq 1$. There is a* CZK *protocol that on infinitely many input lengths n, solves* $\mathsf{GapK}[s - \Delta, s]$ *on $\mathcal{D}$ with error at most $1/n^q$.*

More precisely, there is a polynomially-bounded zero-knowledge interactive protocol (P, V) such that for infinitely many n,

$$\Pr_{x \leftarrow D_n}[\mathsf{K}(x) \leq s - \Delta \text{ and } (P, V) \text{ does not accept } x] \leq 1/2n^q,$$

and

$$\Pr_{x \leftarrow D_n}[\mathsf{K}(x) > s \text{ and } (P, V) \text{ does not reject } x] \leq 1/2n^q.$$

[18]Note that it is possible that an interactive protocol (P, V) neither accepts nor rejects some input x.

Proof. The proof is by a win-win argument. Roughly speaking, if one-way functions exist, then $\mathsf{CZK} = \mathsf{IP} = \mathsf{PSPACE}$, and thus we can implement the algorithm in Theorem 15 in CZK; if one-way functions do not exist, then Corollary 21 shows that GapK can be solved in average in probabilistic polynomial time.

Case I: one-way functions exist. Suppose one-way functions exist, then $\mathsf{CZK} = \mathsf{IP}$ [IY87, BGG$^+$88, Nao91, HILL99]. It is also known that $\mathsf{IP} = \mathsf{PSPACE}$ [LFKN92, Sha92]. Let $\mathcal{A}_1$ be the unconditional heuristic for $\mathsf{GapK}[s - \omega(\log n), s]$ given in Theorem 15 that has error probability at most $1/2n^q$ over $\mathcal{D}$. Note that $\mathcal{A}_1$ can be implemented in PSPACE, thus there is a polynomially-bounded zero-knowledge interactive protocol (P, V) such that for every input $x \in \{0,1\}^\star$:

- if $\mathcal{A}_1(x)$ accepts, then (P, V) accepts x;
- if $\mathcal{A}_1(x)$ rejects, then (P, V) rejects x.

It follows that for every input length n,

$$\begin{aligned} &\Pr_{x \leftarrow \mathcal{D}_n}[\mathsf{K}(x) \le s - \Delta \text{ and } (P,V) \text{ does not accept } x] \\ = &\Pr_{x \leftarrow \mathcal{D}_n}[\mathsf{K}(x) \le s - \Delta \text{ and } \mathcal{A}_1(x) \text{ rejects}] \le 1/2n^q, \end{aligned}$$

and

$$\begin{aligned} &\Pr_{x \leftarrow \mathcal{D}_n}[\mathsf{K}(x) > s \text{ and } (P,V) \text{ does not reject } x] \\ = &\Pr_{x \leftarrow \mathcal{D}_n}[\mathsf{K}(x) > s \text{ and } \mathcal{A}_1(x) \text{ accepts}] \le 1/2n^q. \end{aligned}$$

Case II: one-way functions do not exist. Suppose no one-way functions exist. By Corollary 21, there is a PPT algorithm $\mathcal{A}_2$ for $\mathsf{GapK}[s-\Delta, s]$ that has error at most $1/2n^q$ over $\mathcal{D}$, on infinitely many input lengths. Consider the following trivial CZK protocol for $\mathsf{GapK}[s - \Delta, s]$: on input x, the verifier outputs $\mathcal{A}_2(x)$ without interacting with the prover at all. This trivial protocol solves $\mathsf{GapK}[s - \Delta, s]$ on average on infinitely many input lengths. $\square$

Similarly, one can also show unconditionally that $\mathsf{GapMCSP}$ is solvable by a CZK protocol on average:

Corollary 54. *Let $0 < \delta < 1$, $q \ge 1$, and $\mathcal{D} = \{D_n\}$ be an (n^δ)-locally samplable distribution. Let $s = \Omega(n^\delta)$. Then there is a CZK protocol that on infinitely many input lengths n, solves $\mathsf{GapMCSP}[s, sn^{5\delta}]$ on $\mathcal{D}$ with error at most $1/n^q$.*

Proof Sketch. Suppose one-way functions exist. As MCSP is solvable in polynomial space, it follows that $\mathsf{GapMCSP}[s, sn^{5\delta}] \in \mathsf{CZK}$.

On the other hand, suppose there are no one-way functions. By Theorem 32, there is a randomized polynomial-time algorithm (i.e. a trivial CZK protocol) for $\mathsf{GapMCSP}[s, sn^{5\delta}]$ that has error at most $1/n^q$ over $\mathcal{D}$, on infinitely many input lengths. $\square$

5.3 Limitations of Randomized Local Reductions

In this subsection, we discuss limitations on the power of *local reductions* for proving hardness of $\mathsf{GapMCSP}$. An (n^δ)-local reduction from a language L to a language L' is a polynomial-time algorithm R mapping m-bits to n-bits such that for all x

$$L(x) = 1 \iff L'(R(x)) = 1$$

and the i'th output bit of $R(x)$ is computable in deterministic time n^δ given i and random access to x.

Murray and Williams [MW17] show that while almost all known NP-complete problems are NP-complete under $(n^{o(1)})$-local reductions, MCSP is *unconditionally* not NP-hard under *deterministic* $(n^{.49})$-local reductions. As a result, there is a formal sense in which a NP-hardness result for MCSP must "look very different" from existing NP-hardness results.

A natural question is exactly how different would such a reduction have to be. For example, what if one considers *randomized* local reductions instead of deterministic local reductions? Could there be a relatively simple randomized local reduction from, say, SAT to MCSP? This question gains further motivation in light of recent results showing NP-hardness for the multi-output and conditional versions of MCSP using relatively simple randomized reductions [ILO20, Ila20].

Our main result in this subsection is to give evidence against the existence of an (n^δ)-local *randomized* reduction from SAT to $\mathsf{GapMCSP}$ with a sufficiently large gap. A randomized (n^δ)-local reduction from L to L' is a randomized polynomial-time algorithm R that maps m bits to n bits such that for all x

$$\Pr_R[L(x) = L'(R(x))] \geq 2/3$$

and the i'th output bit of $R(x)$ is computable in deterministic time n^δ given i and random access to both x and the randomness used by R.

Before we state our results, we introduce some notation. Let $T \in \{0,1\}^n$, $\rho \in \{0,1,\star\}^m$, $r \in \{0,1\}^m$, $i \in [n]$. Unless otherwise stated, let LSamp be an (n^δ)-locally samplable distribution for some $0 < \delta < 1$. Throughout this subsection, let $\mathcal{O}$ denote a (deterministic) oracle that on input (T, ρ, i) outputs an estimate to the quantity

$$\Pr_{\tilde{r} \leftarrow \rho}[\mathsf{LSamp}(n, i, \tilde{r}) = T(i)]$$

with additive error at most .01.[19]

Let the *unlikeliness* of r relative to $\mathcal{O}$, denoted $u^{\mathcal{O}}(r)$, be the number of iterations needed for the following loop to terminate. Initially, set $\rho = \star^m$ and $T = \mathsf{LSamp}(n, r)$. While there is a (lexicographically first) coordinate $i \in [n]$ such that $\mathcal{O}(T, \rho, i) < 3/4$, set $\rho(j) = r(j)$ for all indices $j \in [n]$ of r that are queried by LSamp on input (n, i, r) (recall LSamp gets random access to r). This completes our definition of $u^{\mathcal{O}}(r)$. We also let $\rho^{\mathcal{O}}(r)$ denote the value of ρ when the above loop terminates.

We now bound the probability that LSamp outputs some fixed string and has high unlikeliness.

Lemma 55. *Fix some $T \in \{0,1\}^n$. Then*

$$\Pr_{r \leftarrow \{0,1\}^m}[\mathsf{LSamp}(n, r) = T \text{ and } u^{\mathcal{O}}(r) \geq k] \leq (4/5)^k.$$

Proof. Fix $T \in \{0,1\}^m$. We will analyze how the loop above that defines $u^{\mathcal{O}}(r)$ acts when r is chosen uniformly at random. In particular, we consider the following loop. Set ρ initially to be $\star^m$. While there exists a (lexicographically first) coordinate $i \in [n]$ with $\mathcal{O}(T, \rho, i) < 3/4$, simulate running LSamp on (n, i, ρ) and whenever LSamp queries an index j of ρ that is undefined, set $\rho(j)$ to be an independent random bit and respond with that bit.

Let ρ_k denote the value of ρ on the k'th iteration of the loop. Our convention is that $\rho_0 = \star^m$ is the value of ρ before any iteration. We will show that for all k

$$\Pr[\rho_k \text{ is defined and } \exists \tilde{r} \text{ that agrees with } \rho_k \text{ satisfying } T = \mathsf{LSamp}(n, \tilde{r})] \leq (4/5)^k$$

[19]To make the output of the $\mathcal{O}$ usable to polynomial-time algorithms, we assume its output has bit length $O(n)$.

where the probability is over the randomness in the loop. Observe that this probability bound implies the lemma.

We prove the probability bound by induction on k. For the inductive step, fix a restriction[20] ρ_k and assume i is a (lexicographically first) coordinate with $\mathcal{O}(T, \rho_k, i) < 3/4$. Let ρ_{k+1} be the random variable that is equal to the (updated) value of ρ after one iteration of the above loop starting with $\rho = \rho_k$. Then we have that

$$\begin{aligned}
&\Pr_{\rho_{k+1}}[\exists \tilde{r}' \text{ that agrees with } \rho_{k+1} \text{ and } T = \mathsf{LSamp}(n, \tilde{r}')]\\
\leq &\Pr_{\rho_{k+1}}[\exists \tilde{r}' \text{ that agrees with } \rho_{k+1} \text{ and } T(i) = \mathsf{LSamp}(n, i, \tilde{r}')]\\
\leq &\Pr_{\tilde{r} \leftarrow \rho_k}[\mathsf{LSamp}(n, i, \tilde{r}) = T(i)]\\
\leq &\, 4/5
\end{aligned}$$

where the second inequality comes from the fact that ρ' defines all the coordinates needed to compute $\mathsf{LSamp}(n, i, \tilde{r})$ (and chooses the coordinates not already defined by ρ uniformly at random), and the third inequality comes from the approximation guarantee on $\mathcal{O}$ and the assumption that $\mathcal{O}(T, \rho', i) < 3/4$.

Inductively, we can apply this bound to get that

$$\Pr[\rho_k \text{ is defined and } \exists \tilde{r} \text{ that agrees with } \rho_k \text{ satisfying } T = \mathsf{LSamp}(n, \tilde{r})] \leq (4/5)^k. \qquad \square$$

On the other hand, we show that if T has high circuit complexity, then T can never be output on an r with low unlikeliness.

Lemma 56. *Let $r \in \{0,1\}^m$ and $T = \mathsf{LSamp}(n, r)$. Then $\mathsf{CC}(T) \leq O(u^{\mathcal{O}}(r) n^{3\delta})$.*

Proof. Let $\rho = \rho^{\mathcal{O}}(r) \in \{0, 1, \star\}^m$. By construction of ρ and the approximation guarantee on $\mathcal{O}$, we know that for all $i \in [n]$

$$\Pr_{\tilde{r} \leftarrow \rho}[\mathsf{LSamp}(n, i, \tilde{r}) = T(i)] \geq 2/3.$$

We also know that the number of binary values in ρ is at most $u^{\mathcal{O}}(r) n^{\delta}$ (since LSamp runs in time n^{δ} so each iteration of the loop sets at most n^{δ} coordinates). Thus, applying Lemma 29, we get that $\mathsf{CC}(T) \leq O(u^{\mathcal{O}}(r) n^{3\delta})$. $\square$

Next, we show that there is an efficient randomized algorithm with the properties we want from $\mathcal{O}$.

Proposition 57. *There is a polynomial-time randomized algorithm $\mathcal{A}$ such that for all $T \in \{0,1\}^n, \rho \in \{0, 1, \star\}^m$, and $i \in [n]$, we have that $\mathcal{A}(T, \rho)$ estimates the quantity*

$$\Pr_{\tilde{r} \leftarrow \rho}[\mathsf{LSamp}(n, i, \tilde{r}) = T(i)]$$

within additive error at most .01. Moreover, the failure probability of the algorithm is at most $2^{-(n+m)^2}$.

Proof. The algorithm is simple. Given ρ and T, it just samples a uniformly random $\tilde{r}$ agreeing with ρ and checks if $\mathsf{LSamp}(n, i, \tilde{r}) = T(i)$. It repeats this independently a (sufficiently large) polynomial number of times and outputs the empirical probability. This algorithm clearly runs in polynomial time, and its correctness follows from a Chernoff bound. $\square$

Finally, we use all this machinery to show that one can *unconditionally* solve $\mathsf{GapMCSP}$ (with a sufficiently large gap) on any (n^{δ})-locally samplable distribution on average.

[20]To emphasize, ρ_k is some fixed element of $\{0, 1, \star\}^m$, not an uncertain random variable.

Theorem 58. *There is a randomized polynomial-time algorithm $\mathcal{A}$ such that $\mathcal{A}(n, r)$ computes*

$$\mathsf{GapMCSP}[s, \omega(n^{5\delta} s \log s)](\mathsf{LSamp}(n, r))$$

with error at most $O(2^{-n^\delta})$ when r is chosen uniformly at random.

Proof. The average-case algorithm is simple to describe. Given an input (n, r), pick an efficiently computable oracle $\mathcal{O}$ (we describe how below) and output NO if $u^{\mathcal{O}}(r) > k$ and output YES otherwise, where we set

$$k = n^{2\delta} s \log s.$$

We now specify how to pick the oracle $\mathcal{O}$. Let $\mathcal{A}'$ be the algorithm in Proposition 57. We cannot just use $\mathcal{A}'$ as our oracle $\mathcal{O}$ because it does not give deterministic answers to queries ($\mathcal{A}'$ is randomized). So instead, we mimic Adleman's construction to get an oracle that is deterministic and (with high probability) computes the approximation we desire. In other words, we will choose our oracle probabilistically, after we make this choice, the output of the oracle is deterministic. In detail, we sample $s \leftarrow \{0,1\}^{\mathrm{poly}(n)}$ and let $\mathcal{O}$ be the algorithm that runs $\mathcal{A}'$ and uses s for its randomness. In this case, the output of $\mathcal{O}$ is deterministic and because the failure probability of $\mathcal{A}'$ is at most $2^{-(n+m)^2}$, a union bound implies that, with all but exponentially small probability, we have that $\mathcal{O}$ computes

$$\Pr_{\tilde{r} \leftarrow \rho}[\mathsf{LSamp}(n, i, \tilde{r}) = T(i)]$$

with .01 additive error for all $\tilde{r} \in \{0,1,\star\}^m, T \in \{0,1\}^n$, and $i \in [n]$. Thus, in our proof of correctness, we can assume that our choice of $\mathcal{O}$ does indeed compute

$$\Pr_{\tilde{r} \leftarrow \rho}[\mathsf{LSamp}(n, i, \tilde{r}) = T(i)]$$

with .01 additive error for all $\tilde{r} \in \{0,1,\star\}^m, T \in \{0,1\}^n$, and $i \in [n]$. This completes our description of our algorithm. (The chance that this does not occur can be absorbed into the failure probability of our algorithm.)

It is easy to see that this algorithm runs in probabilistic polynomial time since $\mathcal{A}'$ (and hence $\mathcal{O}$) is computable in probabilistic polynomial time and the loop defining $u^{\mathcal{O}}$ is efficient given oracle access to $\mathcal{O}$. It remains to show correctness.

First, we show this algorithm always rejects NO instances. If $u^{\mathcal{O}}(r) \le k$, then Lemma 56 implies $\mathsf{CC}(\mathsf{LSamp}(n, r)) = O(kn^{3\delta}) = O(n^{5\delta} s \log s)$. So, this algorithm rejects all NO instances when n is sufficiently large.

Next, we show this algorithm usually accepts YES instances. We prove this by applying Lemma 55 and union bounding over the number of truth tables with circuit complexity at most s:

$$\begin{aligned}
&\Pr_{r \leftarrow \{0,1\}^m}[\mathsf{CC}(\mathsf{LSamp}(n, r)) \le s \text{ and } u^{\mathcal{O}}(r) > k] \\
\le &\sum_{T \in \{0,1\}^n : \mathsf{CC}(T) \le s} \Pr_{r \leftarrow \{0,1\}^m}[\mathsf{LSamp}(n, r) = T \text{ and } u^{\mathcal{O}}(r) > k] \\
\le &\sum_{T \in \{0,1\}^n : \mathsf{CC}(T) \le s} (4/5)^k \\
\le &\, 2^{O(s \log(n+s))}(4/5)^k \\
\le &\, O(2^{-n^\delta})
\end{aligned}$$

□

As a corollary, we get that, under certain parameters, if L has a local reduction to $\mathsf{GapMCSP}$, then solving L on any locally samplable distribution is easy on average.

Corollary 59. *Assume there is a $(n^{\delta/2})$-local reduction from a language L to* $\mathsf{GapMCSP}[s, \omega(n^{3\delta} s \log s)]$. *Let* LSamp *be a $(n^{\delta/2})$-locally samplable distribution. Then there is a randomized polynomial-time algorithm solving L on* LSamp *with error at most $2^{-n^{\Omega(1)}}$.*

Proof. Compose the $(n^{\delta/2})$-local reduction from L to $\mathsf{GapMCSP}[s, \omega(n^{3\delta} s \log s)]$ with the $(n^{\delta/2})$-local sampling algorithm for LSamp. This gives a new (n^{δ})-locally samplable distribution LSamp' where

$$L(\mathsf{LSamp}(n, r)) = \mathsf{GapMCSP}[s, \omega(n^{3\delta} s \log s)](\mathsf{LSamp}'(\mathrm{poly}(n), r)).$$

Then the Corollary follows from Theorem 58. □

Conversely, setting $L = \mathsf{SAT}$, we get that if SAT is hard under any locally samplable distribution, then SAT does not locally reduce to $\mathsf{GapMCSP}$ (with certain parameters) even if one allows for randomness.

Corollary 60. *If* SAT *is hard on some $(n^{\delta/2})$-locally samplable distribution, then there is no randomized (n^{δ})-local reduction from* SAT *to* $\mathsf{GapMCSP}[s, \omega(n^{3\delta} s \log s)]$

Finally, we justify the assumption of Corollary 60. In particular, if there exists a one-way function computable in NC^0, then SAT is average-case hard on some $O(\log n)$-locally samplable distribution. As a corollary, under plausible assumptions, there is no randomized $n^{0.01}$-local reduction from SAT to $\mathsf{GapMCSP}[n^{0.2}, n^{0.8}]$.

Theorem 61. *Suppose there are one-way functions computable in NC^0. Then there is an $O(\log n)$-locally samplable distribution $\mathcal{D}$ such that for every polynomial p, no PPT algorithm solves* SAT *with probability $1/2 + 1/p(n)$. (Actually, the local sampler only makes $O(1)$ queries to the random tape; it spends $O(\log n)$ time to output each bit of the* SAT *instance.)*

Proof Sketch. If there are one-way functions computable in NC^0, then there are also PRGs computable in NC^0 [HRV13]. Let $G : \{0,1\}^{n-\sqrt{n}} \to \{0,1\}^n$ be a PRG computable in NC^0_d, i.e. each output bit of G only depends on d input bits, where d is an absolute constant. Let $\mathcal{D}_y$ be the following distribution over length-n strings: w.p. $1/2$, we sample a uniformly random length-n string, and w.p. $1/2$, we sample a random output of G. Let $\mathcal{D}$ be the following distribution: we sample $y \leftarrow \mathcal{D}_y$, and encode the following assertion

$$\text{“}\exists s \in \{0,1\}^{n-\sqrt{n}}, \text{s.t. } G(s) = y\text{”}$$

into a SAT instance. We can verify that $\mathcal{D}_y$ is $O(d \log n)$-samplable, and the pseudorandomness of G implies that SAT is average-case hard on $\mathcal{D}_y$. □

6 Acknowledgments

We thank Igor Carboni Oliveira for collaborating on this project at an early stage, and Shuichi Hirahara and Ryan Williams for useful discussions.

References

[ACM+21] Eric Allender, Mahdi Cheraghchi, Dimitrios Myrisiotis, Harsha Tirumala, and Ilya Volkovich. One-way functions and a conditional variant of MKTP. *Electron. Colloquium Comput. Complex.*, 28:9, 2021. URL: https://eccc.weizmann.ac.il/report/2021/009. 2, 9

[AD97] Miklós Ajtai and Cynthia Dwork. A public-key cryptosystem with worst-case/average-case equivalence. In Frank Thomson Leighton and Peter W. Shor, editors, *Proceedings of the Twenty-Ninth Annual ACM Symposium on the Theory of Computing, El Paso, Texas, USA, May 4-6, 1997*, pages 284–293. ACM, 1997. doi:10.1145/258533.258604. 1

[Adl78] Leonard Adleman. Two theorems on random polynomial time. In *Proceedings of the 19th Annual Symposium on Foundations of Computer Science*, SFCS '78, page 75–83, USA, 1978. IEEE Computer Society. doi:10.1109/SFCS.1978.37. 10

[AGGM06] Adi Akavia, Oded Goldreich, Shafi Goldwasser, and Dana Moshkovitz. On basing one-way functions on NP-hardness. In *Proceedings of the 38th Annual ACM Symposium on Theory of Computing, Seattle, WA, USA, May 21-23, 2006*, pages 701–710. ACM, 2006. doi:10.1145/1132516.1132614. 1

[AIK06] Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. Cryptography in NC^0. *SIAM J. Comput.*, 36(4):845–888, 2006. doi:10.1137/S0097539705446950. 4, 7, 15, 24, 26

[AKS98] Noga Alon, Michael Krivelevich, and Benny Sudakov. Finding a large hidden clique in a random graph. In *Proceedings of the Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, 25-27 January 1998, San Francisco, California, USA*, pages 594–598. ACM/SIAM, 1998. URL: http://dl.acm.org/citation.cfm?id=314613.315014. 5, 29

[Ale11] Michael Alekhnovich. More on average case vs approximation complexity. *Comput. Complex.*, 20(4):755–786, 2011. doi:10.1007/s00037-011-0029-x. 1

[All01] Eric Allender. When worlds collide: Derandomization, lower bounds, and Kolmogorov complexity. In *FST TCS 2001: Foundations of Software Technology and Theoretical Computer Science, 21st Conference, Bangalore, India, December 13-15, 2001, Proceedings*, volume 2245 of *Lecture Notes in Computer Science*, pages 1–15. Springer, 2001. doi:10.1007/3-540-45294-X_1. 7, 11

[App13] Benny Applebaum. Pseudorandom generators with long stretch and low locality from random local one-way functions. *SIAM J. Comput.*, 42(5):2008–2037, 2013. doi:10.1137/120884857. 5, 30

[App16] Benny Applebaum. Cryptographic hardness of random local functions - survey. *Comput. Complex.*, 25(3):667–722, 2016. doi:10.1007/s00037-015-0121-8. 5, 30

[BB15] Andrej Bogdanov and Christina Brzuska. On basing size-verifiable one-way functions on NP-hardness. In *Theory of Cryptography - 12th Theory of Cryptography Conference, TCC 2015, Warsaw, Poland, March 23-25, 2015, Proceedings, Part I*, volume 9014 of *Lecture Notes in Computer Science*, pages 1–6. Springer, 2015. doi:10.1007/978-3-662-46494-6_1. 1

[BFKL93] Avrim Blum, Merrick L. Furst, Michael J. Kearns, and Richard J. Lipton. Cryptographic primitives based on hard learning problems. In *Advances in Cryptology - CRYPTO '93, 13th Annual International Cryptology Conference, Santa Barbara, California, USA, August 22-26, 1993, Proceedings*, volume 773 of *Lecture Notes in Computer Science*, pages 278–291. Springer, 1993. doi:10.1007/3-540-48329-2_24. 8

[BGG$^+$88] Michael Ben-Or, Oded Goldreich, Shafi Goldwasser, Johan Håstad, Joe Kilian, Silvio Micali, and Phillip Rogaway. Everything provable is provable in zero-knowledge. In *CRYPTO '88*, volume 403 of *Lecture Notes in Computer Science*, pages 37–56. Springer, 1988. doi:10.1007/0-387-34799-2_4. 8, 31

[BM84] Manuel Blum and Silvio Micali. How to generate cryptographically strong sequences of pseudo-random bits. *SIAM J. Comput.*, 13(4):850–864, 1984. doi:10.1137/0213053. 1

[BT06] Andrej Bogdanov and Luca Trevisan. On worst-case to average-case reductions for NP problems. *SIAM J. Comput.*, 36(4):1119–1159, 2006. doi:10.1137/S0097539705446974. 1

[CHO$^+$20] Lijie Chen, Shuichi Hirahara, Igor Carboni Oliveira, Ján Pich, Ninad Rajgopal, and Rahul Santhanam. Beyond natural proofs: Hardness magnification and locality. In *11th Innovations in Theoretical Computer Science Conference, ITCS 2020, January 12-14, 2020, Seattle, Washington, USA*, volume 151 of *LIPIcs*, pages 70:1–70:48. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.ITCS.2020.70. 2

[CIKK16] Marco L. Carmosino, Russell Impagliazzo, Valentine Kabanets, and Antonina Kolokolova. Learning algorithms from natural proofs. In *31st Conference on Computational Complexity, CCC 2016, May 29 to June 1, 2016, Tokyo, Japan*, volume 50 of *LIPIcs*, pages 10:1–10:24. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016. doi:10.4230/LIPIcs.CCC.2016.10. 2

[CJW19] Lijie Chen, Ce Jin, and R. Ryan Williams. Hardness magnification for all sparse NP languages. In David Zuckerman, editor, *60th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2019, Baltimore, Maryland, USA, November 9-12, 2019*, pages 1240–1255. IEEE Computer Society, 2019. doi:10.1109/FOCS.2019.00077. 2

[CJW20] Lijie Chen, Ce Jin, and R. Ryan Williams. Sharp threshold results for computational complexity. In Konstantin Makarychev, Yury Makarychev, Madhur Tulsiani, Gautam Kamath, and Julia Chuzhoy, editors, *Proccedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing, STOC 2020, Chicago, IL, USA, June 22-26, 2020*, pages 1335–1348. ACM, 2020. doi:10.1145/3357713.3384283. 2

[Coo71] Stephen A. Cook. The complexity of theorem-proving procedures. In *Proceedings of the 3rd Annual ACM Symposium on Theory of Computing, May 3-5, 1971, Shaker Heights, Ohio, USA*, pages 151–158. ACM, 1971. doi:10.1145/800157.805047. 2

[FK03] Uriel Feige and Robert Krauthgamer. The probable value of the Lovász–Schrijver relaxations for maximum independent set. *SIAM J. Comput.*, 32(2):345–370, 2003. doi:10.1137/S009753970240118X. 29

[FKS82] Michael L. Fredman, János Komlós, and Endre Szemerédi. Storing a sparse table with $O(1)$ worst case access time. In *23rd Annual Symposium on Foundations of Computer Science, Chicago, Illinois, USA, 3-5 November 1982*, pages 165–169. IEEE Computer Society, 1982. doi:10.1109/SFCS.1982.39. 22

[GGM86] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *J. ACM*, 33(4):792–807, 1986. doi:10.1145/6490.6503. 1, 4, 6, 23, 24

[GM84] Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *J. Comput. Syst. Sci.*, 28(2):270–299, 1984. doi:10.1016/0022-0000(84)90070-9. 1

[Gol00] Oded Goldreich. Candidate one-way functions based on expander graphs. *Electron. Colloquium Comput. Complex.*, 7(90), 2000. URL: http://eccc.hpi-web.de/eccc-reports/2000/TR00-090/index.html. 5, 30

[Gol01] Oded Goldreich. *The Foundations of Cryptography - Volume 1: Basic Techniques*. Cambridge University Press, 2001. doi:10.1017/CBO9780511546891. 1, 10, 23

[Har83] Juris Hartmanis. Generalized Kolmogorov complexity and the structure of feasible computations (preliminary report). In *24th Annual Symposium on Foundations of Computer Science, Tucson, Arizona, USA, 7-9 November 1983*, pages 439–445. IEEE Computer Society, 1983. doi:10.1109/SFCS.1983.21. 2

[HILL99] Johan Håstad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. A pseudorandom generator from any one-way function. *SIAM J. Comput.*, 28(4):1364–1396, 1999. doi:10.1137/S0097539793244708. 1, 6, 7, 23, 24, 31

[Hir18] Shuichi Hirahara. Non-black-box worst-case to average-case reductions within NP. In *59th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2018, Paris, France, October 7-9, 2018*, pages 247–258. IEEE Computer Society, 2018. doi:10.1109/FOCS.2018.00032. 2

[Hir20] Shuichi Hirahara. Characterizing average-case complexity of PH by worst-case meta-complexity. In *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020, Durham, NC, USA, November 16-19, 2020*, pages 50–60. IEEE, 2020. doi:10.1109/FOCS46700.2020.00014. 2

[Hir21] Shuichi Hirahara. Average-case hardness of NP from exponential worst-case hardness assumptions. In *STOC 2021, to appear*, 2021. URL: https://eccc.weizmann.ac.il/report/2021/058. 2

[HRV13] 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. doi:10.1137/100814421. 7, 23, 24, 35

[HS17] Shuichi Hirahara and Rahul Santhanam. On the average-case complexity of MCSP and its variants. In *32nd Computational Complexity Conference, CCC 2017, July 6-9, 2017, Riga, Latvia*, volume 79 of *LIPIcs*, pages 7:1–7:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017. doi:10.4230/LIPIcs.CCC.2017.7. 7

[IL89] Russell Impagliazzo and Michael Luby. One-way functions are essential for complexity based cryptography (extended abstract). In *30th Annual Symposium on Foundations of Computer Science, Research Triangle Park, North Carolina, USA, 30 October - 1 November 1989*, pages 230–235. IEEE Computer Society, 1989. doi:10.1109/SFCS.1989.63483. 7, 14

[IL90] Russell Impagliazzo and Leonid A. Levin. No better ways to generate hard NP instances than picking uniformly at random. In *31st Annual Symposium on Foundations of Computer Science, St. Louis, Missouri, USA, October 22-24, 1990, Volume II*, pages 812–821. IEEE Computer Society, 1990. doi:10.1109/FSCS.1990.89604. 4, 7, 8, 14

[Ila20] Rahul Ilango. Approaching MCSP from above and below: Hardness for a conditional variant and $\mathsf{AC}^0[p]$. In *11th Innovations in Theoretical Computer Science Conference, ITCS 2020, January 12-14, 2020, Seattle, Washington, USA*, volume 151 of *LIPIcs*, pages 34:1–34:26. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.ITCS.2020.34. 32

[ILL89] R. Impagliazzo, L. A. Levin, and M. Luby. Pseudo-random generation from one-way functions. In *Proceedings of the Twenty-First Annual ACM Symposium on Theory of Computing*, STOC '89, page 12–24, New York, NY, USA, 1989. Association for Computing Machinery. doi:10.1145/73007.73009. 11

[ILO20] Rahul Ilango, Bruno Loff, and Igor Carboni Oliveira. NP-hardness of circuit minimization for multi-output functions. In *35th Computational Complexity Conference, CCC 2020, July 28-31, 2020, Saarbrücken, Germany (Virtual Conference)*, volume 169 of *LIPIcs*, pages 22:1–22:36. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.CCC.2020.22. 32

[IN96] Russell Impagliazzo and Moni Naor. Efficient cryptographic schemes provably as secure as subset sum. *J. Cryptol.*, 9(4):199–216, 1996. doi:10.1007/BF00189260. 1

[IW97] Russell Impagliazzo and Avi Wigderson. $\mathsf{P} = \mathsf{BPP}$ if E requires exponential circuits: Derandomizing the XOR lemma. In *Proceedings of the Twenty-Ninth Annual ACM Symposium on Theory of Computing*, STOC '97, page 220–229, New York, NY, USA, 1997. Association for Computing Machinery. doi:10.1145/258533.258590. 16

[IY87] Russell Impagliazzo and Moti Yung. Direct minimum-knowledge computations. In *CRYPTO '87*, volume 293 of *Lecture Notes in Computer Science*, pages 40–51. Springer, 1987. doi:10.1007/3-540-48184-2_4. 31

[Jer92] Mark Jerrum. Large cliques elude the Metropolis process. *Random Struct. Algorithms*, 3(4):347–360, 1992. doi:10.1002/rsa.3240030402. 5, 29

[JP00] Ari Juels and Marcus Peinado. Hiding cliques for cryptographic security. *Des. Codes Cryptogr.*, 20(3):269–280, 2000. doi:10.1023/A:1008374125234. 1, 29

[Kar72] Richard M. Karp. Reducibility among combinatorial problems. In *Proceedings of a symposium on the Complexity of Computer Computations, held March 20-22, 1972, at the IBM Thomas J. Watson Research Center, Yorktown Heights, New York, USA*, The IBM Research Symposia Series, pages 85–103. Plenum Press, New York, 1972. doi:10.1007/978-1-4684-2001-2_9. 2

[KC00] Valentine Kabanets and Jin-Yi Cai. Circuit minimization problem. In *Proceedings of the Thirty-Second Annual ACM Symposium on Theory of Computing, May 21-23, 2000, Portland, OR, USA*, pages 73–79. ACM, 2000. doi:10.1145/335305.335314. 2

[Ko86] Ker-I Ko. On the notion of infinite pseudorandom sequences. *Theor. Comput. Sci.*, 48(3):9–33, 1986. doi:10.1016/0304-3975(86)90081-2. 2

[Ko91] Ker-I Ko. On the complexity of learning minimum time-bounded Turing machines. *SIAM J. Comput.*, 20(5):962–986, 1991. doi:10.1137/0220059. 2, 11

[Kuc95] Ludek Kucera. Expected complexity of graph partitioning problems. *Discret. Appl. Math.*, 57(2-3):193–212, 1995. doi:10.1016/0166-218X(94)00103-K. 5, 29

[Lev73] Leonid A. Levin. Universal sequential search problems. *Problemy peredachi informatsii*, 9(3):115–116, 1973. 2

[Lev84] Leonid A. Levin. Randomness conservation inequalities; information and independence in mathematical theories. *Inf. Control.*, 61(1):15–37, 1984. doi:10.1016/S0019-9958(84)80060-1. 7

[Lev86] Leonid A. Levin. Average case complete problems. *SIAM J. Comput.*, 15(1):285–286, 1986. doi:10.1137/0215020. 2

[Lev03] Leonid A. Levin. The tale of one-way functions. *Probl. Inf. Transm.*, 39(1):92–103, 2003. doi:10.1023/A\%3A1023634616182. 2

[LFKN92] Carsten Lund, Lance Fortnow, Howard J. Karloff, and Noam Nisan. Algebraic methods for interactive proof systems. *J. ACM*, 39(4):859–868, 1992. doi:10.1145/146585.146605. 31

[LO21] Zhenjian Lu and Igor Carboni Oliveira. An efficient coding theorem via probabilistic representations and its applications. In *ICALP 2021, to appear*, 2021. URL: https://eccc.weizmann.ac.il/report/2021/041. 7

[LP20] Yanyi Liu and Rafael Pass. On one-way functions and Kolmogorov complexity. In *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020, Durham, NC, USA, November 16-19, 2020*, pages 1243–1254. IEEE, 2020. doi:10.1109/FOCS46700.2020.00118. 2, 3, 4, 8

[LP21a] Yanyi Liu and Rafael Pass. Cryptography from sublinear-time average-case hardness of time-bounded Kolmogorov complexity. In *STOC 2021, to appear*, 2021. URL: https://eccc.weizmann.ac.il/report/2021/055. 2, 3

[LP21b] Yanyi Liu and Rafael Pass. On one-way functions from NP-complete problems. *Electron. Colloquium Comput. Complex.*, 28:59, 2021. URL: https://eccc.weizmann.ac.il/report/2021/059. 9

[LP21c] Yanyi Liu and Rafael Pass. On the possibility of basing cryptography on $\mathsf{EXP} \neq \mathsf{BPP}$. In *CRYPTO 2021, to appear*, 2021. URL: https://eccc.weizmann.ac.il/report/2021/056. 8

[MMW19] Dylan M. McKay, Cody D. Murray, and R. Ryan Williams. Weak lower bounds on resource-bounded compression imply strong separations of complexity classes. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC 2019, Phoenix, AZ, USA, June 23-26, 2019*, pages 1215–1225. ACM, 2019. doi:10.1145/3313276.3316396. 2

[MPW15] Raghu Meka, Aaron Potechin, and Avi Wigderson. Sum-of-squares lower bounds for planted clique. In Rocco A. Servedio and Ronitt Rubinfeld, editors, *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015, Portland, OR, USA, June 14-17, 2015*, pages 87–96. ACM, 2015. doi:10.1145/2746539.2746600. 29

[MW17] Cody D. Murray and R. Ryan Williams. On the (non) NP-hardness of computing circuit complexity. *Theory Comput.*, 13(1):1–22, 2017. doi:10.4086/toc.2017.v013a004. 6, 32

[Nao91] Moni Naor. Bit commitment using pseudorandomness. *J. Cryptol.*, 4(2):151–158, 1991. doi:10.1007/BF00196774. 31

[Oli19] Igor Carboni Oliveira. Randomness and intractability in Kolmogorov complexity. In *46th International Colloquium on Automata, Languages, and Programming, ICALP 2019, July 9-12, 2019, Patras, Greece*, volume 132 of *LIPIcs*, pages 32:1–32:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPIcs.ICALP.2019.32. 7

[OPS19] Igor Carboni Oliveira, Ján Pich, and Rahul Santhanam. Hardness magnification near state-of-the-art lower bounds. In *34th Computational Complexity Conference, CCC 2019, July 18-20, 2019, New Brunswick, NJ, USA*, volume 137 of *LIPIcs*, pages 27:1–27:29. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPIcs.CCC.2019.27. 2

[OS17] Igor Carboni Oliveira and Rahul Santhanam. Conspiracies between learning algorithms, circuit lower bounds, and pseudorandomness. In *32nd Computational Complexity Conference, CCC 2017, July 6-9, 2017, Riga, Latvia*, volume 79 of *LIPIcs*, pages 18:1–18:49. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017. doi:10.4230/LIPIcs.CCC.2017.18. 2, 8

[OS18] Igor Carboni Oliveira and Rahul Santhanam. Hardness magnification for natural problems. In *59th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2018, Paris, France, October 7-9, 2018*, pages 65–76. IEEE Computer Society, 2018. doi:10.1109/FOCS.2018.00016. 2

[PS19] Ján Pich and Rahul Santhanam. Why are proof complexity lower bounds hard? In *60th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2019, Baltimore, Maryland, USA, November 9-12, 2019*, pages 1305–1324. IEEE Computer Society, 2019. doi:10.1109/FOCS.2019.00080. 2

[RR97] Alexander A. Razborov and Steven Rudich. Natural proofs. *J. Comput. Syst. Sci.*, 55(1):24–35, 1997. doi:10.1006/jcss.1997.1494. 4

[RS21] Hanlin Ren and Rahul Santhanam. Hardness of KT characterizes parallel cryptography. In *CCC 2021, to appear*, 2021. URL: https://eccc.weizmann.ac.il/report/2021/057. 2, 8

[San20] Rahul Santhanam. Pseudorandomness and the minimum circuit size problem. In *11th Innovations in Theoretical Computer Science Conference, ITCS 2020, January 12-14, 2020, Seattle, Washington, USA*, volume 151 of *LIPIcs*, pages 68:1–68:26. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.ITCS.2020.68. 2, 4, 8

[Sha92] Adi Shamir. $\mathsf{IP} = \mathsf{PSPACE}$. *J. ACM*, 39(4):869–877, 1992. doi:10.1145/146585.146609. 8, 31

[Sip83] Michael Sipser. A complexity theoretic approach to randomness. In *Proceedings of the 15th Annual ACM Symposium on Theory of Computing, 25-27 April, 1983, Boston, Massachusetts, USA*, pages 330–335. ACM, 1983. doi:10.1145/800061.808762. 2

[Tra84] Boris A. Trakhtenbrot. A survey of Russian approaches to perebor (brute-force searches) algorithms. *IEEE Ann. Hist. Comput.*, 6(4):384–400, 1984. doi:10.1109/MAHC.1984.10036. 2

[Vad06] Salil P. Vadhan. An unconditional study of computational zero knowledge. *SIAM J. Comput.*, 36(4):1160–1214, 2006. doi:10.1137/S0097539705447207. 30

[Vad12] Salil P. Vadhan. Pseudorandomness. *Found. Trends Theor. Comput. Sci.*, 7(1-3):1–336, 2012. doi:10.1561/0400000010. 11

[Wil16] R. Ryan Williams. Natural proofs versus derandomization. *SIAM J. Comput.*, 45(2):497–529, 2016. doi:10.1137/130938219. 2

[Yao82] Andrew Chi-Chih Yao. Theory and applications of trapdoor functions (extended abstract). In *23rd Annual Symposium on Foundations of Computer Science, Chicago, Illinois, USA, 3-5 November 1982*, pages 80–91. IEEE Computer Society, 1982. doi:10.1109/SFCS.1982.45. 1, 3, 10

A Unconditional Hardness of GapK

Recall that Theorem 15 presents an algorithm for $\mathsf{GapK}[n-\Delta, n]$ with error probability $2^{-\Delta/3}$. We show that such algorithms have some limits, in the sense that even if we allow an arbitrarily large (but finite) running time, no algorithm could improve the error parameter to $o(2^{-\Delta}/n)$.

Theorem 62. *No algorithm computes* $\mathsf{GapK}[n-\Delta, n]$ *on the uniform distribution* $\mathcal{D} = \{U_n\}$ *with error at most* $o(2^{-\Delta}/n)$.

Proof. For contradiction, suppose there is an algorithm $\mathcal{A}$ that solves $\mathsf{GapK}[n-\Delta, n]$ on the uniform distribution with error probability $o(2^{-\Delta}/n)$. Let $x_1, x_2, \ldots, x_\ell$ denote the strings of length n that are rejected by $\mathcal{A}$ in lexicographic order. Let i be the smallest index such that $\mathsf{K}(x_i) \geq n-\Delta$. Since $\mathcal{A}(x)$ fails with probability $o(2^{-\Delta}/n)$, we know that $i \leq o(2^{n-\Delta}/n)$. Since we can describe y by revealing i, n and the code of $\mathcal{A}$, we get that

$$\mathsf{K}(y) \leq \log n + \log i + O(1) \leq n - \Delta - \omega(1),$$

which is a contradiction. $\square$