Electronic Colloquium on Computational Complexity, Report No. 165 (2023)

SAT Reduces to the Minimum Circuit Size Problem with a Random Oracle

Rahul Ilango
MIT
rilango@mit.edu

Abstract

The Minimum Circuit Size Problem (MCSP) asks, given the truth table of a Boolean function f and an integer s, if there is a circuit computing f of size at most s. It has been an open question since Levin's seminal work on NP-completeness whether MCSP is NP-complete. This question has drawn further interest in light of connections between MCSP and other areas, such as learning theory, average-case complexity, cryptography, and proof complexity. These connections arise from seemingly special properties of MCSP that are not currently known for any NP-complete problem.

We give, in our view, the strongest evidence yet that MCSP is in fact NP-complete. Specifically, we show that, with probability one, there is a P/poly reduction from the NP-hard problem of approximating vertex cover on hypergraphs to MCSP on circuits that have access to a uniformly random oracle $\mathcal{O}$ (the reduction can be made uniform if it is given access to $\mathcal{O}$). This resolves an open question of Huang, Ilango, and Ren (STOC 2023), who conjectured such a reduction exists. Curiously, a key part of our reduction is computing a cryptographic proof of work.

Our reduction yields near-optimal additive hardness of approximation and extends to computing time-bounded Kolmogorov complexity (K^t). Heuristically "instantiating" $\mathcal{O}$ with real-world cryptographic hash functions, we get a plethora of candidate uniform deterministic polynomial-time many-one reductions from SAT to MCSP and K^t in the standard unrelativized world. To our knowledge, no candidate reduction from SAT to MCSP or K^t was known previously. Moreover, our results hold in the regime where K^t has a non-black-box worst-case to average-case reduction (Hirahara, FOCS 2018). Thus, intriguingly, the existence of sufficiently "unstructured" functions implies a problem with a known (non-black-box) worst-case to average-case reduction is NP-complete.

ISSN 1433-8092

Contents

1 Introduction

The Minimum Circuit Size Problem (MCSP) [KC00] is the following computational task:

- **Given:** a Boolean function $f : \{0,1\}^n \to \{0,1\}$ (represented by its truth table[1] of length 2^n) and a size parameter s
- **Decide:** if there is a Boolean circuit[2] that computes f of size at most s.

MCSP has a fascinating history, which we now informally discuss, both to give the reader context and because it is interesting in its own right. Theoretical study of MCSP first began in the Soviet Union in the 1950s [Tra84]. At the time, researchers were interested in problems that required "perebor" (which roughly translates to "exhaustive search") to solve. They conjectured that MCSP was such a problem. (In fact, the conjecture was even once "explicitly claimed as a proven fact" [Tra84]. Had this proof been correct, it would have resolved $\mathsf{P} \neq \mathsf{NP}$ before the question was even formally asked!) At least so far, this conjecture has proved accurate. MCSP remains one of the few problems (another prominent example is Circuit Satisfiability) for which there are *no* known algorithms significantly improving over brute-force search.

Interest in MCSP continued in the 1970s. In a fascinating blog post [Lev], Levin recounts that when he first discovered the phenomenon of NP-completeness, he felt (at the time) that the problems he showed NP-complete (e.g., formula satisfiability, subgraph isomorphism, set cover) were of "narrow interest." Levin delayed publishing his results [Lev73] "for a long time," hoping to show a stronger result that included at least one "popular" problem, focusing on factoring, graph isomorphism, and MCSP. Over fifty years later, we now have evidence that factoring and graph isomorphism are unlikely[3] to NP-complete, but the question of whether MCSP is NP-complete remains open. Indeed, even basic facts about the complexity of MCSP are unknown. For example, it is consistent with current knowledge that the decision version of MCSP is solvable in *linear* time but the search version (where one needs to output an optimal circuit) cannot be solved *any* faster than brute-force search. Some researchers have speculated[4] that MCSP may be NP-intermediate (meaning that it is neither in P nor NP-complete).

The "modern" era of MCSP research began with the work of Kabanets and Cai in 2000 [KC00]. Prior to this, interest in MCSP was mainly motivated from two points of view: first, as a natural problem of practical importance (for building hardware), and second, as a seemingly intractable problem whose complexity was unknown. Beginning largely with the work of Kabanets and Cai, a new motivation emerged: MCSP has intriguing properties not known for any NP-complete problem.[5] Furthermore, these properties have led to connections with other areas of theoretical computer science, including average-case complexity, learning theory, proof complexity, and cryptography. We will give a brief, informal, and non-exhaustive survey of some of these developments in the next subsection. As a taste, using these connections one can, for example, connect the NP-completeness of MCSP (and related problems) to showing that $\mathsf{P} \neq \mathsf{NP}$ implies:

- NP is hard on average (i.e., giving a worst-case to average-case reduction for SAT).
- There is no "natural proof" of certain circuit lower bound statements.
- PAC-learning functions computable by small circuits using membership queries is intractable.
- Cryptographic one-way functions exist.[6]

[1] The truth table of f is the 2^n bit string $f(0^n) \dots f(1^n)$.

[2] To make this definition precise we need to specify a model of circuits and a specific notion of circuit size. Our results hold for most reasonable choices, but, for concreteness, consider the circuits with fan-in two AND and OR gates as well as NOT gates. The size of a circuit is the number of AND and OR gates.

[3] This is because factoring is in coNP and graph isomorphism is in quasi-polynomial time [Bab16].

[4] For example, MCSP is currently listed as a candidate NP-intermediate problem on Wikipedia.

[5] Of course, if MCSP turns out to be NP-complete, then many of these properties will carry over to every other NP-complete problem.

[6] In contrast to the other bullet points (where the connections are completely formal), this connection is still an active area of research, as we will discuss later.

1.1 Background: MCSP and Metacomplexity

MCSP is an example of a *metacomplexity* problem. In general, a metacomplexity problem asks one to determine the complexity of a given object according to some complexity measure. In the case of MCSP, this complexity measure is circuit complexity. There are several other important metacomplexity problems, such as the problem of computing time-bounded Kolmogorov complexity [Sip83; Ko86; Har83; Ko91]. For a Boolean string x, the *t-time-bounded Kolmogorov complexity* of x, denoted $\mathsf{K}^t(x)$, is the length of the shortest program that outputs x in time at most t (see Section 2 for a formal definition).

Metacomplexity and Learning Theory. Carmosino, Impagliazzo, Kabanets, and Kolokolova [CIKK16] show that average-case algorithms for MCSP on a restricted circuit class $\mathcal{C}$ imply learning algorithms for functions computable in $\mathcal{C}$. Using this connection, they give a PAC-learning algorithm for $\mathsf{AC}^0[p]$ on the uniform distribution with membership queries, resolving a longstanding open question in learning theory. Their learning algorithm relies on an average-case $\mathsf{AC}^0[p]$-MCSP algorithm given by Razborov and Rudich [RR97], which is based on the lower bounds methods of Razborov [Raz87] and Smolensky [Smo87]. Connections between metacomplexity and learning are explored in several other papers (e.g. [IL90; CIKK17; HN21; BCK+22]).

Metacomplexity and Cryptography. Kabanets and Cai [KC00] show that if $\mathsf{MCSP} \in \mathsf{P}$, then one-way functions do not exist. As a result, under a widely believed cryptographic assumption, MCSP is intractable.

A natural question to ask is whether a converse holds. In a breakthrough result, Liu and Pass [LP20] show that one-way functions exist if and only if computing time-bounded Kolmogorov complexity is mildly hard on average on the uniform distribution. Connections with the existence of one-way functions have been proved for several other metacomplexity problems [IL90; San20; ACM+21; LP21a; LP21b; RS21; IRS22; LP22; HIL+23; Hir23; LP23].

Metacomplexity and Average-Case Complexity. A basic open question in complexity theory is to show that if SAT is hard to solve in the worst-case, then SAT is also hard to solve on average. In the language of Impagliazzo's five worlds [Imp95], this corresponds to showing we do not live in "Heuristica." There are several barrier results (e.g. [FF93; BT06]) showing that to resolve this question one needs to use "non-black box" techniques. In an influential paper, Hirahara [Hir18] gives a *non-black box* worst-case to average-case reduction for approximating MCSP and K^t. Building on this metacomplexity approach, Hirahara [Hir21] shows that an exponential *worst-case* lower bound on UP implies the *average-case* hardness of NP. This is a very active area of research [San20; Hir20; HN21; HN22; HS22; Hir22b; GKLO22; GK22; CHV22].

Metacomplexity and Proof Complexity. A central question in proof complexity is understanding the difficulty of proving circuit lower bounds. The famous "natural proofs barrier" of Razborov and Rudich [RR97] is perhaps the best explanation we have so far of why proving circuit lower bounds is difficult. One can interpret Razborov and Rudich's result as saying that any "natural proof" of a circuit lower bound for a class $\mathcal{C}$ implies an average-case algorithm for solving MCSP on $\mathcal{C}$-circuits, a consequence that is considered unlikely even for relatively weak circuit classes. Moreover, using a worst-case to average-case reduction, Hirahara [Hir18] shows the worst-case hardness of approximating MCSP implies the non-existence of "natural proofs" of certain circuit lower bound statements.

Hardness of Metacomplexity: Progress and Barriers. The best known unconditional hardness for MCSP is a result by Allender and Das [AD17] showing that all problems in the cryptographically important class SZK (statistical zero knowledge) reduce to MCSP. We also know lower bounds for computing MCSP against weak circuit classes, almost matching the best known for explicit functions [GII+19; CKLM19; KKL+20; GR22]. These results are essentially all that is known (unconditionally) about the hardness of MCSP.

To explain the difficulty of showing hardness for MCSP, researchers have proved several barrier results. One line of research [KC00; MW17; HP15; SS20] shows that proving MCSP is NP-complete under certain types of reductions implies widely-believed, but seemingly difficult to show, lower bounds. For example, Murray and Williams [MW17] show that if MCSP is NP-complete under deterministic polynomial-time many-one reductions, then $\mathsf{EXP} \neq \mathsf{ZPP}$. We stress that it is widely believed that $\mathsf{EXP} \neq \mathsf{ZPP}$ is true, but, at least so far, it seems beyond the reach of current techniques to prove it.

Another line of research [HW16; AHK17; AH19; RS22] looks at relativization barriers and oracle versions of MCSP. For example, Ren and Santhanam [RS22] construct an oracle relative to which MCSP is in P but NP requires exponential time. Hirahara and Wantanabe [HW16] show, under a plausible assumption, that any "oracle independent" randomized many-one reduction cannot show that MCSP is NP-hard.

A third line of research constrains what properties reductions from SAT to MCSP can have. For example, Murray and Williams [MW17] unconditionally rule out the possibility of a "local" reduction from SAT to MCSP. Saks and Santhanam show that, under a plausible assumption, any randomized polynomial-time many-one reduction from SAT to approximating K^t needs to run in time that grows (essentially) polynomially with t.

Despite these barrier results, significant progress has been made on showing hardness for variations of MCSP. For example, we know NP-hardness for all the following problems:

- MCSP on DNF formulas [Mas79; Czo99; UVS06; AHM+08; Fel09; KS08]
- MCSP on circuits with $\mathcal{O}$-oracle gates where $\mathcal{O}$ is a PSPACE-complete problem [ABK+06][7]
- MCSP on DNF-XOR formulas [HOS18]
- Conditional string versions (i.e., corresponding to the following problem: given two strings x and y, how hard is it to compute x given access to y) of MCSP [Ila20a; HIR23], K^t [LP22; Hir22b; HIR23], and another complexity measure KT [ACM+21]
- MCSP for multi-output functions [ILO20]
- MCSP on constant-depth formulas [Ila20b] (under quasipolynomial-time reductions)
- MCSP on (unbounded) formulas [Ila21] (under subexponential-time reductions)
- Partial function versions (i.e., one is given a truth table of a *partial* function $f : \{0,1\}^n \to \{0,1,\star\}$) of MCSP [Hir22a; Ila20b][8] and K^t [Hir22a]

Before we go on to discuss our results, we discuss a subset of the prior work above that is especially relevant to this paper. First is the recent tour-de-force result by Hirahara [Hir22a]. Hirahara shows that the partial function versions of MCSP and K^t complexity are NP-hard under randomized polynomial-time reductions. Hirahara's tools include state-of-the-art hardness of approximation, secret sharing schemes for monotone access structures, and one-time information-theoretic encryptions.

In light of Hirahara's breakthrough result, a tantalizing possibility is that perhaps it is within reach to show, unconditionally, that MCSP (on total functions) is NP-hard under, say, polynomial-time randomized reductions. Unfortunately, Hirahara's proof crucially relies on using partial functions, and the seemingly minor leap from partial functions to total functions causes (at least for now) major problems. Indeed, there are even formal barriers. Hirahara's proof that the partial function version of computing K^t is NP-hard is "oracle independent." On the other hand, Hirahara and Wantanabe [HW16] show that a similar oracle independent reduction is not possible for the total function version.

A commonly studied [ABK+06; AHK17; AH19] version of MCSP is $\mathsf{MCSP}^{\mathcal{O}}$, where one considers circuits with $\mathcal{O}$-oracle gates.[9] For example, as mentioned previously, $\mathsf{MCSP}^{\mathsf{PSPACE}}$ is PSPACE-complete [ABK+06].

[7] Actually, this problem is even hard for PSPACE.

[8] [Ila20b] only showed NP-hardness under subexponential time reductions. [Hir22a] shows NP-hardness under polynomial time reductions.

[9] We note that the problem $\mathsf{MCSP}^{\mathcal{O}}$ is different from the related problem MOCSP [Ila20a] (Minimum Oracle Circuit Size Problem). In $\mathsf{MCSP}^{\mathcal{O}}$, the oracle is fixed in the definition of the problem while in MOCSP the oracle is an *input*.

A recent paper by Huang, Ilango, and Ren [HIR23] introduces the idea of studying the case where $\mathcal{O}$ is a *uniformly random oracle* (we will more carefully define what this means in the next subsection). Many conjectures in complexity theory are provable relative to a random oracle (e.g., $\mathsf{P} \neq \mathsf{NP}$ [BG81]), and this is often viewed as good evidence that the corresponding conjectures are true (with a few notable exceptions that we discuss in the next subsection).

Huang, Ilango, and Ren show that for a metacomplexity problem called mvMCSP, one can prove strong hardness of approximation for $\mathsf{mvMCSP}^{\mathcal{O}}$ when $\mathcal{O}$ is a random oracle. In more detail, mvMCSP asks, given a relation $R \subseteq \{0,1\}^n \times \{0,1\}^m$ what is the minimum circuit complexity of any function f such that $(x, f(x)) \in R$ for all x. Since partial functions are a special case of relations, Hirahara's [Hir22a] recent result also shows that mvMCSP is NP-hard unconditionally. Huang, Ilango, and Ren show that with probability one over a random oracle $\mathcal{O}$ that $\mathsf{3\text{-}SAT}$ reduces (under quasipolynoimal time $\mathcal{O}$-oracle reductions) to large approximations (size s versus s^k for all constants k) of $\mathsf{mvMCSP}^{\mathcal{O}}$. Intriguingly, this result follows almost directly from Micali's construction [Mic00] of succinct non-interactive arguments in the random oracle model.

Furthermore, they conjecture a similar statement holds for MCSP and that this "would give strong evidence that MCSP is indeed NP-complete" (we will discuss why in the next subsection).

Conjecture 1 (Huang, Ilango, and Ren [HIR23])**.** *With probability one over the choice of a random oracle $\mathcal{O}$, $\mathsf{3\text{-}SAT}$ reduces to $\mathsf{MCSP}^{\mathcal{O}}$ under quasipolynomial-time reductions with $\mathcal{O}$-oracle access.*

1.2 Our Results

Our main result is to prove Conjecture 1.[10] Specifically, our hardness results are proven in a model where we equip all circuits and Turing machines with oracle access to a uniformly random function[11] $\mathcal{O} : \{0,1\}^\star \to \{0,1\}$. This model is often referred to as the *Random Oracle Model* [BR93].

Informally, our main result is the following.

Theorem 2 (Informal)**.** *In the Random Oracle Model, there is a (black-box) uniform polynomial-time deterministic many-one reduction from an NP-hard variant of vertex cover to both MCSP and computing K^t. Moreover, these reductions give optimal (up to a constant factor) additive hardness of approximation.*

At the cost of introducing non-uniformity, we can state our result more succinctly (an analogous result also holds for K^t).

Theorem 3 (Informal)**.** *With probability one over a random oracle $\mathcal{O}$,*

$$\mathsf{NP} \subseteq \mathsf{P}^{\mathsf{MCSP}^{\mathcal{O}}}/\mathsf{poly}.$$

We believe that Theorem 2 and Theorem 3 are the strongest evidence yet that MCSP and K^t are in fact NP-complete. We discuss some aspects of this result before formally stating our theorems and defining $\mathsf{MCSP}^{\mathcal{O}}$.

The Reduction. We emphasize that the notion of reduction in Theorem 2 is the standard one: a uniform deterministic polynomial-time many-one reduction, with the caveat that the reduction gets oracle access to $\mathcal{O}$ (this caveat is removed in Theorem 3 using non-uniformity).

Our reduction is from approximating vertex cover on τ-uniform hypergraphs (which we reformulate[12] as a variant of set cover called Gap τ-Frequency Set Cover). Strong NP-hardness of approximation is known [DGKR05] for this problem even when τ is constant (which is our setting).

The reduction is the same for both MCSP and K^t but with a different size threshold. We state (a randomized version[13] of) the reduction in Theorem 2 below and give detailed intuition for each step in the reduction. In our view, a significant feature is that, although our proof of correctness is rather technical, the reduction itself is quite simple!

[10] In fact, our result is stronger than Conjecture 1 since we use polynomial-time reductions.

[11] Formally, this means that for all x the value of $\mathcal{O}(x)$ is independently sampled uniformly from $\{0,1\}$.

[12] This reformulation is not for any technical reason. We felt that thinking about set covers was more intuitive than hypergraphs.

[13] One can easily derandomize it using the random oracle.

Reduction from Gap τ-Frequency Set Cover to MCSP and K^t

Parameter: a power of two $\lambda = \Theta(\log n)$.

Oracles: For all powers of two n, we have oracle access to $\mathcal{O}_n : [n] \times \{0,1\}^\lambda \times \{0,1\}^{2\lambda} \to \{0,1\}^\lambda$.

Given: a set cover instance $S_1, \ldots, S_m \subseteq [n]$ with τ-frequency[a] where n and τ are powers of two.

1. Pick $sk_1, \ldots, sk_m \in \{0,1\}^\lambda$ uniformly at random.
   - Intuition: One can think of each sk_j as a "secret key" associated with a set in the set cover instance. These secret keys will be used in an "encryption scheme" later.
2. For all $i \in [n]$ pick $v_i \in \{0,1\}^\lambda$ uniformly at random.
   - Intuition: One can think of v_i as a uniformly random message (that we will later encrypt) associated with each element of the ground set $[n]$.
3. For all $i \in [n]$ and $k \in [\tau]$, let $c_{i,k}$ be a uniformly random element (chosen using rejection sampling) of the set $\{c \in \{0,1\}^{2\lambda} : \mathcal{O}_n(i, sk_j, c) = v_i\}$ where j is the index of the kth set containing i.
   - Intuition: This is the crucial part of the reduction. There are two things that are happening here. First, one can think of $c_{i,k}$ as an encryption of the message v_i using the secret key for the set S_j that contains i. (This allows one to decrypt encrypting the message v_i if one knows at least one secret key for a set that contains i, since $\mathcal{O}_n(i, sk_j, c_{i,k}) = v_i$.) The second (more subtle but critical) thing is that the specific encryption scheme we use has the property that the cipher texts are in some strong sense incompressible, *even if one knows the secret key sk_j and the value v_i it decrypts to.* In part, this is because the rejection sampling procedure is effectively doing an idealized cryptographic proof of work. It may not be clear why this property is important yet, but we dedicate the entirety of Section 1.3 to giving intuition about why we use this encryption scheme and why this property is useful. We also note that because $\lambda = O(\log n)$ the rejection sampling step runs in polynomial time with high probability.
4. Output the $4\tau n\lambda$-bit truth table of the function $f : [n] \times [\tau] \times \{0,1\} \times [2\lambda] \to \{0,1\}$ given by[b]

   $$f(i,k,b,d) = \begin{cases} d\text{'th bit of } c_{i,k}, & \text{if } b = 0 \\ d\text{'th bit of } v_i, & \text{if } b = 1 \end{cases}.$$

   We also output some complexity threshold that depends on the precise problem we are reducing to.
   - Intuition: the truth table of f is essentially the string that concatenates all the encryptions $c_{i,k}$ and all the messages v_i. Ideally, one might hope that the "optimal" way to compute f is to have memorized all the $c_{i,k}$ as well as the secret keys sk_j for all j in an optimal set cover. Then one can decrypt "on the fly" to find the values of v_i. This would relate the complexity of computing f to the size of the optimal set cover. Unfortunately, this bound fails badly in the information-theoretic setting: one can instead memorize all the v_i and save on describing each $c_{i,k}$ knowing that it decrypts to v_i. But, crucially, this may require a lot of time (i.e., queries to the oracle). In particular, this is where we need the incompressibility property of the $c_{i,k}$.

[a] This means that every $i \in [n]$ is contained in *exactly* τ sets.

[b] Since each v_i is a λ-bit string, and $d \in [2\lambda]$, we must adopt a convention for what the d'th bit of v_i is when $d > \lambda$. Any reasonable choice suffices, but for concreteness we say to output 0 when this is the case.

Is this good evidence MCSP *is* NP*-complete?* As mentioned previously, we believe our results are strong evidence that MCSP and K^t are indeed NP-complete. A skeptic might wonder why these results are compelling given that we know counterexamples [Kur83; Har83; CCG+94] to the random oracle hypothesis [BG81] (the hypothesis that structural results that hold relative to a random oracle hold in the unrelativized world). Of course, being "strong evidence" is an informal claim, so we cannot give a truly rigorous answer, but, still, we try to illustrate three aspects of our thinking.

1. First, we address the known counterexamples to the random oracle hypothesis.

   Arguably, the most compelling counterexample is that $\mathsf{IP} = \mathsf{PSPACE}$ [Sha92] but $\mathsf{IP}^{\mathcal{O}} \neq \mathsf{PSPACE}^{\mathcal{O}}$ for a random oracle $\mathcal{O}$ [CCG+94]. Note that IP and PSPACE are two *equal* classes in the unrelativized world that become *unequal* in the random oracle model. In contrast, to the best of our knowledge, there is no known example of inclusions in the random oracle model which fail in the unrelativized model. Our result, $\mathsf{NP} \subseteq \mathsf{P}^{\mathsf{MCSP}^{\mathcal{O}}}/\mathsf{poly}$, is an inclusion.

   Moreover, in our inclusion, the random oracle $\mathcal{O}$ is only given to the class we are lower bounding. It is easy to see that the IP versus PSPACE counterexample fails in this setting, since $\mathsf{IP} = \mathsf{PSPACE}$ immediately implies that $\mathsf{PSPACE} \subseteq \mathsf{IP}^{\mathcal{O}}$ and $\mathsf{IP} \subseteq \mathsf{PSPACE}^{\mathcal{O}}$.

   Next, as argued in [KMR95], perhaps the strongest reason for believing the random oracle heuristic is the idea that sufficiently "pseudorandom" cryptographic functions exist in P which mimic random oracles. This reasoning does not apply to extremely strong adversaries such as those corresponding to IP and PSPACE, who can easily break such pseudorandom functions.

   On the other hand, in our setting our adversaries are precisely those cryptographic functions are meant to be secure against: circuits of a bounded size. Moreover, this setting is well-studied: the random oracle methodology [BR93] in cryptography is widely used to give heuristic constructions of cryptography protocols. In particular, if a protocol is secure in the random oracle model, one can heuristically "instantiate" the protocol by replacing the oracle $\mathcal{O}$ with a real-world hash function, like a version of AES handling arbitrarily long inputs [BRS02; DR20].

   This paradigm is extremely successful in practice and is the basis for many real-world cryptographic systems. Unfortunately, it is known that there are protocols that are secure in the random oracle model, but are *insecure* when instantiated by *any* efficiently computable function [CGH04]. However, as noted by Bitansky, Kalai, and Paneth [BKP18] these examples are somewhat unsatisfying:

   > "*For several cryptographic hash functions used in practice, the only known separations from random oracles are highly contrived.*"

   It is a longstanding open question to give non-contrived separations between random oracles and hash functions used in practice. As a result, we get a win-win: either (as we suspect) our reduction can be instantiated by real-world hash functions, or it can be used to separate real-world hash functions from random oracles.

2. We feel instantiating $\mathcal{O}$ looks especially plausible. We give formal and informal reasons for this.

   Formally, we observe that the reduction has an extremely nice property: as one may be able to see from the pseudocode above, we only need to evaluate the oracle on inputs of length $O(\log n)$.[14] As a result, our reduction is very insensitive to the running time of the oracle $\mathcal{O}$. Because of our large additive hardness of approximation, this means randomized versions of our reductions hold even if $\mathcal{O}$ can only be evaluated on an input x in time, say, $O(2^{\epsilon|x|})$ for a sufficiently small ϵ. In fact, if one instead aims for a non-uniform reduction, it even suffices for the oracle to have a *non-uniform* circuit of size $O(2^{\epsilon|x|})$. Since *every* function has a circuit of size $O(2^{|x|})$, and our proof shows that most functions have the properties we need, this could make replacing the random oracle $\mathcal{O}$ with some concrete function significantly easier.

[14] Actually this is only true for randomized versions of our reductions. To also derandomize the reduction, one needs to evaluate the oracle on longer inputs.

Relatedly, because our reductions are allowed to be randomized or non-uniform, we have significantly more flexibility in instantiating $\mathcal{O}$ than is usually the case when using the random oracle methodology. For example, it is perfectly okay for the reduction to sample a random key k and set $\mathcal{O} = F_k$ where F_k is a pseudorandom function family. In contrast, usually when using the random oracle methodology one has to choose a single uniform oracle function $\mathcal{O}$ that works for all inputs.

Finally, informally, we feel the reduction is rather intuitive and uses the oracle $\mathcal{O}$ in a natural way. It seemingly lacks the circularity that can preclude instantiation.

3. It seems even easier to instantiate the reduction in Theorem 3. By utilizing non-uniformity, the reduction in Theorem 3 simplifies to just one line!

Non-Uniform Reduction from Gap τ-Frequency Set Cover to K^t and MCSP

Parameter: power of two $\lambda = \mathsf{poly}(n)$

Advice: hardcoded values $v_i \in \{0,1\}^\lambda$ and $c^\star_{i,j} \in \{0,1\}^{2\lambda}$ for all $i \in [n]$ and $j \in [m]$

Given: a τ-frequency set cover instance $S_1, \ldots, S_m \subseteq [n]$ where n and τ are powers of two

1. Output the complexity threshold[a] $2\lambda n\tau + n\lambda/8 - 1$ and the $4\tau n\lambda$-bit truth table of the function $f : [n] \times [\tau] \times \{0,1\} \times [2\lambda] \to \{0,1\}$ given by

$$f(i,k,b,d) = \begin{cases} d\text{'th bit of } c^\star_{i,j}, & \text{if } b = 0, \text{ where } S_j \text{ is the } k\text{'th set containing } i \\ d\text{'th bit of } v_i, & \text{if } b = 1 \end{cases}.$$

[a] For MCSP we use a different threshold.

Observe that, intriguingly, the circuit implementing this reduction does *not* need oracle access to $\mathcal{O}$. All the information it needs about $\mathcal{O}$ is non-uniformly provided in its polynomial bits of advice (oracle access to $\mathcal{O}$ is just needed for the reduction's correctness). Perhaps because of this, instantiating this non-uniform reduction seems even more plausible than instantiating our uniform reduction. "All one needs to do" is prove the existence of advice such that this reduction succeeds. Our work shows that such advice exists in the random oracle model. We see no reason why such advice should not exist in the unrelativized world.

How do we avoid the barrier results? We now discuss how our reduction overcomes the aforementioned barrier results. We roughly split the barrier results into three categories.

The first category is barrier results showing that any "natural" reduction from SAT to MCSP implies lower bounds, such as $\mathsf{EXP} \neq \mathsf{ZPP}$. Our use of the random oracle allows us to bypass these barriers. This is because, for example, relative to a random oracle EXP is indeed not equal to ZPP.[15]

The second category is related to relativization. This includes the usual relativization barriers (i.e., oracle worlds where $\mathsf{MCSP}^\mathcal{O}$ is easy but $\mathsf{NP}^\mathcal{O}$ is not) as well the limitations of oracle independent reductions (i.e., reductions that work from a language L to $\mathsf{MCSP}^\mathcal{O}$ for all oracles $\mathcal{O}$). We avoid these barriers because our results do not hold for *all* oracles $\mathcal{O}$, but instead only for *random* oracles $\mathcal{O}$.

The last category includes a result of Saks and Santhanam [SS22]. Saks and Santhanam show that, under a plausible assumption, any randomized polynomial-time many-one reduction from SAT to approximating K^t needs to run in time (roughly speaking) that grows polynomially with t. This essentially means that the reduction itself needs to be doing some roughly t-time hard task. In our reduction described above, this is accomplished in step 3, where the reduction essentially does a cryptographic proof of work! In particular, it finds a pre-image of a random string v_i under a hash function $(\mathcal{O}_n(i, sk_j, \cdot))$.

[15] In more detail, one can use the random oracle to show $\mathsf{ZPP}^\mathcal{O} = \mathsf{P}^\mathcal{O}$. On the other hand, $\mathsf{P}^\mathcal{O} \neq \mathsf{EXP}^\mathcal{O}$ because the time-hierarchy theorem relativizes.

Worst-case to Average-case Reductions. A basic open question in complexity theory is connecting the worst-case and average-case complexities of NP. Hirahara [Hir18] shows (roughly) that solving the problem of approximating K^t complexity up to an additive $O(\sqrt{n})$ in the worst-case reduces in a non-black-box way to solving K^t on average (with zero error) on the uniform distribution.

Thus, if one could show that this problem is NP-complete, NP has a worst-case to average-case reduction. Should we expect this problem to be NP-complete? An immediate corollary of Theorem 40 is that it is NP-complete in the random oracle model.

Corollary 4 (Informal)**.** *The approximation of K^t that Hirahara [Hir18] gives a worst-case to average-case reduction for is NP-complete in the random oracle model.*

We also get the following corollary about "instantiating" the random oracle.

Corollary 5 (Informal)**.** *There is a problem Π in NP with a known (non-black-box) worst-case to average-case reduction, such that if one can successfully "instantiate" the random oracle in Theorem 2, then Π is NP-complete.*

What we find most interesting about this result is that it connects the properties of (ideally) *unstructured* hash functions like SHA and AES to basic questions about the *structure* of NP (in particular, does Hirahara's worst-case to average-case apply to NP?).

It is worth noting that the order of the statement in Corollary 5 is important. If one could provably instantiate the reduction in Corollary 5, then one might (although perhaps not necessarily) also be able to prove that NP is hard on average. As a result, one could argue that SAT has a non-black-box worst-case to average-case reduction: this is because assuming SAT is easy on average is a contradiction, so you can prove any consequence.

Finally, a good comparison with Corollary 5 is the result by Huang, Ilango, and Ren [HIR23] showing that the conditional string version of K^t is NP-complete in the "superlinear regime" assuming a strong cryptographic primitive called witness encryption [GGSW13] exists. Hirahara [Hir22b] shows that this problem also has a non-black-box worst-case to average-case reduction.

1.2.1 Formal Theorem Statements

We now give a formal statement of Theorem 2 using the relativized versions of MCSP and K^t. The *$\mathcal{O}$-oracle circuit complexity* of a function f is the size of the smallest circuit that computes f where one allows AND, OR, NOT, and $\mathcal{O}$-oracle gates. Similarly, let $\mathsf{K}^{t,\mathcal{O}}(x)$ denote the size of the smallest description for outputting x in time at most t given oracle access to $\mathcal{O}$. (See Section 2 for formal definitions.)

We first state our theorem for circuit complexity.

Theorem 6. *There is a deterministic polynomial-time algorithm A such that with probability one (over a random oracle $\mathcal{O}$) $A^{\mathcal{O}}$ is a many-one reduction from $\mathsf{3\text{-}SAT}$ to the promise problem of, given a string x, outputting:*

- ***YES:*** *if the $\mathcal{O}$-oracle circuit complexity of x is at most $\theta(|x|)$ (and moreover, this is witnessed by a constant-depth circuit)*
- ***NO:*** *if the $\mathcal{O}$-oracle circuit complexity of x is at least $\theta(|x|) + \Omega(\frac{|x|}{\log |x|})$*

where θ is some function of $|x|$.

Now we state our theorem for time-bounded Kolmogorov complexity.

Theorem 7. *Let $p(\cdot)$ be any polynomial satisfying $p(t) \geq t$. There is a deterministic polynomial-time algorithm A such that with probability one (over a random oracle $\mathcal{O}$), $A^{\mathcal{O}}$ is a many-one reduction from $\mathsf{3\text{-}SAT}$ to the promise problem of, given a string x, outputting:*

- ***YES:*** *if $\mathsf{K}^{t,\mathcal{O}}(x) \leq \theta(|x|)$*

- ***NO:*** *if* $\mathsf{K}^{p(t),\mathcal{O}}(x) \geq \theta(|x|) + \Omega(|x|)$

where t *is some polynomial in* $|x|$ *and* θ *is some function of* $|x|$.

We discuss a few more aspects of our theorems.

Hardness of Approximation. We note that our additive hardness of approximation is within a constant factor of optimal. This is because for all strings x, $\mathsf{K}^t(x) \leq |x| + O(1)$ when $t \geq \mathsf{poly}(|x|)$. Similarly, the circuit complexity of any function $f : \{0,1\}^n \to \{0,1\}$ is at most $O(\frac{2^n}{n})$ which is $O(\frac{|T|}{\log |T|})$ where T is the truth table of f.

Hardness for NP *not* $\mathsf{NP}^{\mathcal{O}}$*.* Our proofs use non-relativizing properties of SAT (specifically, the PCP theorem). As a result, we do *not* show a reduction from $\mathsf{SAT}^{\mathcal{O}}$ to $\mathsf{MCSP}^{\mathcal{O}}$ or $\mathsf{K}^{t,\mathcal{O}}$. We instead show a reduction from (the usual) SAT to $\mathsf{MCSP}^{\mathcal{O}}$ or $\mathsf{K}^{t,\mathcal{O}}$.

A Simpler, Non-Uniform Reduction with Larger Time Gaps and without Oracle Access. One deficiency in Theorem 7 is that the gap in the time-bound from t to t' is only polynomial. At a high level, the reason for this is that the reduction needs to spend a lot of time finding "collision" in a hash function built from $\mathcal{O}$ (indeed, this is in some sense inherent because of the aforementioned result of Saks and Santhanam [SS22]). By instead considering *non-uniform* reductions and hardcoding in these collisions, we show NP-hardness for K^t with an exponential gap in the time bounds.

Theorem 8. *Let* $\epsilon > 0$ *be any sufficiently small constant. Let* $\mathcal{O}$ *be a uniformly random oracle. With probability one there is a non-uniform polynomial-time many-one reduction (where the circuit implementing the reduction does* not *get oracle access to* $\mathcal{O}$*) from* SAT *to the promise problem of, given a string* x*, outputting:*

- ***YES:*** *if* $\mathsf{K}^{t,\mathcal{O}}(x) \leq \theta(|x|)$
- ***NO:*** *if* $\mathsf{K}^{2^{|x|^{1-\epsilon}},\mathcal{O}}(x) \geq \theta(|x|) + \Omega(|x|)$

where t *is a polynomial in* x *and* θ *is some function of* $|x|$.

Although we presented the reduction earlier, we state it again so that we can make a remark. This reduction also works for MCSP with a different setting of the complexity threshold.

Non-Uniform Reduction from Gap τ-Frequency Set Cover to K^t and MCSP

Parameter: power of two $\lambda = \mathsf{poly}(n)$

Advice: hardcoded values $v_i \in \{0,1\}^\lambda$ and $c^\star_{i,j} \in \{0,1\}^{2\lambda}$ for all $i \in [n]$ and $j \in [m]$

Given: a τ-frequency set cover instance $S_1, \ldots, S_m \subseteq [n]$ where n is a power of two

1. Output the complexity threshold[a] $2\lambda n\tau + n\lambda/8 - 1$ and the $4\tau n\lambda$-bit truth table of the function $f : [n] \times [\tau] \times \{0,1\} \times [2\lambda] \to \{0,1\}$ given by

$$f(i,k,b,d) = \begin{cases} d\text{'th bit of } c^\star_{i,j}, & \text{if } b = 0, \text{ where } S_j \text{ is the } k\text{'th set containing } i \\ d\text{'th bit of } v_i, & \text{if } b = 1 \end{cases}.$$

[a] For MCSP we use a different threshold.

Observe that this reduction can be computed extremely easily: in non-uniform AC^0. Allender, Ilango, and Vafa [AIV21] prove that approximating $\mathsf{MCSP}^{\mathcal{O}}$ to a super constant factor is *not* NP-hard under non-uniform AC^0 reductions.[16] In contrast, we show the existence of oracles $\mathcal{O}$ such that NP reduces to large

[16] Actually, they show this for MCSP, but their result relativizes.

additive approximations of $\mathsf{MCSP}^{\mathcal{O}}$ under non-uniform AC^0 many-one reductions. The non-hardness result in [AIV21] perhaps suggests a barrier to achieving better, multiplicative, hardness of approximation using our techniques.

1.3 Our Approach (Conceptually): Pseudo Symmetry of Information

Here we give an overview of the main conceptual idea that led to our results. We do this with the aim of giving as much intuition as possible. We caution the reader that to actually execute the idea in this section, our proofs will diverge significantly from what is described and are more involved. Nevertheless, we feel that what we write below is the "root" of our results.

As mentioned earlier, prior work shows that the conditional string versions of MCSP and K^t are NP-hard [Ila20a; LP22; ACM+21; Hir22b; HIR23] (we will review what this means in a few sentences). Thus, one way to show NP-hardness is to give a reduction from these conditional string problems to the original (non-conditional) problems.

For concreteness, let us consider time-bounded Kolmogorov complexity, which turns out to be the simpler case for us. The conditional time-bounded Kolmogorov complexity of a string x given y, denoted $\mathsf{K}^t(x|y)$, is the length of the shortest program for computing x given access to y in time at most t (see Section 2 for a formal definition). Suppose one wants to estimate the quantity $\mathsf{K}^t(x|y)$ using an oracle that computes $\mathsf{K}^t(\cdot)$. A natural idea is to hope for *symmetry of information* to hold. That is to hope that

$$\mathsf{K}^t(xy) \approx \mathsf{K}^t(x|y) + \mathsf{K}^t(y)$$

with perhaps some loss in the time bounds. Intuitively, symmetry of information says that the "best" way of computing the concatenation xy is to first compute y and then compute x using information about y.

If symmetry of information were true, then we could estimate $\mathsf{K}^t(x|y)$ by $\mathsf{K}^t(xy) - \mathsf{K}^t(y)$, and hope to use this to prove NP-hardness of computing K^t using the existing NP-hardness for computing conditional K^t. Unfortunately, while symmetry of information holds for *time-unbounded* Kolmogorov complexity [ZL70], it is known to fail for *time-bounded* Kolmogorov complexity if one-way functions exist [LM93; LW95]. To get some intuition for this, consider the case where x is chosen uniformly at random from $\{0,1\}^n$ and $y = f(x)$ where $f : \{0,1\}^n \to \{0,1\}^n$ is a cryptographic one-way function. Then it is plausible that $\mathsf{K}^t(x|y) \approx n$ (because f is one-way) but also that

$$\mathsf{K}^t(xy) - \mathsf{K}^t(y) \approx n - n \approx 0,$$

so in this case symmetry of information fails quite badly. Intuitively, this is because the "one-way-ness" of f makes it much cheaper to describe x first and then compute y

Our main idea is to try to replace y with some encoding $\tilde{y}$ of y such that one can easily recover y from $\tilde{y}$ but such that symmetry of information will now hold. In other words, our goal is to construct $\tilde{y}$ from y such that

$$\mathsf{K}^t(x\tilde{y}) \approx \mathsf{K}^t(x|y) + \mathsf{K}^t(\tilde{y}),$$

allowing for some slack in the time bounds. We refer to this idea as *Pseudo Symmetry of Information.*

Our original intuition for constructing $\tilde{y}$ was guided by the idea of encryption. Suppose we let $\tilde{y}$ be an encryption of y according to some ideal (secret key) encryption scheme where the secret key is chosen randomly but then publicly revealed. Then one could decrypt $\tilde{y}$ to obtain y using the publicly known secret key. On the other hand, one could hope that the encryption $\tilde{y}$ is so good that it *only* reveals the value y and essentially looks essentially random otherwise. Then one could hope that the optimal way to describe $x\tilde{y}$ is to first describe $\tilde{y}$, then decrypt using the publicly known secret key to obtain y, and then describe x knowing y. If this were the case, then we would have

$$\mathsf{K}^t(x\tilde{y}) \approx \mathsf{K}^t(x|y) + \mathsf{K}^t(\tilde{y})$$

and thus that

$$\mathsf{K}^t(x|y) \approx \mathsf{K}^t(x\tilde{y}) - \mathsf{K}^t(\tilde{y}).$$

A first attempt is to see if the above approach works using Shannon's one-time-pad encryption scheme. Let $\mathcal{O} : \{0,1\}^\lambda \to \{0,1\}$ denote a uniformly random function (we will use $\mathcal{O}$ as our "one-time pad"). For each $i \in [n]$, we encrypt y_i (the ith bit of y) by picking some uniformly random $r_i \in \{0,1\}^\lambda$ and outputting $(r_i, \mathcal{O}(r_i) \oplus y_i)$. We then let $\tilde{y}$ be the concatenation of $(r_i, \mathcal{O}(r_i) \oplus y_i)$ for all $i \in [n]$. Note that $\tilde{y}$ is a string of length $n(\lambda + 1) = n\lambda + n$.

Now, we can ask: is it true that $\mathsf{K}^t(x\tilde{y}) \approx \mathsf{K}^t(x|y) + \mathsf{K}^t(\tilde{y})$ when we equip all Turing machines with oracle access to $\mathcal{O}$? Unfortunately, no. Consider again the case where $x \in \{0,1\}^n$ is uniformly random and $y = f(x)$ for a one-way function f. Then it is plausible that $\mathsf{K}^t(x|\tilde{y}) \approx n$ and that $\mathsf{K}^t(\tilde{y}) \approx n\lambda + n$. On the other hand, one can describe $x\tilde{y}$ by first describing x and then describing each r_i for all $i \in [n]$. To see that this allows one to quickly recover $x\tilde{y}$, note that one can compute $y = f(x)$, and then compute $\mathcal{O}(r_i) \oplus y_i$. This plausibly gives

$$\mathsf{K}^t(x\tilde{y}) \approx n + n\lambda.$$

So

$$\mathsf{K}^t(x\tilde{y}) - \mathsf{K}^t(\tilde{y}) \approx n + n\lambda - (n\lambda + n) \approx 0,$$

so symmetry of information still fails!

Why does the above approach fail? The problem seems to be that in the encryption the bit $\mathcal{O}(r_i) \oplus y_i$ is completely revealed once y and r_i are known. This gives a non-trivial way of computing $x\tilde{y}$ by describing x first.

This led us to a second attempt using a different encryption construction that removes the $\mathcal{O}(r_i) \oplus y_i$ bit. Again let $\mathcal{O} : \{0,1\}^\lambda \to \{0,1\}$ denote a uniformly random function. The new encryption scheme is as follows: for each $i \in [n]$, we encrypt y_i (the ith bit of y) by picking some uniformly random $r_i \in \{0,1\}^\lambda$ satisfying $\mathcal{O}(r_i) = y_i$. We then let $\tilde{y}$ be the $n\lambda$ length string that is the concatenation of r_i, for all $i \in [n]$.

Unfortunately, symmetry of information still might not hold in the one-way function example. Again when $y = f(x)$, it is plausible that $\mathsf{K}^t(x|\tilde{y}) \approx n$ and that $\mathsf{K}^t(\tilde{y}) \approx n\lambda$. But one can show that

$$\mathsf{K}^t(x\tilde{y}) \approx n + n(\lambda - 1) = n\lambda.$$

The reason for this bound is that you can hardcode in x, then compute y, and then by using query access to the oracle $\mathcal{O}$ one can on average compress each r_i by one bit by using the fact that $\mathcal{O}(r_i) = y_i$. This means that

$$\mathsf{K}^t(x\tilde{y}) - \mathsf{K}^t(\tilde{y}) \approx n\lambda - n\lambda \approx 0,$$

so it appears we have made no progress.

However, note there is one thing nice about this attempt. In order to compress each r_i by one bit, one needs to make (on average two) queries to the oracle. This gives us a computational lever to exploit. If instead of encrypting a single bit of y at a time, we encrypt ℓ bits of y at a time, one might expect that in order to compress the corresponding cipher text by ℓ bits, one might need to make on the order of 2^ℓ queries to the oracle. If ℓ is large, this could take far too much time![17]

This leads us to our third (and final) attempt. The main difference is to use the above scheme but instead of encrypting a single bit of y at a time, encrypt λ bits of y at a time. In more detail, let $\mathcal{O} : \{0,1\}^{2\lambda} \to \{0,1\}^\lambda$ denote a uniformly random function and assume λ evenly divides n. We divide y into n/λ blocks of λ bits. For each $i \in [n/\lambda]$, let y_i denote the i'th block, and we encrypt y_i by picking a uniformly random $r_i \in \{0,1\}^{2\lambda}$ such that $\mathcal{O}(r_i) = y_i$ (one can do this efficiently via rejection sampling if $\lambda = O(\log n)$, but let us ignore time complexity considerations for now). Then let $\tilde{y}$ be the length $2n$ string given by concatenating all these encryptions.

Now let us look at the one-way function case where $y = f(x)$. Then it is plausible that $\mathsf{K}^t(x|\tilde{y}) \approx n$ and that $\mathsf{K}^t(\tilde{y}) \approx 2n$. On the other hand, it appears that the best bound we can give on $\mathsf{K}^t(x\tilde{y})$ is to hardcode in x, then compute y given x, and then, because we are limited to at most t oracle queries by our running time bound, to compress about $\log t$ bits of each r_i. This gives

$$\mathsf{K}^t(x\tilde{y}) \leq n + 2n/\lambda(\lambda - O(\log t)) \leq n + 2n - O(2n/\lambda \log t).$$

[17] One way to view this is that, while in the information-theoretic regime pseudorandomness and incompressibility are equivalent, in the computational regime they are (likely) not!

Now let $\epsilon > 0$ be arbitrary. Then setting $\lambda = \gamma \log t$ for a sufficiently large constant γ that depends on ϵ, we get

$$\mathsf{K}^t(x\tilde{y}) \leq (3-\epsilon)n.$$

If this bound is tight, then we get that

$$\mathsf{K}^t(x\tilde{y}) - \mathsf{K}^t(\tilde{y}) \approx (3-\epsilon)n - 2n \approx (1-\epsilon)n \approx (1-\epsilon)\mathsf{K}^t(x|y)$$

and thus that symmetry of information holds on this example (approximately)! Of course, this is just a test case and not at all a formal proof.

With this in mind, we return to our original goal of showing that computing K^t is NP-hard. At this point there are two main problems:

1. the known NP-hardness of conditional time-bounded Kolmogorov complexity is not sufficient for our purposes, and
2. one needs to actually prove that $\mathsf{K}^t(x\tilde{y}) - \mathsf{K}^t(\tilde{y}) \approx \mathsf{K}^t(x|y)$.

We now discuss the first problem. While we stated earlier that the conditional versions of circuit complexity and time-bounded Kolmogorov complexity are NP-hard to compute, all known (unconditional) reductions only hold in the *sublinear* time-bound regime. For example, we only know how to prove NP-hardness of $\mathsf{K}^t(x|y)$ when the time-bound t is much shorter than the string y (i.e. one does not even have the opportunity to even look at all the bits in y). This is a significant obstacle for the "pseudo symmetry of information" approach we describe above. This is because it only makes sense to consider $\mathsf{K}^t(xy)$ when $t \geq |x| + |y|$, as one needs at least this much time to print all of x and y. Thus, we seem to require the NP-hardness of conditional time-bounded Kolmogorov complexity in the *superlinear* regime (i.e., where $t \geq |y|$). Such NP-hardness is known if one assumes the existence of cryptographic witness encryption [HIR23], but we would like to avoid making another assumption.

It turns out one can use the random oracle to get around this. At a high level, this is done by replacing information theoretic encryption schemes in prior reductions [LP22; Hir22a] with an encryption scheme using the random oracle. Because of the strong security of the random oracle encryption scheme, one can prove that computing conditional K^t complexity is NP-hard in the superlinear regime.

Now, the second problem is actually proving pseudo symmetry of information

$$\mathsf{K}^t(x\tilde{y}) - \mathsf{K}^t(\tilde{y}) \approx \mathsf{K}^t(x|y).$$

We show a version of this is true for the version of time-bounded Kolmogorov complexity equipped with "public randomness" (pK) introduced by Goldberg, Kabanets, Lu, and Oliveira [GKLO22]. In one (very high-level) sentence, the idea behind the proof is to use a hybrid argument to replace the oracle $\mathcal{O}$ with a highly biased oracle $\mathcal{O}'$ and then use $\mathcal{O}'$ to construct small descriptions of x given y.

Theorem 9 (Informal version of Corollary 19)**.** *Let $\mathcal{O}$ be a uniformly random oracle. Let $x, y \in \{0,1\}^n$. Then with probability $1 - 2^{-\Omega(n)}$ we have that*

$$\mathsf{pK}^{t''}(x|y) - n/100 \leq \mathsf{pK}^{t',\mathcal{O}}(x\tilde{y}) - |\tilde{y}| \leq \mathsf{pK}^{t}(x|y) + n/100$$

where $t' = \mathsf{poly}(t)$ and $t'' = \mathsf{poly}(t')$.

However, proving a similar result for other complexity measures like circuit complexity seems quite difficult (and potentially even false). As a result, our actual approach diverges significantly. We carefully combine the ideas described in this subsection with Hirahara's reduction [Hir22a] for the partial function version of MCSP. We then prove this reduction is correct "by hand," using a somewhat technical random oracle argument.

1.4 Related Work

Our results build on techniques and ideas developed in a long sequence of work showing conditional and unconditional NP-hardness of problems related to MCSP [Mas79; Czo99; UVS06; ABK+06; AHM+08; Fel09; HOS18; Ila20a; ILO20; Ila20b; Ila21; ACM+21; LP22; Hir22b; Hir22a; HIR23].

In particular, as stated earlier, we arrived at our reductions by carefully applying the idea of pseudo symmetry of information to the reduction used in Hirahara's recent result [Hir22a], which itself builds on prior reductions. The problem we reduce from, τ-frequency Set Cover, is a special case of Collective Minimum Monotone Satisfying Assignment (the problem reduced from in [Hir22a]). This special case is especially nice as it does not require us to use any secret sharing scheme.

It is also worth noting that τ-frequency Set Cover seems closely related to the r-bounded Set Cover problem which was first used to prove hardness in metacomplexity by Hirahara, Oliveira, and Santhanam [HOS18] and then in several other results [Ila20a; ILO20; ACM+21; LP22].

As mentioned previously, the recent paper of Huang, Ilango, and Ren [HIR23] first introduced the idea of using a random oracle to prove hardness of metacomplexity problems, and they conjectured that one can show that with probability one $\mathsf{3\text{-}SAT}$ reduces to $\mathsf{MCSP}^{\mathcal{O}}$ under $\mathcal{O}$-oracle quasipolynomial-time reductions.

One can view our results as showing that the conditional string versions of MCSP (i.e., MOCSP [Ila20a]) and K^t are NP-hard even when the conditional string is chosen uniformly at random. In contrast, previous NP-hardness results for these problems [Ila20a; ACM+21; LP22; Hir22b; HIR23] required the conditional string to depend on the instance fed to the reduction.

Impagliazzo, Kabanets, and Volkovich [IKV18] show that if cryptographic indistinguishability obfuscators exist, then MCSP is in ZPP if and only if NP is in ZPP. One can view this as a non-black-box reduction from SAT to MCSP under the assumption that indistinguishability obfuscators exist.

1.5 Open Questions

Our main open question is to give a plausible assumption under which our reduction works *without* assuming a random oracle. It would also be useful to better understand what properties of a random oracle are truly necessary to carry out our proof. We remark that the notions of keyless hash functions [BKP18] and extremely lossy functions [Zha19] seem potentially relevant.

It is worth noting that for the purpose of, say, eliminating Heuristica it suffices to instantiate the oracle under the assumption that approximating MCSP and K^t are in P. Could such an assumption be helpful?

Another interesting direction is to explore further extensions. Is it possible, for example, to show *multiplicative* hardness of approximation in the random oracle model? What can be shown in other generic models, where one assumes access to some idealized function?

While our techniques extend to many metacomplexity problems, one exception is formula complexity. At a high level, the difficulty with formula complexity is that it does not interact nicely with pseudo symmetry of information. In particular, one would like to say that if one has computed $\tilde{y}$, then one can recover y by "decrypting," but this could cause a large blowup in the case of formulas. Can one extend our results to capture formula complexity as well?

2 Preliminaries

We assume basic familiarity with computational complexity theory and Kolmogorov complexity, as can be found in the textbooks [AB09] and [LV19] respectively.

For a positive integer n, let $[n]$ denote the set $\{1, \dots, n\}$. For a finite set S, we let $s \leftarrow S$ denote sampling a uniformly random element from S. For two binary strings x and y, we let xy denote their concatenation. $\{0,1\}^\star$ denotes the set of all finite binary strings. The abbreviation i.i.d. stands for independently identically distributed. For an event E, we let $\mathbb{1}[E]$ denote the indicator event on E.

For parameters $p_1, \dots, p_k$, we write $O_{p_1,\dots,p_k}$ and $\Omega_{p_1,\dots,p_k}$ to denote that the hidden constant can depend on $p_1, \dots, p_k$. If no parameters are specified in the subscript, then the hidden constant is independent of all other parameters. We write $\mathsf{poly}(p_1, \dots, p_k)$ to denote some polynomial in the parameters $p_1, \dots, p_k$.

2.1 Random Oracles

A random oracle $\mathcal{O} : \{0,1\}^\star \to \{0,1\}$ is a random variable where for all x the value of $\mathcal{O}(x)$ is set independently to a uniformly random element of $\{0,1\}$.

While, strictly speaking, a random oracle only outputs one bit, one can easily extend the number of outputs of $\mathcal{O}$ through concatenation. For example, for any polynomial p, one can construct from $\mathcal{O}$ a new function $\mathcal{O}' : \{0,1\}^\star \to \{0,1\}^\star$ that maps n bit inputs to $p(n)$ bit outputs by setting

$$\mathcal{O}'(x) = \mathcal{O}(1^{|x|}, 0, x, 0) \dots \mathcal{O}(1^{|x|}, 0, x, p(n) - 1)$$

where $p(n)-1$ is the binary representation of the corresponding integer. Observe that $\mathcal{O}'$ is a random variable where for all x the value of $\mathcal{O}'(x)$ drawn i.i.d. from the uniform distribution on $\{0,1\}^{p(|x|)}$. Moreover, $\mathcal{O}'$ can be computed in polynomial time given oracle access to $\mathcal{O}$.

Another similar idea is that one can construct *multiple independent* random oracles given access to a single random oracle $\mathcal{O} : \{0,1\}^\star \to \{0,1\}$. For example, to construct two independent oracles $\mathcal{O}_0$ and $\mathcal{O}_1$ from $\mathcal{O}$, let $\mathcal{O}_0$ be given by $\mathcal{O}_0(x) = \mathcal{O}(0,x)$, and let $\mathcal{O}_1$ be by $\mathcal{O}_1(x) = \mathcal{O}(1,x)$. We will use both of these ideas in our reduction.

A basic tool for analyzing events in the presence of a random oracle is the Borel-Cantelli Lemma. In particular, one can use the Borel-Cantelli Lemma to show an event occurs with probability one (or zero) when an oracle is chosen uniformly at random.

Lemma 10 (Borel-Cantelli Lemma)**.** *Let E_n be a sequence of events. If $\sum_{n=1}^{\infty} \Pr[E_n] < \infty$, then the probability that infinitely many of the events occur is zero.*

2.2 Circuits and The Minimum Circuit Size Problem

We will consider circuits that use NOT gates and fan-in two AND and OR gates. The size of a circuit is the number of AND and OR gates in the circuit. We note that, for our results, any reasonable variation on these definitions is okay (e.g., gates over the binary basis or size measured by the number of wires).

For a function $\mathcal{O} : \{0,1\}^\star \to \{0,1\}$, an $\mathcal{O}$-oracle circuit is a circuit using NOT gates, fan-in two AND and OR gates, and gates G_n for all $n \in \mathbb{N}$ that take as input $x \in \{0,1\}^n$ about output $\mathcal{O}(x)$. The size of an oracle circuit is the total number of AND and OR gates plus the total number of wires that feed into any G_n gate. Again, our results hold for most reasonable definitions of $\mathcal{O}$-oracle circuits and their size.

Definition 11 ($\mathsf{MCSP}^{\mathcal{O}}$ [ABK+06])**.** The $\mathcal{O}$-oracle Minimum Circuit Size Problem ($\mathsf{MCSP}^{\mathcal{O}}$) is the following computational task:

- **Given:** a Boolean function $f : \{0,1\}^n \to \{0,1\}$ (represented by its truth table of length 2^n) and a size parameter s,
- **Decide:** if there is an $\mathcal{O}$-oracle Boolean circuit that computes f of size at most s.

When the oracle is empty, this problem is the usual MCSP.

2.3 Kolmogorov Complexity

Fix a time-optimal universal Turing machine U with random access to its input. (We refer the reader to [LV19] for a formal definition. Essentially, the machine U takes two inputs (M, x) interprets M as the description of a Turing machine and simulates running M on x. The machine being time-optimal essentially means the simulation blow-up is not too large.)

Definition 12 (Time-Bounded Kolmogorov Complexity [Sip83; Ko86; Har83; Ko91])**.** For an oracle $\mathcal{O}$, a binary string $x \in \{0,1\}^n$ and a (possibly empty) binary string y, the *$\mathcal{O}$-oracle t-time bounded Kolmogorov complexity of x given y*, denoted $\mathsf{K}^{t,\mathcal{O}}(x|y)$, is the minimum length of any d such that $U^{\mathcal{O}}(d, y)$ outputs x in time at most t.

In the case where $\mathcal{O}$ is empty, this is the usual definition of conditional time-bounded Kolmogorov complexity, and we omit the $\mathcal{O}$ superscript. In the case when y is empty, we will also omit y. We refer to d as a *t-time description*.

We will also use a generalization of time-bounded Kolmogorov complexity introduced by Goldberg, Kabanets, Lu, and Oliveira [GKLO22] that allows for U to take as input "public coin" randomness r on a separate tape (to visually separate r from the other inputs we use a colon ";"). Goldberg, Kabanets, Lu, and Oliveira [GKLO22] show that pK^t coincides with K^t (up to an additive logarithmic term) under a derandomization assumption. An excellent exposition on probabilistic time-bounded Kolmogorov complexity can be found in a recent survey by Lu and Oliveira [LO22].

Definition 13 (Probabilistic Time-Bounded Kolmogorov Complexity [GKLO22])**.** For an oracle $\mathcal{O}$, a binary string $x \in \{0,1\}^n$ and a (possibly empty) binary string y, the *$\mathcal{O}$-oracle probabilistic t-time bounded Kolmogorov complexity of x given y* denoted $\mathsf{pK}^{t,\mathcal{O}}(x|y)$, is the minimum length ℓ such that for all least two-thirds of all "public randomness" $r \in \{0,1\}^t$ there exists a $d_r \in \{0,1\}^\ell$ such that $U^{\mathcal{O}}(d_r, y; r)$ outputs x in time at most t.

If $\mathcal{O}$ or y is empty, we omit writing them. It will sometimes be useful to fix a specific choice of public randomness $r \in \{0,1\}^t$. To do this, we let $\mathsf{pK}^{t,\mathcal{O}}(x|y\,;r)$ denote the length of the smallest d such that $U^{\mathcal{O}}(d, y; r)$ outputs x in time at most t.

The following proposition follows from the definition of pK.

Proposition 14. *If $\mathsf{pK}^{t,\mathcal{O}}(x|y) \le \alpha$, then*

$$\Pr_{\substack{r \leftarrow \{0,1\}^t \\ \ell \leftarrow [\alpha] \\ d \leftarrow \{0,1\}^\ell}}[U^{\mathcal{O}}(d, y; r) \textit{outputs } x \textit{ in time } t] \ge \frac{2}{3}\frac{1}{\alpha 2^\alpha}$$

An important property about pK is that it has a *coding theorem.* Informally, this says that if one can sample a string x with probability p in time t, then one can upper bound its pK complexity by roughly $1/p$.

Theorem 15 (Coding Theorem for pK [LOZ22])**.** *Let y be a (possibly empty) binary string. Let A be a randomized Turing machine that runs in time $T \ge n$ on inputs of length n. Let $x \in \{0,1\}^n$ and let p be the probability that $A(1^n 0 y)$ outputs x. Then*

$$\mathsf{pK}^t(x \mid y) \le \log(1/p) + O(|A| + \log(nT))$$

where $t = \mathsf{poly}(T)$ and $|A|$ denotes the length of the code to describe A. We stress that the hidden constants in this statement are universal and do not depend on any parameters.

Proof (Sketch). We give a brief sketch. See [LOZ22] for the full details.

Since A runs in time T, A uses at most T bits of randomness. Let $r \in \{0,1\}^{\mathsf{poly}(T)}$ denote our public randomness. Let α be a parameter we set later.

Let $h_r : \{0,1\}^\alpha \to \{0,1\}^T$ be some seeded pseudorandom generator construction (with some "nice" properties that we will not specify in this sketch) against AC^0 with its seed set to r. Then for if $\alpha \ge \log(1/p) + \Omega(1)$, one can prove that with probability at least $2/3$ (over r) there exists a string $d_r \in \{0,1\}^\alpha$ such that A run on input $(1^n, y)$ with randomness $h_r(d_r)$ outputs x. Thus one can describe x conditioned on y by describing d, the code for h (whose length is bounded by some universal constant), the code for A, and the values n and T. Moreover, one can show this description runs in time polynomial in T. □

3 Pseudo Symmetry of Information

In this section, we show that pseudo symmetry of information holds for pK with a random oracle. While, ultimately, we do not rely on this result in our final reductions, we feel it is a conceptually important building block and is interesting in its own right.

3.1 Weak Symmetry of Information

To begin, we first show that symmetry of information holds unconditionally for pK when one of the strings is chosen uniformly at random. This special case, first studied by Hirahara [Hir21] for (non-probabilistic) time-bounded Kolmogorov complexity, is referred to as *weak symmetry of information*. In more detail, it is easy to see that for any strings $x \in \{0,1\}^n$ and $z \in \{0,1\}^q$ that

$$\mathsf{pK}^{\mathsf{poly}(t,n,q)}(xz) \leq \mathsf{pK}^t(x) + q + O(\log nqt)$$

by giving a t-time description for x and then hardcoding in z. Weak symmetry of information essentially says that for any fixed x this upper bound is approximately tight (with some loss in the time bound) when z is chosen at random.

Weak symmetry of information is an important building block in recent results in average-case complexity [Hir21; HN21; GK22; Hir22b; CHV22], where it is shown that weak symmetry of information holds if NP is easy on average. To our knowledge, it was not known previously that weak symmetry of information can be proved *unconditionally* for pK.

Theorem 16 (Weak symmetry of information for pK)**.** *Let $x \in \{0,1\}^n$, $\Delta \geq 0$, $q \in \mathbb{N}$, and $t \geq 1$. Let $z \leftarrow \{0,1\}^q$. With probability at least $1 - 2^{-\Delta}$ the following holds*

$$\mathsf{pK}^t(xz) \geq \mathsf{pK}^{t'}(x) + q - \Delta - O(\log nqt)$$

where $t' = \mathsf{poly}(n,q,t)$.

Proof. Let $\alpha \in \mathbb{N}$ be a parameter we set later. Let

$$p = \Pr_{z \leftarrow \{0,1\}^q}[\mathsf{pK}^t(xz) \leq \alpha].$$

Our goal is to show that p is small.

By Proposition 14, we know that if $\mathsf{pK}^t(xz) \leq \alpha$, then

$$\Pr_{\substack{\ell \leftarrow [\alpha] \\ d \leftarrow \{0,1\}^\ell \\ r \leftarrow \{0,1\}^t}}[U(d;r) = xz \text{ in time } t] \geq \frac{2}{3}\,\frac{1}{\alpha 2^\alpha}.$$

By the definition of p, we then get that

$$\sum_{z \in \{0,1\}^q} \Pr_{\substack{\ell \leftarrow [\alpha] \\ d \leftarrow \{0,1\}^\ell \\ r \leftarrow \{0,1\}^t}}[U(d;r) = xz \text{ in time } t] \geq 2^q\, p\, \frac{2}{3}\,\frac{1}{\alpha 2^\alpha}.$$

By the disjointness of the events that $U(d;r) = xz$ for distinct z, we get

$$\Pr_{\substack{\ell \leftarrow [\alpha] \\ d \leftarrow \{0,1\}^\ell \\ r \leftarrow \{0,1\}^t}}[U(d;r) = xz \text{ in time } t \text{ for some } z \in \{0,1\}^q] \geq 2^q\, p\, \frac{2}{3}\,\frac{1}{\alpha 2^\alpha}$$

which implies that

$$\Pr_{\substack{\ell \leftarrow [\alpha] \\ d \leftarrow \{0,1\}^\ell \\ r \leftarrow \{0,1\}^t}}[U(d;r) \text{ outputs a string in time } t \text{ whose first } n \text{ bits are } x\ ] \geq 2^q\, p\, \frac{2}{3}\,\frac{1}{\alpha 2^\alpha}. \tag{1}$$

Observe that Equation (1) implies there is a distribution (sample ℓ, d, and r as described and output the first n bits of $U(d;r)$) that outputs x with probability at least $2^q\, p\, \frac{2}{3}\, \frac{1}{\alpha 2^\alpha}$. Moreover this distribution can be sampled in time $\mathsf{poly}(t, \alpha)$ and the code of the sampler has length at most $O(\log nq\alpha t)$. Thus, by the coding theorem (Theorem 15), we have

$$\mathsf{pK}^{t'}(x) \leq -q + \alpha + \log(1/p) + O(\log(\alpha nqt))$$

for some $t' = \mathsf{poly}(t, n, q, \alpha)$. Rearranging, we get

$$p \leq 2^{-q+\alpha-\mathsf{pK}^{t'}(x)+O(\log \alpha nqt)}.$$

The theorem then follows by setting $\alpha = q + \mathsf{pK}^{t'}(x) - \Delta - O(\log nqt)$. Note that with this setting of α, we have that $\alpha \leq q + n + O(1)$, so $t' = \mathsf{poly}(t, n, q)$ and $O(\log \alpha nqt) = O(\log nqt)$. □

3.2 Pseudo Symmetry of Information

We will now build on the ideas used to prove weak symmetry of information to prove pseudo symmetry of information. We begin by introducing some notation we will use throughout this subsection. Let $x \in \{0,1\}^n$. Let $y \in \{0,1\}^m$. Let $\lambda \in \mathbb{N}$ be some parameter. While our theorems hold in a more general setting, to reduce the number of parameters and simplify notation we will concentrate on the case when $n = \Theta(m)$ and assume that λ divides m. For all $i \in [m/\lambda]$, let $y_i \in \{0,1\}^\lambda$ be the string such that y is the concatenation of $y_1, \ldots, y_{m/\lambda}$. To simplify our presentation, we will also only consider oracles $\mathcal{O} : \{0,1\}^{2\lambda} \to \{0,1\}^\lambda$ with a fixed input and output length.

We define a distribution $\mathsf{Encode}[y, \mathcal{O}]$ on $2m$-bit strings as follows:

Sampling procedure for $\mathsf{Encode}[y, \mathcal{O}]$

1. For each $i \in [m/\lambda]$, let $\tilde{y}_i \in \{0,1\}^{2\lambda}$ be a uniformly random element of the set $\{r : \mathcal{O}(r) = y_i\}$. (If this set is ever empty, then "fail" and output 0^{2m}.)
2. Let $\tilde{y} \in \{0,1\}^{2\lambda}$ be the concatenation of all the $\tilde{y}_i$.
3. Output $\tilde{y}$.

In this section, we will not worry about time complexity (since our final reductions are proved using different methods), but we do note that $\mathsf{Encode}[y, \mathcal{O}]$ is samplable, with high probability over a uniformly random $\mathcal{O}$, in time $\mathsf{poly}(m, 2^\lambda)$ using rejection sampling.

Our goal is to show that pseudo symmetry of information holds when one samples $\tilde{y}$ from $\mathsf{Encode}[y, \mathcal{O}]$. In particular, ideally we would like that when $\mathcal{O} : \{0,1\}^{2\lambda} \to \{0,1\}^\lambda$ is chosen uniformly at random and $\tilde{y} \leftarrow \mathsf{Encode}[y, \mathcal{O}]$ that the following (informal statement) holds with high probability:

$$\mathsf{pK}^{t,\mathcal{O}}(x\tilde{y}) \approx \mathsf{pK}^t(x|y) + |\tilde{y}| \approx \mathsf{pK}^t(x|y) + 2m$$

where we allow for a polynomial gap in the time bound between the upper and lower bounds.

On the one hand, it is easy to see that the upper bound holds.

Proposition 17. *Let $\mathcal{O} : \{0,1\}^{2\lambda} \to \{0,1\}^\lambda$ be a uniformly random function and $\tilde{y} \leftarrow \mathsf{Encode}[y, \mathcal{O}]$. Then with probability $1 - m\exp(-2^\lambda)$ we have that*

$$\mathsf{pK}^{\mathsf{poly}(t),\mathcal{O}}(x\tilde{y}) \leq \mathsf{pK}^t(x|y) + 2m + O(\log t\lambda nm).$$

Proof (Sketch). To describe $x\tilde{y}$, we do the following. First describe $\tilde{y}$ (which costs roughly $|\tilde{y}|$ bits). Then, assuming that the sampling algorithm for $\mathsf{Encode}[y, \mathcal{O}]$ did not fail when sampling $\tilde{y}$, we can use the oracle $\mathcal{O}$ to obtain y (using that $y_i = \mathcal{O}(\tilde{y}_i)$). Then we can compute x given y (using a description witnessing $\mathsf{pK}^t(x|y)$). This yields the desired upper bound (if sampling did not fail). The probability (over $\mathcal{O}$) that the sampling from $\mathsf{Encode}[y, \mathcal{O}]$ fails is at most

$$m(1 - 2^{-\lambda})^{2^{2\lambda}} \leq m\exp(-2^\lambda)$$

□

We will show a corresponding lower bound. In the proof we will need to introduce several distributions from which to sample oracles $\mathcal{O}$ (we will use a hybrid argument over these distributions). Let $\mathsf{Oracle.Uniform}$ denote the uniform distribution on oracles $\mathcal{O} : \{0,1\}^{2\lambda} \to \{0,1\}^{\lambda}$.

Theorem 18. *Let $x \in \{0,1\}^n$, $y \in \{0,1\}^m$, and $\lambda \geq \Omega(\log m)$. Assume λ divides m. Then for all $\Delta \leq 2m$*

$$\Pr_{\substack{\mathcal{O}\leftarrow \mathsf{Oracle.Uniform}\\ \tilde{y}\leftarrow \mathsf{Encode}[y,\mathcal{O}]}}[\mathsf{pK}^{t',\mathcal{O}}(x\tilde{y}) \geq \mathsf{pK}^{t}(x|y) + 2m - \Delta] \geq 1 - 2^{-\Delta + O(\frac{m}{\lambda}\log(tmn))}$$

where $t' = \mathsf{poly}(n, m, t)$.

Before we prove Theorem 18, we show its power with the following immediate corollary by setting $m = n$ and $\lambda \geq \Omega(\log n)$.

Corollary 19 (Pseudo Symmetry of Information for pK)**.** *Let n be a power of two. Let $x \in \{0,1\}^n$, $y \in \{0,1\}^n$ and $t = \mathsf{poly}(n)$. Let $\lambda \geq \Omega(\log n)$ be a divisor of n. Let $\Delta = \frac{n}{100}$. Then with probability at least $1 - O(2^{-\Delta/2})$ over $\mathcal{O} \leftarrow \mathsf{Oracle.Uniform}$ and $\tilde{y} \leftarrow \mathsf{Encode}[y, \mathcal{O}]$ we have that*

$$\mathsf{pK}^{t''}(x|y) - \Delta \leq \mathsf{pK}^{t',\mathcal{O}}(x\tilde{y}) - 2n \leq \mathsf{pK}^{t}(x|y) + \Delta$$

where $t' = \mathsf{poly}(t)$ and $t'' = \mathsf{poly}(t')$.

We now prove Theorem 18.

Proof of Theorem 18. Let α be a parameter we set later. Let

$$p = \Pr_{\substack{\mathcal{O}\leftarrow \mathsf{Oracle.Uniform}\\ \tilde{y}\leftarrow \mathsf{Encode}[y,\mathcal{O}]}}[\mathsf{pK}^{t,\mathcal{O}}(x\tilde{y}) \leq \alpha].$$

Our goal is to show that p is small.

We will argue essentially using a hybrid argument where we change the distributions on $\mathcal{O}$ and $\tilde{y}$ twice without changing the probability that $\mathsf{pK}^{t,\mathcal{O}}(x\tilde{y}) \leq \alpha$ by much (multiplicatively). Then we will apply the coding theorem (in a similar manner as in the proof of weak symmetry of information) in order to upper bound p.

Our first change will be to "swap" the order of sampling. Instead of first sampling $\mathcal{O}$ and then sampling $\tilde{y}$ from a distribution that depends on $\mathcal{O}$, we will change to sampling $\tilde{y}$ first (uniformly at random from $\{0,1\}^{2m}$) and then sampling the oracle $\mathcal{O}$ from a distribution that depends on $\tilde{y}$. In particular, for any $\tilde{y} \in \{0,1\}^{2m}$ we define the distribution $\mathsf{Oracle.Swapped}[y, \tilde{y}]$ on oracles as follows:

Sampling Procedure for $\mathsf{Oracle.Swapped}[y, \tilde{y}]$

1. Pick a uniformly random function $\mathcal{O} : \{0,1\}^{2\lambda} \to \{0,1\}^{\lambda}$ satisfying that for all $i \in [m/\lambda]$ that $\mathcal{O}(\tilde{y}_i) = y_{i^\star}$ where $i^\star$ is[a] the least integer such that $\tilde{y}_i = \tilde{y}_{i^\star}$.
2. Output $\mathcal{O}$.

[a] The choice of $i^\star$ is so we smoothly handle the case when there is an i and j such that $\tilde{y}_i = \tilde{y}_j$ but $y_i \neq y_j$.

Using a Chernoff bound, one can show that swapping the order does not affect the probability that $\mathsf{pK}^{t,\mathcal{O}}(x\tilde{y}) \leq \alpha$ by much multiplicatively. The proof is a straightforward calculation that we defer to the end of the subsection.

Proposition 20. *Assume $\lambda \geq \Omega(\log m)$. Then*

$$\Pr_{\substack{\tilde{y}\leftarrow\{0,1\}^{2m}\\ \mathcal{O}\leftarrow \mathsf{Oracle.Swapped}[y,\tilde{y}]}}[\mathsf{pK}^{t,\mathcal{O}}(x\tilde{y}) \leq \alpha] \geq p/4 - 2^{-2m}$$

Our next step is to "decouple" the dependence between sampling $\tilde{y}$ and $\mathcal{O}$. Currently, the distribution $\mathsf{Oracle.Swapped}[y, \tilde{y}]$ depends on $\tilde{y}$. As we will see later, it will be useful to remove this dependence. To do this while still maintaining the probability that $\mathsf{pK}^{t,\mathcal{O}}(x\tilde{y}) \leq \alpha$ multiplicatively, we will sample functions $\mathcal{O}$ that are "biased" towards outputting y. Formally, we define $\mathsf{Oracle.Biased}[y]$ to be the following distribution on oracle functions $\mathcal{O}$:

Sampling Procedure for $\mathsf{Oracle.Biased}[y]$

1. Construct $\mathcal{O} : \{0,1\}^{2\lambda} \to \{0,1\}^{\lambda}$ where for all $z \in \{0,1\}^{2\lambda}$ the value of $\mathcal{O}(z)$ is drawn i.i.d. from the following distribution:

$$\begin{cases} \text{a uniformly random element of } \{0,1\}^{\lambda}, & \text{with probability } 1 - 1/t^2 \\ y_i \text{ for a uniformly random } i \in [m/\lambda], & \text{with probability } 1/t^2. \end{cases}$$

2. Output $\mathcal{O}$.

We claim that changing to $\mathsf{Oracle.Biased}[y]$ does not change the probability that $\mathsf{pK}^{t,\mathcal{O}}(x\tilde{y}) \leq \alpha$ by much multiplicatively, as long as one also includes the choice of public randomness into the probability. Intuitively, this is because t-time adversaries cannot distinguish between the biased oracle and a uniformly random oracle since the biasing only occurs with probability $1/t^2$ and because with not too small probability $\mathcal{O}(\tilde{y}_i) = y_i$ occurs.

Claim 21.

$$\Pr_{\substack{\tilde{y} \leftarrow \{0,1\}^{2m} \\ \mathcal{O} \leftarrow \mathsf{Oracle.Biased}[y] \\ r \leftarrow \{0,1\}^t}} [\mathsf{pK}^{t,\mathcal{O}}(x\tilde{y}; r) \leq \alpha] \geq (\frac{\lambda}{t^2 m})^{m/\lambda} (p/4 - 2^{-2m}) \frac{2}{3}(1 - 1/t).$$

Proof. Consider the following procedure for sampling $\tilde{y}$, r, and $\mathcal{O}$.

1. For each $i \in [m/\lambda]$, let v_i be drawn i.i.d. from the following distribution (it is the same one we use to sample the outputs of $\mathcal{O}$ in $\mathsf{Oracle.Biased}$):

$$\begin{cases} \text{a uniformly random element of } \{0,1\}^{\lambda}, & \text{with probability } 1 - 1/t^2 \\ y_i \text{ for a uniformly random } i \in [m/\lambda], & \text{with probability } 1/t^2. \end{cases}$$

2. Sample $\tilde{y}$ uniformly at random from $\{0,1\}^{2\lambda}$.

3. Sample a uniformly random function $\mathcal{O}' : \{0,1\}^{2\lambda} \to \{0,1\}^{\lambda}$ satisfying $\mathcal{O}'(\tilde{y}_i) = v_{i^\star}$ where $i^\star$ is the least integer satisfying $\tilde{y}_i = \tilde{y}_{i^\star}$.

4. Pick $r \in \{0,1\}^t$ uniformly at random.

5. Construct $\mathcal{O}$ via the following process for each $z \in \{0,1\}^{2\lambda}$: If $z = \tilde{y}_i$ for some i, let $\mathcal{O}(z) = \mathcal{O}'(z)$. Otherwise, then let $\mathcal{O}(z)$ be drawn from the distribution

$$\begin{cases} \mathcal{O}'(z), & \text{with probability } 1 - 1/t^2 \\ y_i \text{ for a uniformly random } i \in [m/\lambda], & \text{with probability } 1/t^2. \end{cases}$$

6. Output $\tilde{y}, r$ and $\mathcal{O}$.

Observe that this procedure is equivalent to sampling $\tilde{y}$ and r uniformly at random and sampling $\mathcal{O}$ from $\mathsf{Oracle.Biased}$.

We will now analyze the sampling procedure in stages to prove the claim. When one runs the sampling procedure, we say step 1 in the sampling procedure is *good* if after step 1 finishes, the following event occurs: for all $i \in [m/\lambda]$, we have $v_i = y_i$. By construction, step 1 is good with probability at least $(\frac{\lambda}{t^2 m})^{m/\lambda}$.

Next, we say that step 3 in the sampling procedure is *good* if after step 3 is completed the following event occurs: $\mathsf{pK}^{t,\mathcal{O}'}(x\tilde{y}) \leq \alpha$. The key observation is that, conditioned step 1 being good, the resulting distribution on $\tilde{y}$ and $\mathcal{O}'$ after step 3 finishes is exactly the distribution obtained by sampling $\tilde{y}$ uniformly and sampling $\mathcal{O}'$ from $\mathsf{Oracle.Swapped}[y, \tilde{y}]$. As a result, using Proposition 20, we get that conditioning on step 1 being good, step 3 is good with probability at least $p/4 - 2^{-2m}$.

We say that step 4 in the sampling procedure is *good* if $\mathsf{pK}^{t,\mathcal{O}'}(x\tilde{y}; r) \leq \alpha$. Conditioning on step 3 being good, step 4 is good with probability at least $2/3$ by the definition of pK.

Finally, we say that step 5 is *good* if $\mathsf{pK}^{t,\mathcal{O}}(x\tilde{y}; r) \leq \alpha$. If step 4 is good, then let d be the description such that $U^{\mathcal{O}'}(d; r) = x\tilde{y}$ in time at most t. Note that when U is run on d, it makes at most t queries to $\mathcal{O}$' (since it runs for time at most t). By construction of $\mathcal{O}$, we get that for any fixed query q, the probability that $\mathcal{O}(q) \neq O'(q)$ is at most $1/t^2$. Thus a union bound implies that the probability that $U^{\mathcal{O}}(d; r) = x\tilde{y}$ is at least $1 - 1/t$.

Putting this all together, we get that the probability that $\mathsf{pK}^{t,\mathcal{O}}(x\tilde{y}; r) \leq \alpha$ is at least

$$(\frac{\lambda}{t^2 m})^{m/\lambda}(p/4 - 2^{-2m})\frac{2}{3}(1 - 1/t)$$

□

Now that we have Claim 21 our goal is to mimic the proof of weak symmetry of information and use Claim 21 to give a way of sampling x. A direct consequence of Claim 21 is that

$$\Pr_{\substack{\tilde{y} \leftarrow \{0,1\}^{2m} \\ \mathcal{O} \leftarrow \mathsf{Oracle.Biased}[y] \\ r \leftarrow \{0,1\}^t \\ \ell \leftarrow [\alpha] \\ d \leftarrow \{0,1\}^\ell}} [U^{\mathcal{O}}(d; r) = x\tilde{y} \text{ in time } t] \geq \frac{1}{\alpha 2^\alpha}(\frac{\lambda}{t^2 m})^{m/\lambda}(p/4 - 2^{-2m})\frac{2}{3}(1 - 1/t).$$

Thus, as in the proof of weak symmetry of information, we can sum over the disjoint events for each $\tilde{y}$ (here is where we need that $\mathsf{Oracle.Biased}$ is decoupled from $\tilde{y}$) to get that

$$\begin{aligned} & \Pr_{\substack{\mathcal{O} \leftarrow \mathsf{Oracle.Biased}[y] \\ r \leftarrow \{0,1\}^t \\ \ell \leftarrow [\alpha] \\ d \leftarrow \{0,1\}^\ell}} [U^{\mathcal{O}}(d; r) \text{ outputs a string whose first } n \text{ bits are } x \text{ in time } t] \qquad (2) \\ \geq \quad & 2^{2m}\frac{1}{\alpha 2^\alpha}(\frac{\lambda}{t^2 m})^{m/\lambda}(p/4 - 2^{-2m})\frac{2}{3}(1 - 1/t) \end{aligned}$$

We want to use the above probability bound to obtain a distribution that is efficiently samplable given access to y and that outputs x with somewhat high probability. At first glance, this may seem problematic because one needs to sample $\mathcal{O} \leftarrow \mathsf{Oracle.Biased}[y]$ which could take a long time. Luckily, this is not a problem because $\mathsf{Oracle.Biased}[y]$ is an extremely simple distribution: every bit in the truth table output by $\mathsf{Oracle.Biased}[y]$ is an i.i.d. sample from a distribution that is known (assuming one knows y). As a result, instead of sampling $\mathcal{O}$ all at once, we can sample the responses to individual queries to $\mathcal{O}$ "on the fly." In particular, consider the distribution sampled as follows:

1. Pick $r \leftarrow \{0,1\}^t$ and $\ell \leftarrow [\alpha]$ and $d \leftarrow \{0,1\}^\ell$

2. Simulate running $U^{\mathcal{O}}(d; r)$ for at most t steps with an oracle $\mathcal{O}$ that constructed "on the fly" to match the distribution as if $\mathcal{O}$ is sampled from $\mathcal{D}_{biased}[y, \lambda, k]$. In particular, whenever U queries $\mathcal{O}$ on a point

q, do the following. If $\mathcal{O}$ was previously queried on q, respond to the query with whatever the previous response was. If $\mathcal{O}$ has never been queried on q previously, then respond with

$$\begin{cases} \text{a uniformly random element of } \{0,1\}^\lambda, & \text{with probability } 1 - 1/t^2 \\ y_i \text{ for a uniformly random } i \in [m/\lambda], & \text{with probability } 1/t^2. \end{cases}$$

3. Output the first n bits that $U^{\mathcal{O}}(d;r)$ outputs

Observe that the output of this distribution is the same as the one in Equation (2)! Thus, we get that this distribution outputs x with probability at least

$$2^{2m}\frac{1}{\alpha 2^\alpha}(\frac{\lambda}{t^2 m})^{m/\lambda}(p/4 - 2^{-2m})\frac{2}{3}(1-1/t).$$

Moreover, if one has access[18] to y, then this distribution can be sampled in time at most $\mathsf{poly}(t,n,m,\lambda)$ using a sampler whose code length is at most $O(\log tnm\lambda)$. Then, applying the coding theorem (Theorem 15) with the conditional string y, we get that

$$\mathsf{pK}^{t'}(x|y) \le -2m + \alpha\log\alpha + \log(\frac{1}{p/4 - 2^{-2m}}) + O(\frac{m}{\lambda}\log(tmn\lambda))$$

for some $t' = \mathsf{poly}(t,n,m)$. Rearranging, we get that

$$p \le 2^{-2m+\alpha-\mathsf{pK}^{t'}(x|y)+O(\frac{m}{\lambda}\log(tmn))} + 2^{-2m+2},$$

so setting $\alpha = 2m + \mathsf{pK}^{t'}(x|y) - \Delta$ gives

$$p \le 2^{-\Delta+O(\frac{m}{\lambda}\log(tmn))} + 2^{-2m+2}.$$

Finally since $\Delta \le 2m$ (by assumption), we get

$$p \le 2^{-\Delta+O(\frac{m}{\lambda}\log(tmn))} + 2^{-2m+2} \le 2^{-\Delta+O(\frac{m}{\lambda}\log(tmn))},$$

as desired. □

It remains to prove Proposition 20, which we restate below.

Proposition 20. *Assume $\lambda \ge \Omega(\log m)$. Then*

$$\Pr_{\substack{\tilde{y}\leftarrow\{0,1\}^{2m} \\ \mathcal{O}\leftarrow\mathsf{Oracle.Swapped}[y,\tilde{y}]}}[\mathsf{pK}^{t,\mathcal{O}}(x\tilde{y}) \le \alpha] \ge p/4 - 2^{-2m}$$

In fact Proposition 20 is a specific case of a more general fact, that says the analogous statement holds for all events E.

Proposition 22. *Assume $\lambda \ge \Omega(\log m)$. Let E be any set. Then*

$$\Pr_{\substack{\tilde{y}\leftarrow\{0,1\}^{2m} \\ \mathcal{O}\leftarrow\mathsf{Oracle.Swapped}[y,\tilde{y}]}}[(\mathcal{O},\tilde{y}) \in E] \ge \frac{1}{4}\Pr_{\substack{\mathcal{O}\leftarrow\mathsf{Oracle.Uniform} \\ \tilde{y}\leftarrow\mathsf{Encode}[y,\mathcal{O}]}}[(\mathcal{O},\tilde{y}) \in E] - 2^{-2m}$$

[18] Access to y is needed for step (2) where on a fresh query we may need to respond with y_i.

Proof. Throughout this proof, $\mathcal{O}$ and $\tilde{y}$ will always denote a function from 2λ bits to λ bits and an element of $\{0,1\}^{2m}$ respectively. We say a pair $(\mathcal{O}, \tilde{y})$ is *valid* if $\mathcal{O}(\tilde{y}_i) = y_i$ for all i. We say a pair $(\mathcal{O}, \tilde{y})$ is *good* if $(\mathcal{O}, \tilde{y})$ is valid and

$$\prod_{i \in [m/\lambda]} |\mathcal{O}^{-1}(y_i)| \geq 2^m/4.$$

Let E' be the subset of E consisting of elements of E that are good. Then we can directly calculate

$$\begin{aligned}
&\Pr_{\substack{\tilde{y} \leftarrow \{0,1\}^{2m} \\ \mathcal{O} \leftarrow \mathsf{Oracle.Swapped}[y,\tilde{y}]}}[(\mathcal{O}, \tilde{y}) \in E] \\
\geq &\Pr_{\substack{\tilde{y} \leftarrow \{0,1\}^{2m} \\ \mathcal{O} \leftarrow \mathsf{Oracle.Swapped}[y,\tilde{y}]}}[(\mathcal{O}, \tilde{y}) \in E'] \\
\geq &|E'| 2^{-2m} (2^{-\lambda})^{2^{2\lambda} - m/\lambda} \\
= &|E'| 2^{-\lambda 2^{2\lambda} - m}.
\end{aligned}$$

On the other hand, we have that

$$\begin{aligned}
&\Pr_{\substack{\mathcal{O} \leftarrow \mathsf{Oracle.Uniform} \\ \tilde{y} \leftarrow \mathsf{Encode}[y,\mathcal{O}]}}[(O, \tilde{y}) \in E] \\
\leq &\Pr_{\substack{\mathcal{O} \leftarrow \mathsf{Oracle.Uniform} \\ \tilde{y} \leftarrow \mathsf{Encode}[y,\mathcal{O}]}}[(O, \tilde{y}) \text{ is not good}] + \Pr_{\substack{\mathcal{O} \leftarrow \mathsf{Oracle.Uniform} \\ \tilde{y} \leftarrow \mathsf{Encode}[y,\mathcal{O}]}}[(O, \tilde{y}) \in E']
\end{aligned}$$

We bound both these terms separately. For the second term, we can directly calculate

$$\begin{aligned}
&\Pr_{\substack{\mathcal{O} \leftarrow \mathsf{Oracle.Uniform} \\ \tilde{y} \leftarrow \mathsf{Encode}[y,\mathcal{O}]}}[(O, \tilde{y}) \in E'] \\
= &\sum_{(\mathcal{O}^\star, \tilde{y}^\star) \in E'} \Pr_{\substack{\mathcal{O} \leftarrow \mathsf{Oracle.Uniform} \\ \tilde{y} \leftarrow \mathsf{Encode}[y,\mathcal{O}]}}[\mathbb{1}[\mathcal{O} = \mathcal{O}^\star \text{ and } \tilde{y} = \tilde{y}^\star] \\
= &\sum_{(\mathcal{O}^\star, \tilde{y}^\star) \in E'} 2^{-\lambda 2^{2\lambda}} \prod_{i \in [m/\lambda]} \frac{1}{|(\mathcal{O}^\star)^{-1}(y_i)|} \\
\leq &2^{-\lambda 2^{2\lambda}} \sum_{(\mathcal{O}^\star, \tilde{y}^\star) \in E'} 4 \cdot 2^{-m} \\
= &4|E'| 2^{-\lambda 2^{2\lambda} - m}
\end{aligned}$$

where the inequality comes from $(\mathcal{O}, \tilde{y})$ being good.

Next, we bound the first term

$$\begin{aligned}
&\Pr_{\substack{\mathcal{O} \leftarrow \mathsf{Oracle.Uniform} \\ \tilde{y} \leftarrow \mathsf{Encode}[y,\mathcal{O}]}}[(O, \tilde{y}) \text{ is not good}] \\
= &\Pr_{\substack{\mathcal{O} \leftarrow \mathsf{Oracle.Uniform} \\ \tilde{y} \leftarrow \mathsf{Encode}[y,\mathcal{O}]}}[\prod_{i \in [m/\lambda]} |\mathcal{O}^{-1}(y_i)| < 2^m/4] \\
\leq &\Pr_{\substack{\mathcal{O} \leftarrow \mathsf{Oracle.Uniform} \\ \tilde{y} \leftarrow \mathsf{Encode}[y,\mathcal{O}]}}[\text{ there exists } i \text{ such that } |\mathcal{O}^{-1}(y_i)| < 2^\lambda 4^{-\lambda/m}] \\
\leq &\Pr_{\substack{\mathcal{O} \leftarrow \mathsf{Oracle.Uniform} \\ \tilde{y} \leftarrow \mathsf{Encode}[y,\mathcal{O}]}}[\text{ there exists } i \text{ such that } |\mathcal{O}^{-1}(y_i)| \leq 2^\lambda (1 - \frac{1}{2m})] \\
\leq &\frac{m}{\lambda} \cdot \exp(-\frac{2^\lambda}{8m^2}) \\
\leq &2^{-2m}
\end{aligned}$$

where the first equality uses that the only way that $(\tilde{y}, \mathcal{O})$ is not valid when $\tilde{y}$ is sampled from $\mathsf{Encode}[y, \mathcal{O}]$ is if $|\mathcal{O}^{-1}(y_i) = 0|$ for some i, the second inequality comes from $4^{-\lambda/m} \le 4^{-1/m} \le (1-2/m)$, the third inequality comes from applying a Chernoff bound to the random variable $|\mathcal{O}^{-1}(y_i)|$ which is the sum of independent Bernoulli random variables with expectation 2^λ, and the last inequality comes from the assumption that $\lambda \ge \Omega(\log m)$ for a sufficiently large hidden constant.

Putting our bounds together, we have that

$$\Pr_{\substack{\mathcal{O} \leftarrow \mathsf{Oracle.Uniform} \\ \tilde{y} \leftarrow \mathsf{Encode}[y,\mathcal{O}]}}[(O, \tilde{y}) \in E] \le 4|E'|2^{-\lambda 2^\lambda - m} + 2^{-2m}$$

and that

$$\Pr_{\substack{\tilde{y} \leftarrow \{0,1\}^{2m} \\ \mathcal{O} \leftarrow \mathsf{Oracle.Swapped}[y,\tilde{y}]}}[(\mathcal{O}, \tilde{y}) \in E] \ge |E'|2^{-\lambda 2^\lambda - m}.$$

Combining the two inequalities above, we have that

$$\Pr_{\substack{\tilde{y} \leftarrow \{0,1\}^{2m} \\ \mathcal{O} \leftarrow \mathsf{Oracle.Swapped}[y,\tilde{y}]}}[(\mathcal{O}, \tilde{y}) \in E] \ge \frac{1}{4} \Pr_{\substack{\mathcal{O} \leftarrow \mathsf{Oracle.Uniform} \\ \tilde{y} \leftarrow \mathsf{Encode}[y,\mathcal{O}]}}[(O, \tilde{y}) \in E] - 2^{-2m}$$

as desired.

□

4 Reduction

Our hardness results are proved using a single reduction with different settings of parameters and amounts of non-uniformity. We describe the reduction in this section. To begin, we state the problem we reduce from: a variant of Set Cover.

4.1 Bounded Frequency Set Cover

Many restrictions of set cover are known to be NP-complete. In our case, we use a restriction that will eventually enable near-optimal additive hardness of approximation and ensure the reduction works for MCSP (for MCSP, good hardness of approximation is crucial because the best known bound on the complexity of a random function is only tight to a $(1 - o(1))$ factor).

Definition 23 (τ-Frequency Set Cover)**.** Let $\tau \in \mathbb{N}$. τ-Frequency Set Cover is the following problem:

- **Given:** a threshold $\theta \in [m]$ and subsets $S_1, \ldots, S_m$ of $[n]$ with the property that for all $i \in [n]$ the number of subsets that contain i is exactly τ (i.e., $|\{j : i \in S_j\}| = \tau$).
- **Decide:** does there exist a $J \subseteq [m]$ of size at most θ such that $\bigcup_{j \in J} S_j = [n]$.

We note that, without loss of generality, one can assume that $m \le \tau n$ (since each element of the ground set is in at most τ sets, the number of non-empty sets is at most τn). We also define some notation now that we will use later: for $i \in [n]$ and $k \in [\tau]$, let $\mathsf{Index}(i, k)$ denote the lexicographically kth j such that $i \in S_j$.

The τ-frequency set cover problem is easily seen to be equivalent to vertex cover on τ-uniform hypergraphs,[19] and near-optimal inapproximability is known for vertex cover on τ-uniform hypergraphs. Although we do not need the full strength of it, we state (a slightly weaker, easier-to-state version of) the best-known hardness of approximation.

[19] To translate a τ-frequency set cover instance to a vertex cover instance, let the vertex set be $[m]$ and for each $i \in [n]$ create a hyperedge that contains the indices of all sets that contain i.

Theorem 24 (Dinur, Guruswami, Khot, and Regev [DGKR05])**.** *Let τ be any sufficiently large integer constant. Then given an instance of τ-frequency set cover on a universe of size n where n is a power of two*[20] *it is* NP*-hard to distinguish:*

- ***YES case:*** *there is a set cover of size $\frac{2}{\tau}n$*
- ***NO case:*** *there is no set cover of $n/3$*

We will refer to the above problem as *Gap τ-Frequency Set Cover.*

4.2 Reduction

We now describe a randomized version of our reduction. We then show how to modify it to get our final deterministic reduction. For convenience, assume τ is a power of two (we have the freedom to set τ so this is without loss of generality). We will also assume we have access to oracles we call $\mathcal{O}_n$. We will later describe how to construct $\mathcal{O}_n$ from $\mathcal{O}$.

Reduction from Gap τ-Frequency Set Cover to MCSP and K^t

Parameters: a secret key length λ that is a function of n computable in time $O(\log \lambda(n))$ and that is always a power of two.

Oracles: For all powers of two n, we have oracle access to $\mathcal{O}_n : [n] \times \{0,1\}^\lambda \times \{0,1\}^{2\lambda} \to \{0,1\}^\lambda$.

Given: an instance $S_1, \ldots, S_m \subseteq [n]$ of Gap τ-Frequency Set Cover where n is a power of two

1. Pick $sk_1, \ldots, sk_m \in \{0,1\}^\lambda$ uniformly at random.
   - One can think of sk_j as a "secret key" that is associated with each set in the set cover instance.
2. For all $i \in [n]$, pick $v_i \in \{0,1\}^\lambda$ uniformly at random.
   - One can think of v_i as a random "message" value that is associated with each element of the ground set we want to cover.
3. For all $i \in [n]$ and $k \in [\tau]$, let $c_{i,k}$ be a uniformly random element of the set $\{c \in \{0,1\}^{2\lambda} : \mathcal{O}_n(i, sk_j, c) = v_i\}$ where j is the index[a] of the kth set containing i.
   - More specifically, we sample $c_{i,k}$ using rejection sampling: run $\mathcal{O}_n(i, sk_j, \cdot)$ on uniformly random inputs until it evaluates to v_i. (This is essentially a cryptographic proof of work!) We note, however, that this step could potentially not halt.
   - One can think of $c_{i,k}$ as a random "encoding" of an "encryption" of the message v_i using the secret key associated with the set S_j. Note that one can "decrypt" $c_{i,k}$ to recover v_i if one knows the secret key sk_j since $\mathcal{O}_n(i, sk_j, c_{i,k}) = v_i$.
4. Output the $4\tau n\lambda$-bit truth table of the function $f : [n] \times [\tau] \times \{0,1\} \times [2\lambda] \to \{0,1\}$ given by[b]

$$f(i,k,b,d) = \begin{cases} d\text{'th bit of } c_{i,k}, & \text{if } b = 0 \\ d\text{'th bit of } v_i, & \text{if } b = 1 \end{cases}.$$

[20] We can assume that n is a power of two by a padding argument. In particular, pad instances with m extra ground set elements until it is a power of two. Then add τ identical sets each containing all of the m new ground set elements. The size of the optimal set cover will change by exactly an additive τ. We have implicitly absorbed this additive τ into the stated hardness of approximation.

We also output some complexity threshold, but this will depend on the precise problem we are reducing to.

- Ideally, one might hope that the "optimal" way to compute f is to have memorized all the $c_{i,k}$ as well as the secret keys sk_j for all j in an optimal set cover. Then one can decrypt to find the values of v_i. This would relate the complexity of computing f to the size of the optimal set cover. This bound fails badly in the information-theoretic setting: one can instead memorize all the v_i and save on describing each $c_{i,k}$ knowing that it decrypts to v_i, but, crucially, this may require a lot of time.

[a] i.e., $j = \mathsf{Index}(i,k)$.

[b] Since each v_i is a λ-bit string, and $d \in [2\lambda]$, we must adopt a convention for what the d'th bit of v_i is when $d > \lambda$. Any reasonable choice suffices, but for concreteness we say to output 0 when this is the case.

This completes the description of the (randomized) reduction.

We now describe how to modify the above reduction to turn it into our final deterministic reduction. The deterministic reduction just simulates running the randomized reduction above, but whenever the randomized reduction asks for its i'th random bit, the deterministic reduction uses the bit given by evaluating a random oracle $\mathcal{R}_n$ (that outputs one bit and is independent from all $\mathcal{O}_n$ random oracles) on input i.

We stress that while we have described a sequence of distinct and independent oracles $\mathcal{R}_n$ and $\mathcal{O}_n$ for all powers of two n, in reality there is just a single random oracle $\mathcal{O} : \{0,1\}^\star \to \{0,1\}$ from which these other oracles are derived from. Specifically, for any power of two n, we let

$$\mathcal{O}_n(x) = \mathcal{O}(0, 1^{\log n}, 0, 1^1, 0, x) \dots \mathcal{O}(0, 1^{\log n}, 0, 1^\lambda, 0, x)$$

for any x of the appropriate length, and we let

$$\mathcal{R}_n(x) = \mathcal{O}(1, 1^n, 0, 1^{2^\lambda}, 0, x).$$

Since λ (as a function of n) is computable in time $O(\log \lambda(n))$, note that we can evaluate $\mathcal{R}_n$ and $\mathcal{O}_n$ on any input x in time at most $O(|x| + n + 2^\lambda)$ and $O(|x| + \lambda + \log n)$ respectively given oracle access to $\mathcal{O}$. It will be important (for derandomizing the reduction) that the "random" coins generated by $\mathcal{R}_n$ come from evaluating $\mathcal{O}$ on long ($> 2^\lambda$) input lengths.

4.3 Analysis

First, we analyze the running time of the reduction. The dominant factor in the running time is the amount of time it takes to generate each $c_{i,k}$ using rejection sampling.

Proposition 25. *Fix any instance of Gap τ-Frequency Set Cover. With probability at least $1 - 2^{-n2^\lambda}$ over a uniformly random oracle $\mathcal{O}$, the reduction on this instance runs in time at most $\mathsf{poly}(n, 2^\lambda)$.*

Proof. It is easy to see that every step in the reduction runs in time $\mathsf{poly}(n, 2^\lambda)$ except perhaps generating each $c_{i,k}$. The time it takes to generate $c_{i,k}$ is $\mathsf{poly}(n, 2^\lambda)$ multiplied by the number of times one needs to rejection sample. The probability that one rejection samples T times without succeeding when $\mathcal{O}_n$ and each v_i is chosen uniformly at random is at most

$$(1 - 2^{-\lambda})^T \leq e^{-T2^{-\lambda}}.$$

Thus, the proposition follows by setting T to be a sufficiently large polynomial in n and 2^λ and then union bounding over all values of $i \in [n]$ and $k \in [\tau]$. □

In Section 6, we show the following upper bounds on the complexity of f on YES instances.

Lemma 26. *On a YES instance of Gap τ-Frequency Set Cover, if the reduction outputs[21] f, then*

[21] We say this because the reduction could fail to halt, in which case we give no guarantee.

- $\mathsf{K}^{t,\mathcal{O}}(f) \leq 2\lambda n\tau + 2\lambda n/\tau + 3n\log(\tau n) + O(\log(\tau\lambda n))$ *for some* $t = \mathsf{poly}(n,\tau,\lambda)$*, and*
- *there is a constant-depth* $\mathcal{O}$*-oracle circuit for* f *of size at most*

$$(1+o(1))[2\lambda n\tau\frac{1}{\log(n\tau\log(2\lambda))} + 2\lambda n\frac{1}{\tau\log 2n/\tau} + 2\log(\tau n)n\frac{1}{\log n} + O(\tau+\lambda+\log n)]$$

In Section 7, we show the following lower bounds on the complexity of f on NO instances.

Lemma 27. *Assume* $\lambda \geq \Omega_\tau(\log n)$*. Fix any NO instance of Gap* τ*-Frequency Set Cover. With probability at least* $1-2^{-(1-o_\tau(1))n\lambda/8}$ *over the choice of* $\mathcal{O}$ *the reduction on this instance outputs an* f *satisfying*

- $\mathsf{K}^{t,\mathcal{O}}(f) \geq 2\lambda n\tau + n\lambda/8$ *where* $t = 2^{\lambda/O(\tau)}$ *, and*
- *the* $\mathcal{O}$*-oracle circuit complexity of* f *is at least* $\frac{2\lambda n\tau + n\lambda/8}{\log(2\lambda n\tau+n\lambda/8)}$.

5 Hardness Results

Using the reduction and analysis in Section 4, we prove our main theorems. We restate the theorems below for the reader's convenience.

Theorem 7. *Let* $p(\cdot)$ *be any polynomial satisfying* $p(t) \geq t$*. There is a deterministic polynomial-time algorithm* A *such that with probability one (over a random oracle* $\mathcal{O}$*),* $A^{\mathcal{O}}$ *is a many-one reduction from* 3-SAT *to the promise problem of, given a string* x*, outputting:*

- ***YES:*** *if* $\mathsf{K}^{t,\mathcal{O}}(x) \leq \theta(|x|)$
- ***NO:*** *if* $\mathsf{K}^{p(t),\mathcal{O}}(x) \geq \theta(|x|) + \Omega(|x|)$

where t *is some polynomial in* $|x|$ *and* θ *is some function of* $|x|$*.*

Proof. We run the reduction in Section 4.2, setting τ to be some sufficiently large power of two and setting λ to be the smallest power of two greater than $\gamma\log n$ for some sufficiently large constant γ (that can depend on τ).

We will show that with probability one over the choice of random oracle that this reduction is correct (i.e., it runs in polynomial time and is sound and complete) for all but finitely many input lengths. To do this we show that for all n, the probability that there exists an instance of Gap τ-Frequency Set Cover on the ground set $[n]$ on which the reduction fails to be efficient, sound, or complete is at most $2^{-\Omega(n)}$. Then the Borel-Cantelli lemma implies that with probability 1 over the choice of $\mathcal{O}$ that the reduction fails on at most finitely many input lengths, proving the theorem.

Now fix any instance of Gap τ-Frequency Set Cover on the ground set $[n]$ where n is a power of two. By Proposition 25, the probability that the reduction on this instance does not finish in time

$$\mathsf{poly}(n, 2^\lambda) = \mathsf{poly}(n)$$

is at most 2^{-n2^λ}.

If this instance is a YES instance and the reduction outputs f, Lemma 26 implies

$$\mathsf{K}^{t,\mathcal{O}}(f) \leq 2\lambda n\tau + 2\lambda n/\tau + 3n\log(\tau n) + O(\log(\tau\lambda n))$$

for some $t = \mathsf{poly}(n,\lambda,\tau)$.

On the other hand, if this instance is a NO instance and the reduction outputs f, then Lemma 27 says that with probability $1-2^{-(1-o(1))n\lambda/8}$ over the choice of $\mathcal{O}$ that

$$\mathsf{K}^{t',\mathcal{O}}(f) \geq 2\lambda n\tau + n\lambda/8$$

where

$$t' \geq 2^{\lambda/O(\tau)} \geq 2^{\gamma\log(n)/O(\tau)} = n^{\gamma/O(\tau)} \geq p(t)$$

by setting γ to be sufficiently large compared to τ and using that $t = \mathsf{poly}(n, \lambda, \tau)$.

Observe that, when the two bounds hold, the difference between the complexity of f on NO instances and YES instances is at least

$$\begin{aligned}
&2\lambda n\tau + n\lambda/8 - (2\lambda n\tau + 2\lambda n/\tau + 3n\log(\tau n) + O(\log(\tau\lambda n))) \\
=&n\lambda(1/8 - 2/\tau) - 3n\log(\tau n) - O(\log(\tau\lambda n)) \\
\geq&\Omega_\tau(n\lambda) \\
\geq&\Omega_{\tau,\gamma}(|f|)
\end{aligned}$$

by setting τ and γ to be sufficiently large constants.

Thus, putting it all together, the probability (over $\mathcal{O}$) that there exists an instance of Gap τ-Frequency Set Cover on the ground set $[n]$ where the reduction fails (either it does not run in time $\mathsf{poly}(n)$ or the desired lower bound does not hold for a NO instance), is at most

$$n^{\tau m}(2^{-(1-o(1))n\lambda/8} + 2^{-n2^\lambda}) \leq n^{\tau^2 n}2^{-(1-o(1))n\lambda/8} \leq 2^{-(1-o(1))n\lambda/8+\tau^2 n\log n} \leq 2^{-\Omega(n)}$$

where we are union bounding over all $n^{\tau m}$ many instances of τ-frequency set cover and using that $m \leq \tau n$ and setting γ to be a constant sufficiently larger than τ. □

Next, we prove hardness for MCSP.

Theorem 6. *There is a deterministic polynomial-time algorithm A such that with probability one (over a random oracle $\mathcal{O}$) $A^{\mathcal{O}}$ is a many-one reduction from* 3-SAT *to the promise problem of, given a string x, outputting:*

- ***YES:*** *if the $\mathcal{O}$-oracle circuit complexity of x is at most $\theta(|x|)$ (and moreover, this is witnessed by a constant-depth circuit)*
- ***NO:*** *if the $\mathcal{O}$-oracle circuit complexity of x is at least $\theta(|x|) + \Omega(\frac{|x|}{\log|x|})$*

where θ is some function of $|x|$.

Proof. The proof is essentially the same as the proof of Theorem 7. As before, we set τ to be some sufficiently large power of two, and after choosing τ we set λ to be the smallest power of two greater than $\gamma\log n$ for some sufficiently large constant γ (that can depend on τ).

The only change to the proof is the complexity bounds on YES and NO instances. On YES instances, Lemma 26 says that there is a constant-depth $\mathcal{O}$-oracle for f of size at most

$$\begin{aligned}
&(1+o(1))[2\lambda n\tau\frac{1}{\log(n\tau\log(2\lambda))} + 2\lambda n\frac{1}{\tau\log(2\lambda n/\tau)} + 2\log(\tau n)n/\log n + O(\tau+\lambda+\log n)] \\
\leq&(1+o_{\tau,\gamma}(1))\frac{1}{\log n}[2\lambda n\tau + 2\lambda n/\tau]
\end{aligned}$$

by setting γ to be sufficiently large compared to τ.

On the other hand, Lemma 27 says with probability $1 - 2^{-(1-o(1))n\lambda/8}$ on NO instances we get that the $\mathcal{O}$-oracle circuit complexity of f is at least

$$\begin{aligned}
&\frac{2\lambda n\tau + n\lambda/8}{\log(2\lambda n\tau + n\lambda/8)} \\
\geq&(1-o_{\tau,\lambda}(1))\frac{2\lambda n\tau + n\lambda/8}{\log(n)}.
\end{aligned}$$

Thus, the difference between the complexity of f on NO instances (when the bound holds) and YES instances is at least

$$\begin{aligned}
&(1 - o_{\tau,\gamma}(1))\frac{1}{\log n}[2\lambda n\tau + n\lambda/8 - 2\lambda n\tau - 2\lambda n/\tau] \\
\geq &(1 - o_{\tau,\gamma}(1))\frac{1}{\log n} n\lambda(1/8 - 2/\tau) \\
\geq &\Omega_{\tau,\gamma}(\frac{|f|}{\log |f|})
\end{aligned}$$

by setting τ and γ to be sufficiently large constants. The remainder of the proof is the same as the proof of Theorem 7. □

One deficiency in our reduction is that it runs in time $\mathsf{poly}(n, 2^\lambda)$. As a result, one can only afford to set $\lambda = O(\log n)$ in order to get a polynomial time reduction. This limits the time gap between the YES and NO instances of time-bounded Kolmogorov complexity in Theorem 7. At a high level, the reduction spends most of its time "finding collisions" in $\mathcal{O}$. We show one can replace the exponential dependence on λ with a polynomial dependence if one uses polynomial-time *non-uniform* reductions. Essentially, the idea is that a non-uniform reduction can instead hardcode in collisions.

Theorem 8. *Let $\epsilon > 0$ be any sufficiently small constant. Let $\mathcal{O}$ be a uniformly random oracle. With probability one there is a non-uniform polynomial-time many-one reduction (where the circuit implementing the reduction does* not *get oracle access to $\mathcal{O}$) from* SAT *to the promise problem of, given a string x, outputting:*

- ***YES:*** *if $\mathsf{K}^{t,\mathcal{O}}(x) \leq \theta(|x|)$*
- ***NO:*** *if $\mathsf{K}^{2^{|x|^{1-\epsilon}},\mathcal{O}}(x) \geq \theta(|x|) + \Omega(|x|)$*

where t is a polynomial in x and θ is some function of $|x|$.

Proof. We make a slight modification to the original reduction. The only difference between this reduction and the original reduction is in step 3, highlighted in blue. For comparison, the old step 3 is also included highlighted in red.

Modified Reduction from Gap τ-Frequency Set Cover to K^t

Parameters: a secret key length λ that is a function of n and that is always a power of two.

Oracles: For all $n \in \mathbb{N}$, we have oracle access to a function $\mathcal{O}_n : [n] \times \{0,1\}^\lambda \times \{0,1\}^{2\lambda} \to \{0,1\}^\lambda$.

Given: a set cover instance $S_1, \ldots, S_m \subseteq [n]$ with τ-frequency where n is a power of two

1. Pick $sk_1, \ldots, sk_m \in \{0,1\}^\lambda$ uniformly at random.
2. For all $i \in [n]$ pick $v_i \in \{0,1\}^\lambda$ uniformly at random.
3. (Previously: For all $i \in [n]$ and $k \in [\tau]$, let $c_{i,k}$ be a uniformly random element of the set $\{c \in \{0,1\}^{2\lambda} : \mathcal{O}_n(i, sk_j, c) = v_i\}$ where $j = \mathsf{Index}(i,k)$.
   (a) For all $i \in [n]$ and $j \in [m]$, let $c^\star_{i,j}$ be a uniformly random element of the set $\{c \in \{0,1\}^{2\lambda} : \mathcal{O}_n(i, sk_j, c) = v_i\}$.
   (b) For all $i \in [n]$ and $k \in [\tau]$, let $c_{i,k} = c^\star_{i,j}$ where $j = \mathsf{Index}(i,k)$.

4. Output the $4\tau n\lambda$-bit truth table of the function $f : [n] \times [\tau] \times \{0,1\} \times [2\lambda] \to \{0,1\}$ given by

$$f(i,k,b,d) = \begin{cases} d\text{'th bit of } c_{i,k}, & \text{if } b = 0 \\ d\text{'th bit of } v_i, & \text{if } b = 1 \end{cases}.$$

We also output some complexity threshold, which we set later.

We use the same trick as before to make this reduction deterministic. The deterministic reduction simulates running the randomized reduction above, but whenever the randomized reduction asks for its i'th random bit, the deterministic reduction just uses the bit given by evaluating a random oracle $\mathcal{R}_n$ (that outputs one bit and is independent from all $\mathcal{O}_n$ random oracles) on input i. Importantly, note that the sequence of "random" bits we deterministically generate is, by construction, the same for any instance of τ-frequency set cover on the same ground set $[n]$. We will use this observation later. This completes the description of the new reduction.

It is easy to see that this reduction is functionally equivalent to the old reduction. The only difference is that we generate some extra $c^\star_{i,j}$ values that are not used. Thus, we can reuse our analysis of the previous reduction to show that with probability one over the choice of $\mathcal{O}$ this is indeed a reduction from Gap τ-Frequency Set Cover to K^t. However, we will now consider a different setting of λ.

Set τ to be a sufficiently large power of two and set λ to be the smallest power of two greater than n^γ for some sufficiently large constant γ (that can depend on ϵ and τ). Our setting of λ makes the reduction run in exponential time, but we will fix this later.

On a YES instance, Lemma 26 implies

$$\mathsf{K}^{t,\mathcal{O}}(f) \le 2\lambda n\tau + 2\lambda n/\tau + 3n\log(\tau n) + O(\log(\tau\lambda n))$$

for some $t = \mathsf{poly}(n, \lambda, \tau)$.

On the other hand, using Lemma 27 and the Borel-Cantelli lemma, with probability one over the choice of $\mathcal{O}$ for all NO instances where n is sufficiently large we have that

$$\mathsf{K}^{t',\mathcal{O}}(f) \ge 2\lambda n\tau + n\lambda/8$$

where

$$t' \ge 2^{\lambda/O(\tau)} \ge 2^{|f|^{1-\epsilon}}$$

using that $|f| = 4\tau n\lambda = 4\tau n^{1+\gamma}$ and setting γ to be sufficiently large (depending on ϵ and τ). Thus, as before, we have that the complexity gap between the YES and NO case is at least $\Omega_{\gamma,\tau}(|f|)$. But now the time gap is significantly larger.

The key new observation is that on every instance with the same ground set $[n]$, the values of the secret keys $sk_1, \ldots, sk_m$ and the messages $v_1, \ldots, v_n$ and the strings $c^\star_{i,j}$ for all $i \in [n]$ and $j \in [m]$ that the reduction generates are all the same! This is because, as we noted previously, the "random coins" the reduction deterministically generates using $\mathcal{R}_n$ are the same on any instance with ground set $[n]$. Looking at the code for the reduction, this means that all the $sk_1, \ldots, sk_m$ and $v_1, \ldots, v_n$ and $c^\star_{i,j}$ will be the same.

Thus, we can obtain a non-uniform reduction by hardcoding all these values into the reduction. The new reduction is extremely simple. We state it below.

Non-Uniform Reduction from Gap τ-Frequency Set Cover to K^t

Parameters: a secret key length λ that is a power of two.

Advice: hardcoded values $v_i \in \{0,1\}^\lambda$ and $c^\star_{i,j} \in \{0,1\}^{2\lambda}$ for all $i \in [n]$ and $j \in [m]$

Given: a τ-frequency set cover instance $S_1, \ldots, S_m \subseteq [n]$ where n is a power of two

1. Output the complexity threshold $2\lambda n\tau + n\lambda/8 - 1$ and the $4\tau n\lambda$-bit truth table of the function

$f : [n] \times [\tau] \times \{0,1\} \times [2\lambda] \to \{0,1\}$ given by

$$f(i,k,b,d) = \begin{cases} d\text{'th bit of } c^{\star}_{i,\mathsf{Index}(i,k)}, & \text{if } b = 0 \\ d\text{'th bit of } v_i, & \text{if } b = 1 \end{cases}.$$

It is easy to see that after all this information is hardcoded in, the reduction is computable in (non-uniform) polynomial time. In fact, it is computable by a non-uniform AC^0 circuit. □

The same argument also gives a non-uniform reduction for MCSP. However, unlike for K^t, we do not appear to improve any parameters over the uniform reduction.

Theorem 28. *Let $\mathcal{O}$ be a uniformly random oracle. With probability one there is a non-uniform polynomial-time many-one reduction (where the reduction does* not *get oracle access to $\mathcal{O}$) from* SAT *to the promise problem of, given a string x, outputting:*

- ***YES:*** *if the $\mathcal{O}$-oracle circuit complexity of x is at most $\theta(|x|)$*
- ***NO:*** *if the $\mathcal{O}$-oracle circuit complexity of x is at least $\theta(|x|) + \Omega(\frac{|x|}{\log |x|})$*

where θ is some function of $|x|$.

6 Reduction Completeness: Complexity Upper Bounds

In this section, we prove Lemma 26, which we restate below.

Lemma 26. *On a YES instance of Gap τ-Frequency Set Cover, if the reduction outputs*[22] *f, then*

- *$\mathsf{K}^{t,\mathcal{O}}(f) \le 2\lambda n\tau + 2\lambda n/\tau + 3n\log(\tau n) + O(\log(\tau\lambda n))$ for some $t = \mathsf{poly}(n,\tau,\lambda)$, and*
- *there is a constant-depth $\mathcal{O}$-oracle circuit for f of size at most*

$$(1+o(1))[2\lambda n\tau\frac{1}{\log(n\tau\log(2\lambda))} + 2\lambda n\frac{1}{\tau\log 2n/\tau} + 2\log(\tau n)n\frac{1}{\log n} + O(\tau+\lambda+\log n)]$$

Proof. Fix any YES instance. Let $S_{j_1},\dots,S_{j_{OPT}}$ be an optimal set cover. Since this is a YES instance, $OPT \le 2n/\tau$. We will argue about the cases of time-bounded Kolmogorov complexity and circuit complexity separately in the following subsections.

6.1 Time-Bounded Kolmogorov Complexity

To get a description of f we hardcode in:

- $c_{i,k} \in \{0,1\}^{2\lambda}$ for all $i \in [n]$ and $k \in [\tau]$. Cost: $2\tau n\lambda$ bits.
- $sk_{j_1},\dots,sk_{j_{OPT}} \in \{0,1\}^{\lambda}$. Cost: $OPT \cdot \lambda$ bits.
- For each element of $i \in [n]$, a tuple (i,q,k) with the property that $i \in S_{j_q}$ and $j_q = \mathsf{Index}(i,k)$. Such a tuple must exist because $S_{j_1},\dots,S_{j_{OPT}}$ is an optimal set cover. Cost: at most $3n\log(\tau n)$ bits.
- The values n, λ, OPT, and τ as well as some other minor encoding overhead. Cost: $O(\log\tau\lambda n)$ bits.

[22] We say this because the reduction could fail to halt, in which case we give no guarantee.

Thus, we have hardcoded in everything needed to compute f except the values v_i for all $i \in [n]$. We can compute any v_i using the above information by first retrieving the tuple (i, q, k) starting with i (recall this tuple satisfies $i \in S_{j_q}$ and $j_q = \mathsf{Index}(i, k)$), second retrieving the secret key sk_{j_q} and then lastly "decrypting" $c_{i,k}$ by running $\mathcal{O}_n(i, sk_{j_q}, c_{i,k})$ which evaluates to v_i by construction.

This gives a description of f of size at most

$$2\lambda n\tau + \lambda OPT + 3n\log(\tau n) + O(\log(\lambda n)) \leq 2\lambda n\tau + 2\lambda n/\tau + 3n\log(\tau n) + O(\log(\tau\lambda n))$$

Moreover, this description runs in time $\mathsf{poly}(n, \tau, \lambda)$. (Recall, we can query $\mathcal{O}_n$ on an input x given access to $\mathcal{O}$ in time $O(|x| + \lambda + \log n)$.)

6.2 Circuit Complexity

Here we use Lupanov's construction [Lup58] of near-optimal circuits for random Boolean functions. Lupanov's original construction only considers the case where the Boolean function has a single output, but it readily generalizes to the multi-output case, which we state below.[23] For completeness, we give a proof of Theorem 29 in Appendix A.

Theorem 29 (Multi-Output Lupanov)**.** *Let $f : \{0,1\}^n \to \{0,1\}^m$ with $m \geq 2$. Then there is a constant-depth circuit computing f of size $(1 + o(1))\frac{m2^n}{n\log m}$.*

To construct a circuit for f, we use Lupanov's result to build the following subcircuits (below it is useful to recall that n, τ and λ are all powers of two):

1. A subcircuit that given $(i, k) \in [n] \times [\tau]$ outputs $c_{i,k}$. This is a function with $\log(n\tau)$ bits of inputs and 2λ bits of output, so it can be implemented by a circuit of size at most

$$(1 + o(1))2\lambda \cdot \frac{2^{\log(n\tau)}}{\log(n\tau\log(2\lambda))} = (1 + o(1))2\lambda n\tau\frac{1}{\log(n\tau\log(2\lambda))}$$

2. A subcircuit that given $q \in [OPT]$ outputs sk_{j_q}. The input length is[24] $\log OPT$ and the output length is λ. Thus, it can be implemented by a circuit of size

$$(1 + o(1))\lambda\frac{2^{\log OPT}}{\log OPT} = (1 + o(1))\lambda\frac{OPT}{\log OPT}.$$

3. A subcircuit that given $i \in [n]$ outputs (q, k) such that $i \in S_{j_q}$ (the existence of such a q is guaranteed by being a set cover) and $j_q = \mathsf{Index}(i, k)$. It can be implemented by a circuit of size

$$(1 + o(1))2\log(\tau n)n/\log n.$$

To compute f using these subcircuits we can use the following procedure on input (i, k, b, d):

1. Run subcircuit (3) on input i above to obtain $(q, k^\star)$ satisfying $i \in S_{j_q}$ and $j_q = \mathsf{Index}(i, k^\star)$.

2. Run subcircuit (2) on input q to obtain $sk^\star = sk_{j_q}$

3. If $b = 0$, run subcircuit (1) on input (i, k) to obtain $c_{i,k}$. If $b = 1$, run subcircuit (1) on input $(i, k^\star)$ to obtain $c_{i,k^\star}$. For efficiency, it is important that we only make one call to subcircuit (1), so the branching on $b = 0$ and $b = 1$ will happen *before* subcircuit (1) is run to determine what input to run it on.

[23] For the setting of parameters in our reduction, it actually suffices to just use a naive way of constructing multi-output circuits where one just treats each output bit as an independent single-output function and apply Lupanov's construction for each one. This will not be a near-optimal circuit, but for our setting of parameters, this loss is tolerable because of the hardness of approximation we have. Nevertheless, we decided to apply Lupanov's near-optimal construction here.

[24] This assumes that OPT is a power of two, which may not be true. Luckily, in our analysis, we will substitute OPT with the upper bound $n\tau/2$ which is a power of two, so this will not affect our final calculation.

4. If $b = 0$, then output the d'th bit of $c_{i,k}$.

5. Otherwise, $b = 1$, so run $\mathcal{O}$ on input $(i, sk^\star, c_{i,k^\star})$ to get v_i and output the d'th bit of v_i.

Observe that this procedure requires at most one call to each of the subcircuits and can be implemented by a circuit of size $O(\tau + \lambda + \log n)$ plus the complexity of each subcircuit.

Thus, using that $OPT \leq 2n/\tau$, the circuit complexity of f is at most

$$\begin{aligned}&(1+o(1))[2\lambda n\tau\frac{1}{\log(n\tau\log(2\lambda))} + \lambda OPT/\log OPT + 2\log(\tau n)n/\log n + O(\tau+\lambda+\log n)]\\ =&(1+o(1))[2\lambda n\tau\frac{1}{\log(n\tau\log(2\lambda))} + 2\lambda n/\tau\frac{1}{\log 2n/\tau} + 2\log(\tau n)n/\log n + O(\tau+\lambda+\log n)]\end{aligned}$$

Finally, we remark that this circuit can also be made to have constant depth. This is because Lupanov's construction has constant depth.

□

7 Reduction Soundness: Complexity Lower Bounds

In this section, we will prove Lemma 27. To do this, we will prove a more general result that bounds the probability a deterministic adversary with a limited number of $\mathcal{O}$-oracle queries outputs f.

Lemma 30. *Assume $\lambda \geq \Omega_\tau(\log n)$. Fix any (deterministic) decision tree P that makes $q \leq 2^{\lambda/O(\tau)}$ oracle queries of length at most $2^{\lambda/O(\tau)}$ to $\mathcal{O}$ and then outputs a string. Fix any NO instance of Gap τ-Frequency Set Cover. Let f be the output of the reduction with oracle $\mathcal{O}$ on this instance.*[25] *Then*

$$\Pr_{\mathcal{O}}[P^{\mathcal{O}} \text{ outputs } f] \leq 2^{-(1-o(1))(2\lambda n\tau + n\lambda/4)}.$$

We will prove Lemma 30 in Section 7.1. Assuming that Lemma 30 is true, we prove Lemma 27, which we restate below.

Lemma 27. *Assume $\lambda \geq \Omega_\tau(\log n)$. Fix any NO instance of Gap τ-Frequency Set Cover. With probability at least $1 - 2^{-(1-o_\tau(1))n\lambda/8}$ over the choice of $\mathcal{O}$ the reduction on this instance outputs an f satisfying*

- *$\mathsf{K}^{t,\mathcal{O}}(f) \geq 2\lambda n\tau + n\lambda/8$ where $t = 2^{\lambda/O(\tau)}$, and*
- *the $\mathcal{O}$-oracle circuit complexity of f is at least $\frac{2\lambda n\tau + n\lambda/8}{\log(2\lambda n\tau + n\lambda/8)}$.*

Proof. Let s be a parameter we set later. First, we show the $\mathsf{K}^{t,\mathcal{O}}$ lower bound. The number of descriptions of length at most s is at most 2^{s+1}. Any program that runs in time $2^{\lambda/O(\tau)}$ can make at most $2^{\lambda/O(\tau)}$ queries to $\mathcal{O}$ and each of these queries will be of length at most $2^{\lambda/O(\tau)}$. Thus, using Lemma 30, the probability that there is a description of size at most s that computes f in time $2^{\lambda/O(\tau)}$ is at most

$$2^{s+1}2^{-(1-o(1))(2\lambda n\tau + n\lambda/4)}.$$

Setting $s = 2\lambda n\tau + n\lambda/8$, we get that this probability is at most

$$2^{-(1-o_\tau(1))n\lambda/8}.$$

Next, we argue about circuit complexity. The number of fan-in two, AND, OR, NOT gate circuits of size at most $s \geq n$ is (e.g. [FM05]) at most

$$2^{(1+o(1))s\log s}.$$

[25] Note that the reduction might not output any f (it may not halt). In the probability bound below, we take "$P^{\mathcal{O}}$ outputs f" to mean that the reduction halts and outputs f and $P^{\mathcal{O}}$ also outputs f.

Setting $s = \frac{2\lambda n\tau + n\lambda/8}{\log(2\lambda n\tau + n\lambda/8)}$, we get that the number of circuits of size at most s is at most

$$2^{(1+o(1))(2\lambda n\tau + n\lambda/8)}$$

Any circuit of size s can make at most s oracle queries and every query will be of length at most s. By setting the hidden constant large enough in the assumption that $\lambda \geq \Omega_\tau(\log n)$, we get that $s \leq 2^{\lambda/O(\tau)}$. Thus, by Lemma 30, the probability that there is a circuit of size at most s that computes f is at most

$$2^{(1+o(1))(2\lambda n\tau + n\lambda/8)} 2^{-(1-o(1))(2\lambda n\tau + n\lambda/4)} \leq 2^{-(1-o_\tau(1))n\lambda/8}$$

□

7.1 Proof of Lemma 30

In this subsection, we will prove Lemma 30 except for some straightforward probability calculations that we defer to Section 7.2.

Our argument is divided into five steps. In each step, we reveal some of the randomness used to generate $\mathcal{O}_n$, $\mathcal{R}_n$, and f (which are all implicitly derived from the random choice of $\mathcal{O}$) and analyze a few important random variables. In the last step, $\mathcal{O}_n$, $\mathcal{R}_n$, and f will be fully revealed, and we will show that the probability that $P^{\mathcal{O}}$ outputs f is small.

Step 1: Reveal only the values of $\mathcal{O}$ and $\mathcal{O}_n$ that P queries.

In step 1, we run P with the random oracle $\mathcal{O}$. Whenever P queries $\mathcal{O}$, we respond with a uniformly random bit b and set $\mathcal{O}$ on that input to that value b (this is a common random oracle technique called lazy query evaluation).

To simplify our analysis, if the program happens to query a point that corresponds to one of the bits of $\mathcal{O}_n(x)$ for some x, then we will reveal all the bits of $\mathcal{O}_n(x)$. More specifically, recall that

$$\mathcal{O}_n(x) = \mathcal{O}(1, 1^{\log n}, 0, 1^1, 0, x) \ldots \mathcal{O}(1, 1^{\log n}, 0, 1^\lambda, 0, x).$$

If P ever queries a point of the form $(1, 1^{\log n}, 0, 1^i, 0, x)$ for some $i \in [\lambda]$, then we will pretend the program also queries $(1, 1^{\log n}, 0, 1^j, 0, x)$ for all $j \in [\lambda]$ and reveal all corresponding values of $\mathcal{O}$.

After step 1 finishes, the output of $P^{\mathcal{O}}$ is determined. We interpret the output of $P^{\mathcal{O}}$ as the truth table of a Boolean function $f' : [n] \times [\tau] \times \{0,1\} \times [2\lambda] \to \{0,1\}$. (If the output of $P^{\mathcal{O}}$ is not the truth table of such a function, then the probability that f' equals the f is zero and we are done. So we assume this is not the case.) For all $i \in [n]$ and $k \in [\tau]$, we let

- $c'_{i,k} \in \{0,1\}^{2\lambda}$ denote the string whose dth bit is given by $f'(i,k,0,d)$, and
- $v'_i \in \{0,1\}^\lambda$ denote the string whose dth bit is given by $f'(i,k,1,d)$.

Before we move on to the next step, we introduce some notation and a few random variables that will be useful later. Let Q denote the set of strings that the value of $\mathcal{O}_n$ has been revealed on (the size of Q is at most q).

We say a prefix[26] $i \in [n]$ has a *w-collision*, if there is a v such that

$$|\{(i, sk, c) \in Q : \mathcal{O}_n(i, sk, c) = v\}| \geq w.$$

Let C_w be the random variable given by the number of prefixes $i \in [n]$ with a w-collision. We show tail bounds on C_w.

Proposition 31. *The probability (over the randomness in step 1) that $C_w \geq t_w$ for all $2 \leq w \leq \tau$ is at most*

$$2^{\tau n \log(2\tau q) - \sum_{2 \leq w \leq \tau} t_w \lambda}.$$

We defer the proof of Proposition 31, which is a direct calculation, to Section 7.2.

[26] We call this a prefix because i is the first input taken by f.

Step 2: Reveal the values of $\mathcal{R}_n$ that determine the secret keys.

Recall, the ith "random" coin used in the reduction is deterministically generated by the value of

$$\mathcal{R}_n(i) = \mathcal{O}(0, 1^n, 0, 1^{2^\lambda}, 0, i)$$

In step 2, we reveal all the values of $\mathcal{O}$ needed to "randomly sample" the values of $sk_1, \ldots, sk_m$ in the reduction.

After step 2 finishes, we let $skHit \subseteq [m]$ be the set given by

$$skHit = \{k \in [m] : \text{there exists } i \text{ and } c \text{ such that } (i, sk_k, c) \in Q\ \}.$$

Intuitively, $skHit$ is the collection of set indices that P "successfully found the secret keys for."

Crucially, the values of $\mathcal{R}_n$ are determined by running $\mathcal{O}$ on inputs of length greater than 2^λ. On the other hand, P only queries $\mathcal{O}$ on inputs of length at most $2^{\lambda/O(\tau)}$. Thus, the values $sk_1, \ldots, sk_m$ are still uniformly random even conditioned on any information revealed in step 1. Using this, we can show a probability bound on $|skHit|$ using tail bounds on the binomial distribution. We defer the proof to Section 7.2.

Proposition 32. *The probability (over the randomness in step 2 conditioned on all the information revealed in step 1) that $|skHit| = t$ is at most $2^{-t\lambda + m\log(4mq)}$.*

We introduce one more random variable. For any $i \in [n]$ and $k \in [\tau]$, we say $c'_{i,k}$ is *confirmed* if $(i, sk_{\mathsf{Index}(i,k)}, c'_{i,k})$ is in Q and $\mathcal{O}_n(i, sk_{\mathsf{Index}(i,k)}, c'_{i,k}) = v'_i$. Intuitively, this means that based on queries that P has made, it has confirmed that $c'_{i,k}$ will decrypt to v'_i. Let B be the number (i,k) tuples such that $c'_{i,k}$ is not confirmed.

We will argue that B must be large. Let $Missed \subseteq [n]$ be the set

$$Missed = [n] \setminus \bigcup_{d \in skHit} S_d.$$

In other words, $Missed$ is the subset of the ground set that is not covered by sets whose indices are in $skHit$. Intuitively, this corresponds to the set of ground set elements for which P does not know the "secret keys" needed to decrypt the message corresponding to the ground set element.

We lower bound B in terms of the size of $Missed$ and the number of collisions $P^{\mathcal{O}}$ finds.

Proposition 33.

$$B \geq n(\tau - 1) + |Missed| - \sum_{w=2}^{\tau} C_w$$

Proof. The total number of confirmed queries is at most

$$\begin{aligned}
&\sum_{i \in [n]} \sum_{k \in [\tau]} \mathbb{1}[c_{i,k} \text{ is confirmed}] \\
=& \sum_{i \in [n] \setminus Missed} \sum_{k \in [\tau]} \mathbb{1}[c_{i,k} \text{ is confirmed}] \\
\leq& \sum_{i \in [n] \setminus Missed} 1 + \mathbb{1}[i \text{ has a 2-collision}] + \cdots + \mathbb{1}[i \text{ has a } \tau\text{-collision}] \\
\leq& n - |Missed| + \sum_{w=2}^{\tau} C_w
\end{aligned}$$

where the first equality comes from Proposition 34 below and the middle inequality comes from Proposition 35 below. Using this, we get that

$$B = \sum_{i \in [n], k \in [\tau]} \mathbb{1}[c_{i,k} \text{ is not confirmed}] \geq n(\tau - 1) + |Missed| - \sum_{w=2}^{\tau} C_w.$$

□

Proposition 34. *Let $i \in [n]$ and $k \in [\tau]$. If $i \in Missed$, then $c'_{i,k}$ is not confirmed.*

Proof. We prove the contrapositive. Suppose $c'_{i,k}$ is confirmed. Then (by definition) $(i, sk_{\mathsf{Index}(i,k)}, c'_{i,k})$ is in Q. Then $\mathsf{Index}(i,k)$ is in $skHit$ (by construction of $skHit$). Then $S_{\mathsf{Index}(i,k)}$ contains i and $\mathsf{Index}(i,k) \in skHit$, so $i \notin Missed$. □

Proposition 35. *Fix i. If $c'_{i,k}$ is confirmed for w distinct values of k, then there is a w-collision on i.*

Proof. For every $c'_{i,k}$ that is confirmed we have that

$$\mathcal{O}(i, sk_k, c'_{i,k}) = v'_i$$

and $(i, sk_k, c'_{i,k}) \in Q$. Thus, if $c'_{i,k}$ is confirmed for w distinct values of k, then there is a w-collision on i. □

Step 3: Reveal the oracle $\mathcal{O}_n$ on inputs to corresponding to each $c'_{i,k}$

In step 3, we reveal, for all $i \in [n]$ and $k \in [\tau]$, the value of $\mathcal{O}_n$ on the input $(i, sk_{\mathsf{Index}(i,k)}, c'_{i,k})$. In other words, for any $i \in [n]$ and $k \in [\tau]$, if $\mathcal{O}_n$ on input $(i, sk_{\mathsf{Index}(i,k)}, c'_{i,k})$ was not already revealed before step 3, it is set to a uniformly random element of $\{0,1\}^\lambda$. This completes the description of step 3.

After step 3 finishes, we define the following event. We say f' is a *valid encoding* if for all $i \in [n]$ and $k \in [\tau]$ we have $\mathcal{O}_n(i, sk_{\mathsf{Index}(i,k)}, c'_{i,k}) = v'_i$. Intuitively, this says that all the encryptions $c'_{i,k}$ correctly decrypt to v'_i. (Note that any f output by the reduction will have the valid encoding property by construction.)

We can bound the probability that f' is a valid encoding using the B random variable.

Proposition 36. *Conditioned on the value of B and all the information revealed in prior steps, the probability (over the randomness in step 3) that f' is a valid encoding is at most $2^{-B\lambda}$.*

Proof. For each $c'_{i,k}$ that is not confirmed, the probability that $\mathcal{O}(i, sk_{\mathsf{Index}(i,k)}, c'_{i,k}) = v'_i$ is at most $2^{-\lambda}$ (either the point was not queried before, in which case the probability is exactly $2^{-\lambda}$ or it was queried before in which case the probability is zero since it must be that $\mathcal{O}(i, sk_{\mathsf{Index}(i,k)}, c'_{i,k}) \neq v'_i$ for it not to be confirmed).

Since the non-zero probabilities are independent, we get that the probability of f' being a valid encoding is at most $2^{-B\lambda}$. □

Combining the lower bound on B in Proposition 33 with the probability bounds on $|skHit|$ and C_w given in Proposition 32 and Proposition 31, we can bound the probability that f' is a valid encoding. The proof is a somewhat tedious calculation, and we defer it to Section 7.2.

Proposition 37. *The probability (over the randomness in steps 1, 2, and 3) that f' is a valid encoding is at most*

$$2^{-\lambda n(\tau-1) - OPT \cdot \lambda + m \log(4mq) + \tau n \log(\tau q) + \tau \log(n) + \log m}.$$

Here OPT denotes the size of the optimal set cover.

Step 4: Reveal the rest of $\mathcal{O}_n$.

We now reveal $\mathcal{O}_n$ on all remaining inputs. In other words, we set the value of $\mathcal{O}_n$ on any input not already set to a uniformly random element of $\{0,1\}^\lambda$.

After this step 4 finishes, we say that $\mathcal{O}_n$ is *good* if for all $i \in [n]$ and $k \in [\tau]$ we have that

$$|\{c : \mathcal{O}_n(i, sk_{\mathsf{Index}(i,k)}, c) = v'_i\}| \geq 2^{(1-o(1))\lambda}.$$

We show that $\mathcal{O}_n$ must be good except with extremely small probability using a Chernoff bound. We defer the proof to Section 7.2.

Proposition 38. *The probability (over the randomness in step 4 conditioned on all the information revealed in prior steps) that $\mathcal{O}$ is not good is at most*

$$2^{-(1-o(1))\lambda n(\tau+1)}.$$

Step 5: Reveal the remaining values of $\mathcal{R}_n$ that determine f.

We now finish sampling f completely (up until now we have just sampled the secret keys). In particular, this means fully revealing the "random" coins deterministically generated by $\mathcal{R}_n$ in order to choose each $v_i \in \{0,1\}^\lambda$ and each $c_{i,k}$.

By the same query length argument as in step 2, we get that, even conditioned on the information revealed in the prior steps, these coins are i.i.d. uniform. This means that in step 5 each v_i is uniformly random from $\{0,1\}^\lambda$ for all $i \in [n]$ and each $c_{i,k}$ is chosen uniformly from the set $\{c \in \{0,1\}^{2\lambda} : \mathcal{O}_n(i, sk_{\mathsf{Index}}(i,k), c) = v_i\}$ for all $i \in [n]$ and $k \in [\tau]$.

We can bound the probability that $f = f'$ given that f' is a valid encoding.

Proposition 39. *Conditioned on the event that f' is a valid encoding, the probability (over the randomness in steps 4 and 5) that $f = f'$ is at most*

$$2^{-(1-o(1))\lambda n(\tau+1)}$$

Proof. Since each v_i is chosen independently uniformly at random independently from $\{0,1\}^\lambda$ and since each $c_{i,k}$ is chosen independently uniformly at random from the set $\{c : \mathcal{O}(i, sk_{\mathsf{Index}(i,k)}, c) = v_i\}$, we get that the probability $f = f'$ is at most

$$2^{-\lambda n} \prod_{i \in [n], k \in [\tau]} \frac{1}{|\{c : \mathcal{O}(i, sk_{\mathsf{Index}(i,k)}, c) = v'_i\}|}.$$

In the event that $\mathcal{O}$ is good, we get that the above quantity is at most

$$2^{-\lambda n} \prod_{i \in [n], k \in [\tau]} \frac{1}{2^{\lambda(1-o(1))}} = 2^{-\lambda n - (1-o(1))n\tau\lambda} = 2^{-(1-o(1))\lambda n(\tau+1)}.$$

On the other hand, the event that $\mathcal{O}$ is not good occurs with probability at most $2^{-(1-o(1))\lambda n(\tau+1)}$. Putting these two bounds together gives the desired bound. □

We are now ready to complete the proof of Lemma 30. We have that

$$\begin{aligned}
&\Pr_{\mathcal{O}}[f = f'] \\
\leq &\Pr_{\mathcal{O}}[f = f' | f' \text{ is a valid encoding}] \Pr_{\mathcal{O}}[f' \text{ is a valid encoding}] + \Pr_{\mathcal{O}}[f = f' | f' \text{ is not a valid encoding}] \\
= &\Pr_{\mathcal{O}}[f = f' | f' \text{ is a valid encoding}] \Pr_{\mathcal{O}}[f' \text{ is a valid encoding}] \\
\leq &\ 2^{-(1-o(1))\lambda n(\tau+1)} 2^{-\lambda n(\tau-1) - OPT\lambda + m\log(4mq) + \tau n \log(\tau q) + \tau \log(n) + \log m} \\
\leq &\ 2^{-(1-o(1))2\lambda n\tau - OPT\lambda + m\log(4mq) + \tau n \log(\tau q) + \tau \log(n) + \log m}.
\end{aligned}$$

On a NO instance, recall that we have that $OPT \geq n/3$. Also, using that $m \leq \tau n$ and $q \leq 2^{\lambda/(128\tau)}$ (recall, our assumption is that $q \leq 2^{\lambda/O(\tau)}$) and the assumption that $\lambda \geq \Omega_\tau(\log n)$ for some sufficiently large hidden constant (that can depend on τ), we have that

$$\begin{aligned}
& m\log(4mq) + \tau n \log(\tau q) + \tau\log(n) + \log m \\
\leq & \tau n \log(4\tau n q) + \tau n \log(\tau q) + \tau \log n + \log(\tau n) \\
\leq & 4\tau n \log(4\tau n q) \\
\leq & 4\tau n \log(4\tau n 2^{\lambda/(128\tau)}) \\
\leq & 4\tau n \log(2^{\lambda/(64\tau)}) \\
\leq & \lambda n/16
\end{aligned}$$

Substituting in both these bounds, we get that on a NO instance

$$\begin{aligned}
& \Pr_{\mathcal{O}}[f = f'] \\
\leq & 2^{-(1-o(1))2\lambda n\tau - OPT\lambda + m\log(4mq) + \tau n \log(\tau q) + \tau\log(n) + \log m} \\
\leq & 2^{-(1-o(1))2\lambda n\tau - \lambda n/3 + \lambda n/16} \\
\leq & 2^{-(1-o(1))2\lambda n\tau - \lambda n/4}
\end{aligned}$$

This completes the proof of Lemma 30.

7.2 Probability Bounds

Proposition 31. *The probability (over the randomness in step 1) that $C_w \geq t_w$ for all $2 \leq w \leq \tau$ is at most*

$$2^{\tau n \log(2\tau q) - \sum_{2\leq w\leq\tau} t_w \lambda}.$$

Proof. Recall P is some arbitrary decision tree and Q is the set of at most q queries that P has made to the oracle $\mathcal{O}_n$. Recall we say that $P^{\mathcal{O}}$ finds a *w-collision* on prefix $i \in [n]$ has a if there is a v such that

$$|\{(i, sk, c) \in Q : \mathcal{O}_n(i, sk, c) = v\}| \geq w.$$

We say $P^{\mathcal{O}}$ finds at least t_w distinct w-collisions if $P^{\mathcal{O}}$ finds a w-collision on at least t_w distinct prefixes.

In words, our goal is to bound the probability that

$$\Pr_{\mathcal{O}}[\ P^{\mathcal{O}} \text{ finds at least } t_w \text{ distinct } w\text{-collisions for all } 2 \leq w \leq \tau]$$

We will analyze a different decision tree P' that queries $\mathcal{O}_n$ on each prefix $i \in [n]$ on q distinct inputs (i, sk, c). Thus, in total, P' makes qn queries. It is easy to see that P' will be only better than P at finding collisions because $\mathcal{O}_n$ is a uniformly random function. Formally, we have that

$$\begin{aligned}
& \Pr_{\mathcal{O}}[C_w \geq t_w \text{ for all } 2 \leq w \leq \tau] \\
= & \Pr_{\mathcal{O}}[\ P^{\mathcal{O}} \text{ finds at least } t_w \text{ distinct } w\text{-collisions for all } 2 \leq w \leq \tau] \\
\leq & \Pr_{\mathcal{O}}[\ (P')^{\mathcal{O}} \text{ finds at least } t_w \text{ distinct } w\text{-collisions for all } 2 \leq w \leq \tau]
\end{aligned}$$

We now begin analyzing P'. To do this, it will be useful to introduce the following definition. Say a function $g : [n] \to [\tau]$ is *good* if for all $2 \leq w \leq \tau$ we have that

$$t_w \leq \sum_{i\in[n]} \mathbb{1}[g(i) \geq w].$$

We now return to analyzing P'. Observe that if $(P')^{\mathcal{O}}$ finds at least t_w distinct w-collisions for all $2 \leq w \leq \tau$, then there exists a good g such that $(P')^{\mathcal{O}}$ finds an $g(i)$-collision on prefix i. (To see this, for each value of i set $g(i)$ to be the max value of $w \in [\tau]$ such that $(P')^{\mathcal{O}}$ finds an w-collision on prefix i.) As a result, we have that

$$\begin{aligned}
&\Pr_{\mathcal{O}}[\ (P')^{\mathcal{O}} \text{ finds at least } t_w \text{ distinct } w\text{-collisions for all } 2 \leq w \leq \tau] \\
\leq &\Pr_{\mathcal{O}}[\text{exists good } g \text{ such that for all } i \in [n] \text{ we have that } (P')^{\mathcal{O}} \text{ finds an } g(i)\text{-collision on prefix } i] \\
\leq &\sum_{\text{good } g} \Pr_{\mathcal{O}}[\text{for all } i \text{ we have } (P')^{\mathcal{O}} \text{ finds an } g(i)\text{-collision on prefix } i]
\end{aligned}$$

Then by the construction of P' (in particular, using that its queries are non-adaptive and independent), we get that

$$\begin{aligned}
&\sum_{\text{good } g} \Pr_{\mathcal{O}}[\text{for all } i \text{ we have } (P')^{\mathcal{O}} \text{ finds an } g(i)\text{-collision on prefix } i] \\
\leq &\sum_{\text{good } g} \prod_{i \in [n]} \Pr_{\mathcal{O}}[(P')^{\mathcal{O}} \text{ finds an } g(i)\text{-collision on prefix } i] \\
\leq &\sum_{\text{good } g} \prod_{i \in [n]} q^{g(i)} 2^{-\lambda(g(i)-1)} \\
\leq &\sum_{\text{good } g} \prod_{i \in [n]} q^{\tau} 2^{-\lambda(g(i)-1)} \\
\leq &\sum_{\text{good } g} q^{\tau n} \prod_{i \in [n]} 2^{-\lambda(g(i)-1)} \\
\leq &\sum_{\text{good } g} q^{\tau n} 2^{-\lambda \sum_{i \in [n]}(g(i)-1)}
\end{aligned}$$

Now, recall the definition of a function $g : [n] \to [\tau]$ being *good* is if for all $2 \leq w \leq \tau$ we have that

$$t_w \leq \sum_{i \in [n]} \mathbb{1}[g(i) \geq w].$$

Summing over $2 \leq w \leq \tau$, the above condition implies that

$$\sum_{2 \leq w \leq \tau} t_w \leq \sum_{2 \leq w \leq \tau} \sum_{i \in [n]} \mathbb{1}[g(i) \geq w] = \sum_{i \in [n]} \sum_{2 \leq w \leq \tau} \mathbb{1}[g(i) \geq w] = \sum_{i \in [n]} (g(i) - 1).$$

Substituting this in we get that

$$\begin{aligned}
&\sum_{\text{good } g} q^{\tau n} 2^{-\lambda \sum_{i \in [n]}(g(i)-1)} \\
\leq &\sum_{\text{good } g} q^{\tau n} 2^{-\lambda \sum_{2 \leq w \leq \tau} t_w} \\
\leq &\tau^n q^{\tau n} 2^{-\lambda \sum_{2 \leq w \leq \tau} t_w} \\
\leq &2^{\tau n \log(2\tau q) - \sum_{2 \leq w \leq \tau} t_w \lambda}.
\end{aligned}$$

Thus, finally, we have that

$$\Pr_{\mathcal{O}}[\ (P')^{\mathcal{O}} \text{ finds at least } t_w \text{ distinct } w\text{-collisions for all } 2 \leq w \leq \tau] \leq 2^{\tau n \log(2\tau q) - \sum_{2 \leq w \leq \tau} t_w \lambda}$$

□

Proposition 32. *The probability (over the randomness in step 2 conditioned on all the information revealed in step 1) that $|skHit| = t$ is at most $2^{-t\lambda+m\log(4mq)}$.*

Proof. Recall, by the query length argument, the strings $sk_1, \dots, sk_m$ are each chosen uniformly at random from $\{0,1\}^\lambda$ *after* Q is determined in step 1. Thus

$$|skHit| = \sum_{d\in[m]} \mathbb{1}[(i, sk_d, c) \in Q \text{ for some } i, c]$$

is the sum of m independent Bernoulli random variables, each with expectation at most $q2^{-\lambda}$ (since Q contains at most q points).

Applying the Chernoff-Hoeffding bound, we get that the probability that $|skHit| \geq t$ is at most

$$2^{-mD(\frac{m-t}{m}||1-q2^{-\lambda})}$$

where $D(a||b) = a\log\frac{a}{b} + (1-a)\log\frac{1-a}{1-b}$ denotes Kullback–Leibler divergence.

We can calculate that

$$\begin{aligned}
&mD(\frac{m-t}{m}||1-q2^{-\lambda})\\
=&mD(1-t/m||1-q2^{-\lambda})\\
=&m[(1-t/m)\log(\frac{1-t/m}{1-q2^{-\lambda}}) + (t/m)\log\frac{t}{mq2^{-\lambda}}]\\
\geq&m[(1-t/m)\log(1-t/m) + (t/m)\log\frac{t}{mq2^{-\lambda}}]\\
\geq&m[-2t/m + (t/m)\log\frac{t}{mq2^{-\lambda}}]\\
=&-2t + t\log\frac{t}{mq2^{-\lambda}}\\
\geq&-2t + t(\lambda - \log(mq))\\
=&t\lambda - t\log(4mq)\\
\geq&t\lambda - m\log(4mq)
\end{aligned}$$

where we use the fact that $(1-x)\log(1-x) \geq -2x$ for all $x \geq 0$ and the fact that $t \leq m$ (since otherwise the proposition is vacuously true). □

Proposition 37. *The probability (over the randomness in steps 1, 2, and 3) that f' is a valid encoding is at most*

$$2^{-\lambda n(\tau-1)-OPT\cdot\lambda+m\log(4mq)+\tau n\log(\tau q)+\tau\log(n)+\log m}.$$

Proof. Proposition 36 says the probability that f' is a valid encoding conditioned on the value of B is at most $2^{-B\lambda}$.

Then applying the lower bound on B in Proposition 33 we know that conditioned on $skHit$ (which determines $Missed$) and C_w that the probability f' is a valid encoding is at most

$$2^{-\lambda n(\tau-1)-\lambda|Missed|+\sum_{w=2}^{\tau} C_w\lambda}.$$

Still conditioning on the values of C_w, but summing over all possible values of $|skHit|$ and multiplying by the probability bounds on $|skHit|$ in Proposition 32, we get the probability that f' is a valid encoding given the values of C_w is at most[27]

[27] In the expression below, $|Missed|$ is really a random variable that depends on $skHit$, but for simplicity we omit writing this dependence.

$$\sum_{|skHit|\in[m]} 2^{-\lambda n(\tau-1)-\lambda|Missed|+\sum_{w=2}^{\tau} C_w\lambda} 2^{-|skHit|\lambda+m\log(4mq)}.$$

We split this sum into two parts: the part of the sum when $|skHit| < OPT$ and the part when $|skHit| \geq OPT$. For the first part, we can bound

$$\begin{aligned}
&\sum_{|skHit|<OPT} 2^{-\lambda n(\tau-1)-\lambda|Missed|+\sum_{w=2}^{\tau} C_w\lambda} 2^{-|skHit|\lambda+m\log(4mq)} \\
\leq &\sum_{|skHit|<OPT} 2^{-\lambda n(\tau-1)-\lambda(OPT-|skHit|)+\sum_{w=2}^{\tau} C_w\lambda} 2^{-|skHit|\lambda+m\log(4mq)} \\
\leq &\sum_{|skHit|<OPT} 2^{-\lambda n(\tau-1)-\lambda OPT+\sum_{w=2}^{\tau} C_w\lambda+m\log(4mq)}
\end{aligned}$$

where the first inequality comes from observing that the union of $OPT-\alpha$ sets from a set cover instance must be missing at least α elements of the ground set (otherwise one could create a set cover smaller than OPT).

For the second part, we can bound

$$\begin{aligned}
&\sum_{|skHit|\geq OPT} 2^{-\lambda n(\tau-1)-\lambda|Missed|+\sum_{w=2}^{\tau} C_w\lambda} 2^{-|skHit|\lambda+m\log(4mq)} \\
\leq &\sum_{|skHit|\geq OPT} 2^{-\lambda n(\tau-1)-\lambda OPT+\sum_{w=2}^{\tau} C_w\lambda+m\log(4mq)}
\end{aligned}$$

Putting these bounds together, we get that the probability that f' is a valid encoding given the values of C_w is at most

$$m2^{-\lambda n(\tau-1)-OPT\lambda+\sum_{w=2}^{\tau} C_w\lambda+m\log(4mq)}$$

We now sum over the potential values of C_w and multiply by the tail bounds on C_w in Proposition 31 to obtain the following bound on the probability that f' is a valid encoding.

$$\begin{aligned}
&\sum_{C_2,\ldots,C_\tau\in[n]} m2^{-\lambda n(\tau-1)-OPT\lambda+\sum_{w=2}^{\tau} C_w\lambda+m\log(4mq)} \cdot 2^{\tau n\log(2\tau q)-\sum_{2\leq w\leq\tau} C_w\lambda} \\
= &\sum_{C_2,\ldots,C_\tau\in[n]} m2^{-\lambda n(\tau-1)-OPT\lambda+m\log(4mq)+\tau n\log(\tau q)} \\
\leq &(n)^\tau m2^{-\lambda n(\tau-1)-OPT\lambda+m\log(4mq)+\tau n\log(\tau q)} \\
\leq &2^{-\lambda n(\tau-1)-OPT\lambda+m\log(4mq)+\tau n\log(\tau q)+\tau\log(n)+\log m}.
\end{aligned}$$

□

Proposition 38. *The probability (over the randomness in step 4 conditioned on all the information revealed in prior steps) that $\mathcal{O}$ is not good is at most*

$$2^{-(1-o(1))\lambda n(\tau+1)}.$$

Proof. Fix any i and k. We will bound the probability of the event that

$$|\{c : \mathcal{O}(i,j,sk_{\mathsf{Index}(i,k)},c) = v_i'\}| < 2^{\lambda-\sqrt{\lambda}}$$

and then union bound over all i and k.

Recall that prior to step 4, $\mathcal{O}$ has only been revealed on only a small number of points. Namely, $\mathcal{O}$ is revealed on at most $q' \le \lambda q + n\tau$ points (λq points from running P in step 1 (the extra factor of λ is because we reveal all of $\mathcal{O}_n$ if it queries just one bit of $\mathcal{O}_n$) and $n\tau$ points from revealing the values of $\mathcal{O}$ on all $c'_{i,k}$ in step 3). Moreover, we have that

$$q' \le \lambda q + n\tau \le (1+o(1))2^{\lambda}/2 + (1+o(1))2^{\lambda}/2 = (1+o(1))2^{\lambda}$$

using the assumptions in the statement of Lemma 30 that $q \le 2^{\lambda/O(\tau)}$ and that $\lambda \ge \Omega_\tau(\log n)$, setting the hidden constants to be large enough. During step 4 the value of $\mathcal{O}$ on the remaining points is independently set to a uniformly random element of $\{0,1\}^{\lambda}$.

Thus the random variable $|\{c : \mathcal{O}(i, sk_{\mathsf{Index}(i,k)}, c) = v'_i\}|$ is the sum of at least $2^{2\lambda} - q' = (1-o(1))2^{2\lambda}$ independent Bernoulli random variables with mean $2^{-\lambda}$. Thus, by a Chernoff bound, we get that the probability that $|\{c : \mathcal{O}(i, sk_{\mathsf{Index}(i,k)}, c) = v'_i\}| \le (1-(1-2^{-\sqrt{\lambda}}))2^{\lambda}$ is at most

$$\exp(-(1-o(1))\frac{(1-2^{-\sqrt{\lambda}})^2}{2}2^{\lambda}) \le \exp(-(1-o(1))2^{\lambda-1}).$$

Union bounding over i and k, we get the probability $\mathcal{O}$ is not good is at most

$$n\tau\exp(-(1-o(1))2^{\lambda-1}) \le \exp(-(1-o(1))2^{\lambda-1} + \log(n\tau)) \le \exp(-(1-o(1))2^{\lambda-1}) \le 2^{-(1-o(1))\lambda n(\tau+1)}$$

again by setting the hidden constant in the assumption $\lambda \ge \Omega(\log n)$ to be sufficiently large enough. □

Acknowledgements

Part of this work was done while the author was participating in the Meta-Complexity program at the Simons Institute. The author is supported by a NSF Graduate Research Fellowship and NSF CCF-1909429

This work greatly benefited from many conversations including those with Marco Carmosino, Shuichi Hirahara, Yizhi Huang, Hanlin Ren, Rahul Santhanam, Neekon Vafa, and Ryan Williams.

References

[AB09] Sanjeev Arora and Boaz Barak. *Computational Complexity - A Modern Approach.* 2009. ISBN: 978-0-521-42426-4. URL: http://www.cambridge.org/catalogue/catalogue.asp?isbn=9780521424264 (cit. on p. 15).

[ABK+06] Eric Allender, Harry Buhrman, Michal Koucký, Dieter van Melkebeek, and Detlef Ronneburger. "Power from Random Strings". In: *SIAM J. Comput.* 35.6 (2006), pp. 1467–1493. URL: https://doi.org/10.1137/050628994 (cit. on pp. 5, 15, 16).

[ACM+21] Eric Allender, Mahdi Cheraghchi, Dimitrios Myrisiotis, Harsha Tirumala, and Ilya Volkovich. "One-Way Functions and a Conditional Variant of MKTP". In: *41st IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2021, December 15-17, 2021, Virtual Conference.* Vol. 213. LIPIcs. 2021, 7:1–7:19. URL: https://doi.org/10.4230/LIPIcs.FSTTCS.2021.7 (cit. on pp. 4, 5, 12, 15).

[AD17] Eric Allender and Bireswar Das. "Zero knowledge and circuit minimization". In: *Inf. Comput.* 256 (2017), pp. 2–8. URL: https://doi.org/10.1016/j.ic.2017.04.004 (cit. on p. 4).

[AH19] Eric Allender and Shuichi Hirahara. "New Insights on the (Non-)Hardness of Circuit Minimization and Related Problems". In: *ACM Trans. Comput. Theory* 11.4 (2019), 27:1–27:27. URL: https://doi.org/10.1145/3349616 (cit. on p. 5).

[AHK17] Eric Allender, Dhiraj Holden, and Valentine Kabanets. "The Minimum Oracle Circuit Size Problem". In: *Comput. Complex.* 26.2 (2017), pp. 469–496. URL: https://doi.org/10.1007/s00037-016-0124-0 (cit. on p. 5).

[AHM+08] Eric Allender, Lisa Hellerstein, Paul McCabe, Toniann Pitassi, and Michael E. Saks. “Minimizing Disjunctive Normal Form Formulas and AC^0 Circuits Given a Truth Table”. In: *SIAM J. Comput.* 38.1 (2008), pp. 63–84. URL: https://doi.org/10.1137/060664537 (cit. on pp. 5, 15).

[AIV21] Eric Allender, Rahul Ilango, and Neekon Vafa. “The Non-hardness of Approximating Circuit Size”. In: *Theory Comput. Syst.* 65.3 (2021), pp. 559–578. URL: https://doi.org/10.1007/s00224-020-10004-x (cit. on pp. 11, 12).

[Bab16] László Babai. “Graph isomorphism in quasipolynomial time [extended abstract]”. In: *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, Cambridge, MA, USA, June 18-21, 2016.* 2016, pp. 684–697. URL: https://doi.org/10.1145/2897518.2897542 (cit. on p. 3).

[BCB13] Samuel Buss, Marco Carmosino, and Radheshyam Balasundaram. *Circuit Complexity, Session 2 Lecture Notes.* 2013. URL: https://mathweb.ucsd.edu/~sbuss/CourseWeb/Math262A_2013F/Scribe02.pdf (cit. on p. 50).

[BCK+22] Eric Binnendyk, Marco Carmosino, Antonina Kolokolova, R. Ramyaa, and Manuel Sabin. “Learning with Distributional Inverters”. In: *International Conference on Algorithmic Learning Theory, 29 March - 1 April 2022, Paris, France.* Vol. 167. Proceedings of Machine Learning Research. 2022, pp. 90–106. URL: https://proceedings.mlr.press/v167/binnendyk22a.html (cit. on p. 4).

[BG81] Charles H. Bennett and John Gill. “Relative to a Random Oracle A, P^{A} != NP^{A} != co-NP^{A} with Probability 1”. In: *SIAM J. Comput.* 10.1 (1981), pp. 96–113. URL: https://doi.org/10.1137/0210008 (cit. on pp. 6, 8).

[BKP18] Nir Bitansky, Yael Tauman Kalai, and Omer Paneth. “Multi-collision resistance: a paradigm for keyless hash functions”. In: *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, Los Angeles, CA, USA, June 25-29, 2018.* 2018, pp. 671–684. URL: https://doi.org/10.1145/3188745.3188870 (cit. on pp. 8, 15).

[BR93] Mihir Bellare and Phillip Rogaway. “Random Oracles are Practical: A Paradigm for Designing Efficient Protocols”. In: *CCS ’93, Proceedings of the 1st ACM Conference on Computer and Communications Security, Fairfax, Virginia, USA, November 3-5, 1993.* 1993, pp. 62–73. URL: https://doi.org/10.1145/168588.168596 (cit. on pp. 6, 8).

[BRS02] John Black, Phillip Rogaway, and Thomas Shrimpton. “Black-Box Analysis of the Block-Cipher-Based Hash-Function Constructions from PGV”. In: *Advances in Cryptology - CRYPTO 2002, 22nd Annual International Cryptology Conference, Santa Barbara, California, USA, August 18-22, 2002, Proceedings.* Vol. 2442. Lecture Notes in Computer Science. 2002, pp. 320–335. URL: https://doi.org/10.1007/3-540-45708-9%5C_21 (cit. on p. 8).

[BT06] Andrej Bogdanov and Luca Trevisan. “On Worst-Case to Average-Case Reductions for NP Problems”. In: *SIAM J. Comput.* 36.4 (2006), pp. 1119–1159. URL: https://doi.org/10.1137/S0097539705446974 (cit. on p. 4).

[CCG+94] Richard Chang, Benny Chor, Oded Goldreich, Juris Hartmanis, Johan Håstad, Desh Ranjan, and Pankaj Rohatgi. “The Random Oracle Hypothesis Is False”. In: *J. Comput. Syst. Sci.* 49.1 (1994), pp. 24–39. URL: https://doi.org/10.1016/S0022-0000(05)80084-4 (cit. on p. 8).

[CGH04] Ran Canetti, Oded Goldreich, and Shai Halevi. “The random oracle methodology, revisited”. In: *J. ACM* 51.4 (2004), pp. 557–594. URL: https://doi.org/10.1145/1008731.1008734 (cit. on p. 8).

[CHV22] Lijie Chen, Shuichi Hirahara, and Neekon Vafa. “Average-Case Hardness of NP and PH from Worst-Case Fine-Grained Assumptions”. In: *13th Innovations in Theoretical Computer Science Conference, ITCS 2022, January 31 - February 3, 2022, Berkeley, CA, USA.* Vol. 215. LIPIcs. 2022, 45:1–45:16. URL: https://doi.org/10.4230/LIPIcs.ITCS.2022.45 (cit. on pp. 4, 18).

[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*. Vol. 50. LIPIcs. 2016, 10:1–10:24. URL: https://doi.org/10.4230/LIPIcs.CCC.2016.10 (cit. on p. 4).

[CIKK17] Marco L. Carmosino, Russell Impagliazzo, Valentine Kabanets, and Antonina Kolokolova. "Agnostic Learning from Tolerant Natural Proofs". In: *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2017, August 16-18, 2017, Berkeley, CA, USA*. Vol. 81. LIPIcs. 2017, 35:1–35:19. URL: https://doi.org/10.4230/LIPIcs.APPROX-RANDOM.2017.35 (cit. on p. 4).

[CKLM19] Mahdi Cheraghchi, Valentine Kabanets, Zhenjian Lu, and Dimitrios Myrisiotis. "Circuit Lower Bounds for MCSP from Local Pseudorandom Generators". In: *46th International Colloquium on Automata, Languages, and Programming, ICALP 2019, July 9-12, 2019, Patras, Greece*. Vol. 132. LIPIcs. 2019, 39:1–39:14. URL: https://doi.org/10.4230/LIPIcs.ICALP.2019.39 (cit. on p. 4).

[Czo99] Sebastian Lukas Arne Czort. "The Complexity of Minimizing Disjunctive Normal Form Formulas". MA thesis. University of Aarhus, 1999 (cit. on pp. 5, 15).

[DGKR05] Irit Dinur, Venkatesan Guruswami, Subhash Khot, and Oded Regev. "A New Multilayered PCP and the Hardness of Hypergraph Vertex Cover". In: *SIAM J. Comput.* 34.5 (2005), pp. 1129–1146. URL: https://doi.org/10.1137/S0097539704443057 (cit. on pp. 6, 26).

[DR20] Joan Daemen and Vincent Rijmen. *The Design of Rijndael - The Advanced Encryption Standard (AES), Second Edition*. Information Security and Cryptography. 2020. ISBN: 978-3-662-60768-8. URL: https://doi.org/10.1007/978-3-662-60769-5 (cit. on p. 8).

[Fel09] Vitaly Feldman. "Hardness of approximate two-level logic minimization and PAC learning with membership queries". In: *J. Comput. Syst. Sci.* 75.1 (2009), pp. 13–26. URL: https://doi.org/10.1016/j.jcss.2008.07.007 (cit. on pp. 5, 15).

[FF93] Joan Feigenbaum and Lance Fortnow. "Random-Self-Reducibility of Complete Sets". In: *SIAM J. Comput.* 22.5 (1993), pp. 994–1005. URL: https://doi.org/10.1137/0222061 (cit. on p. 4).

[FM05] Gudmund Skovbjerg Frandsen and Peter Bro Miltersen. "Reviewing bounds on the circuit size of the hardest functions". In: *Inf. Process. Lett.* 95.2 (2005), pp. 354–357. URL: https://doi.org/10.1016/j.ipl.2005.03.009 (cit. on p. 34).

[GGSW13] Sanjam Garg, Craig Gentry, Amit Sahai, and Brent Waters. "Witness encryption and its applications". In: *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013*. 2013, pp. 467–476. URL: https://doi.org/10.1145/2488608.2488667 (cit. on p. 10).

[GII+19] Alexander Golovnev, Rahul Ilango, Russell Impagliazzo, Valentine Kabanets, Antonina Kolokolova, and Avishay Tal. "$\mathrm{AC}^0[p]$ Lower Bounds Against MCSP via the Coin Problem". In: *46th International Colloquium on Automata, Languages, and Programming, ICALP 2019, July 9-12, 2019, Patras, Greece*. Vol. 132. LIPIcs. 2019, 66:1–66:15. URL: https://doi.org/10.4230/LIPIcs.ICALP.2019.66 (cit. on p. 4).

[GK22] Halley Goldberg and Valentine Kabanets. "A Simpler Proof of the Worst-Case to Average-Case Reduction for Polynomial Hierarchy via Symmetry of Information". In: *Electron. Colloquium Comput. Complex.* TR22-007 (2022). ECCC: TR22-007. URL: https://eccc.weizmann.ac.il/report/2022/007 (cit. on pp. 4, 18).

[GKLO22] Halley Goldberg, Valentine Kabanets, Zhenjian Lu, and Igor Carboni Oliveira. "Probabilistic Kolmogorov Complexity with Applications to Average-Case Complexity". In: *37th Computational Complexity Conference, CCC 2022, July 20-23, 2022, Philadelphia, PA, USA*. Vol. 234. LIPIcs. 2022, 16:1–16:60. URL: https://doi.org/10.4230/LIPIcs.CCC.2022.16 (cit. on pp. 4, 14, 17).

[GR22] Ludmila Glinskih and Artur Riazanov. "MCSP is Hard for Read-Once Nondeterministic Branching Programs". In: *LATIN 2022: Theoretical Informatics - 15th Latin American Symposium, Guanajuato, Mexico, November 7-11, 2022, Proceedings*. Vol. 13568. Lecture Notes in Computer Science. 2022, pp. 626–640. URL: https://doi.org/10.1007/978-3-031-20624-5%5C_38 (cit. on p. 4).

[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*. 1983, pp. 439–445. URL: https://doi.org/10.1109/SFCS.1983.21 (cit. on pp. 4, 8, 16).

[HIL+23] Shuichi Hirahara, Rahul Ilango, Zhenjian Lu, Mikito Nanashima, and Igor C. Oliveira. "A Duality between One-Way Functions and Average-Case Symmetry of Information". In: *Proceedings of the 55th Annual ACM Symposium on Theory of Computing, STOC 2023, Orlando, FL, USA, June 20-23, 2023*. 2023, pp. 1039–1050. URL: https://doi.org/10.1145/3564246.3585138 (cit. on p. 4).

[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*. 2018, pp. 247–258. URL: https://doi.org/10.1109/FOCS.2018.00032 (cit. on pp. 4, 10).

[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*. 2020, pp. 50–60. URL: https://doi.org/10.1109/FOCS46700.2020.00014 (cit. on p. 4).

[Hir21] Shuichi Hirahara. "Average-case hardness of NP from exponential worst-case hardness assumptions". In: *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing, Virtual Event, Italy, June 21-25, 2021*. 2021, pp. 292–302. URL: https://doi.org/10.1145/3406325.3451065 (cit. on pp. 4, 18).

[Hir22a] Shuichi Hirahara. "NP-Hardness of Learning Programs and Partial MCSP". In: *FOCS*. 2022, pp. 968–979 (cit. on pp. 5, 6, 14, 15).

[Hir22b] Shuichi Hirahara. "Symmetry of Information from Meta-Complexity". In: *37th Computational Complexity Conference, CCC 2022, July 20-23, 2022, Philadelphia, PA, USA*. Vol. 234. LIPIcs. 2022, 26:1–26:41. URL: https://doi.org/10.4230/LIPIcs.CCC.2022.26 (cit. on pp. 4, 5, 10, 12, 15, 18).

[HIR23] Yizhi Huang, Rahul Ilango, and Hanlin Ren. "NP-Hardness of Approximating Meta-Complexity: A Cryptographic Approach". In: *Proceedings of the 55th Annual ACM Symposium on Theory of Computing, STOC 2023, Orlando, FL, USA, June 20-23, 2023*. 2023, pp. 1067–1075. URL: https://doi.org/10.1145/3564246.3585154 (cit. on pp. 5, 6, 10, 12, 14, 15).

[Hir23] Shuichi Hirahara. "Capturing One-Way Functions via NP-Hardness of Meta-Complexity". In: *Proceedings of the 55th Annual ACM Symposium on Theory of Computing, STOC 2023, Orlando, FL, USA, June 20-23, 2023*. 2023, pp. 1027–1038. URL: https://doi.org/10.1145/3564246.3585130 (cit. on p. 4).

[HN21] Shuichi Hirahara and Mikito Nanashima. "On Worst-Case Learning in Relativized Heuristica". In: *62nd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2021, Denver, CO, USA, February 7-10, 2022*. 2021, pp. 751–758. URL: https://doi.org/10.1109/FOCS52979.2021.00078 (cit. on pp. 4, 18).

[HN22] Shuichi Hirahara and Mikito Nanashima. "Finding Errorless Pessiland in Error-Prone Heuristica". In: *37th Computational Complexity Conference, CCC 2022, July 20-23, 2022, Philadelphia, PA, USA*. Vol. 234. LIPIcs. 2022, 25:1–25:28. URL: https://doi.org/10.4230/LIPIcs.CCC.2022.25 (cit. on p. 4).

[HOS18] Shuichi Hirahara, Igor Carboni Oliveira, and Rahul Santhanam. "NP-hardness of Minimum Circuit Size Problem for OR-AND-MOD Circuits". In: *33rd Computational Complexity Conference, CCC 2018, June 22-24, 2018, San Diego, CA, USA*. Vol. 102. LIPIcs. 2018, 5:1–5:31. URL: https://doi.org/10.4230/LIPIcs.CCC.2018.5 (cit. on pp. 5, 15).

[HP15] John M. Hitchcock and Aduri Pavan. "On the NP-Completeness of the Minimum Circuit Size Problem". In: *35th IARCS Annual Conference on Foundation of Software Technology and Theoretical Computer Science, FSTTCS 2015, December 16-18, 2015, Bangalore, India*. Vol. 45. LIPIcs. 2015, pp. 236–245. URL: https://doi.org/10.4230/LIPIcs.FSTTCS.2015.236 (cit. on p. 5).

[HS22] Shuichi Hirahara and Rahul Santhanam. "Excluding PH Pessiland". In: *13th Innovations in Theoretical Computer Science Conference, ITCS 2022, January 31 - February 3, 2022, Berkeley, CA, USA*. Vol. 215. LIPIcs. 2022, 85:1–85:25. URL: https://doi.org/10.4230/LIPIcs.ITCS.2022.85 (cit. on p. 4).

[HW16] Shuichi Hirahara and Osamu Watanabe. "Limits of Minimum Circuit Size Problem as Oracle". In: *31st Conference on Computational Complexity, CCC 2016, May 29 to June 1, 2016, Tokyo, Japan*. Vol. 50. LIPIcs. 2016, 18:1–18:20. URL: https://doi.org/10.4230/LIPIcs.CCC.2016.18 (cit. on p. 5).

[IKV18] Russell Impagliazzo, Valentine Kabanets, and Ilya Volkovich. "The Power of Natural Properties as Oracles". In: *33rd Computational Complexity Conference, CCC 2018, June 22-24, 2018, San Diego, CA, USA*. Vol. 102. LIPIcs. 2018, 7:1–7:20. URL: https://doi.org/10.4230/LIPIcs.CCC.2018.7 (cit. on p. 15).

[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*. 1990, pp. 812–821. URL: https://doi.org/10.1109/FSCS.1990.89604 (cit. on p. 4).

[Ila20a] Rahul Ilango. "Approaching MCSP from Above and Below: Hardness for a Conditional Variant and AC0[p]". In: *11th Innovations in Theoretical Computer Science Conference, ITCS 2020, January 12-14, 2020, Seattle, Washington, USA*. Vol. 151. LIPIcs. 2020, 34:1–34:26. URL: https://doi.org/10.4230/LIPIcs.ITCS.2020.34 (cit. on pp. 5, 12, 15).

[Ila20b] Rahul Ilango. "Constant Depth Formula and Partial Function Versions of MCSP are Hard". In: *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020, Durham, NC, USA, November 16-19, 2020*. 2020, pp. 424–433. URL: https://doi.org/10.1109/FOCS46700.2020.00047 (cit. on pp. 5, 15).

[Ila21] Rahul Ilango. "The Minimum Formula Size Problem is (ETH) Hard". In: *62nd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2021, Denver, CO, USA, February 7-10, 2022*. 2021, pp. 427–432. URL: https://doi.org/10.1109/FOCS52979.2021.00050 (cit. on pp. 5, 15).

[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)*. Vol. 169. LIPIcs. 2020, 22:1–22:36. URL: https://doi.org/10.4230/LIPIcs.CCC.2020.22 (cit. on pp. 5, 15).

[Imp95] Russell Impagliazzo. "A Personal View of Average-Case Complexity". In: *Proceedings of the Tenth Annual Structure in Complexity Theory Conference, Minneapolis, Minnesota, USA, June 19-22, 1995*. 1995, pp. 134–147. URL: https://doi.org/10.1109/SCT.1995.514853 (cit. on p. 4).

[IRS22] Rahul Ilango, Hanlin Ren, and Rahul Santhanam. "Robustness of average-case meta-complexity via pseudorandomness". In: *STOC*. 2022, pp. 1575–1583 (cit. on p. 4).

[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*. 2000, pp. 73–79. URL: https://doi.org/10.1145/335305.335314 (cit. on pp. 3–5).

[KKL+20] Valentine Kabanets, Sajin Koroth, Zhenjian Lu, Dimitrios Myrisiotis, and Igor Carboni Oliveira. "Algorithms and Lower Bounds for De Morgan Formulas of Low-Communication Leaf Gates". In: *35th Computational Complexity Conference, CCC 2020, July 28-31, 2020, Saarbrücken, Germany (Virtual Conference)*. Vol. 169. LIPIcs. 2020, 15:1–15:41. URL: https://doi.org/10.4230/LIPIcs.CCC.2020.15 (cit. on p. 4).

[KMR95] Stuart A. Kurtz, Stephen R. Mahaney, and James S. Royer. "The Isomorphism Conjecture Fails Relative to a Random Oracle". In: *J. ACM* 42.2 (1995), pp. 401–420. URL: https://doi.org/10.1145/201019.201030 (cit. on p. 8).

[Ko86] Ker-I Ko. "On the Notion of Infinite Pseudorandom Sequences". In: *Theor. Comput. Sci.* 48.3 (1986), pp. 9–33. URL: https://doi.org/10.1016/0304-3975(86)90081-2 (cit. on pp. 4, 16).

[Ko91] Ker-I Ko. "On the Complexity of Learning Minimum Time-Bounded Turing Machines". In: *SIAM J. Comput.* 20.5 (1991), pp. 962–986. URL: https://doi.org/10.1137/0220059 (cit. on pp. 4, 16).

[KS08] Subhash Khot and Rishi Saket. "Hardness of Minimizing and Learning DNF Expressions". In: *49th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2008, October 25-28, 2008, Philadelphia, PA, USA*. 2008, pp. 231–240. URL: https://doi.org/10.1109/FOCS.2008.37 (cit. on p. 5).

[Kur83] Stuart A. Kurtz. "On the Random Oracle Hypothesis". In: *Inf. Control.* 57.1 (1983), pp. 40–47. URL: https://doi.org/10.1016/S0019-9958(83)80023-0 (cit. on p. 8).

[Lev] Leonid Levin. *Hardness of Search Problems*. URL: https://www.cs.bu.edu/fac/lnd/research/hard.htm (visited on 11/04/2022) (cit. on p. 3).

[Lev73] Leonid Anatolevich Levin. "Universal search problems". In: *Problemy peredachi informatsii* 9.3 (1973), pp. 115–116 (cit. on p. 3).

[LM93] Luc Longpré and Sarah Mocas. "Symmetry of Information and One-Way Functions". In: *Inf. Process. Lett.* 46.2 (1993), pp. 95–100. URL: https://doi.org/10.1016/0020-0190(93)90204-M (cit. on p. 12).

[LO22] Zhenjian Lu and Igor Carboni Oliveira. "Theory and Applications of Probabilistic Kolmogorov Complexity". In: *Bull. EATCS* 137 (2022). URL: http://bulletin.eatcs.org/index.php/beatcs/article/view/700 (cit. on p. 17).

[LOZ22] Zhenjian Lu, Igor Carboni Oliveira, and Marius Zimand. "Optimal Coding Theorems in Time-Bounded Kolmogorov Complexity". In: *49th International Colloquium on Automata, Languages, and Programming, ICALP 2022, July 4-8, 2022, Paris, France*. Vol. 229. LIPIcs. 2022, 92:1–92:14. URL: https://doi.org/10.4230/LIPIcs.ICALP.2022.92 (cit. on p. 17).

[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*. 2020, pp. 1243–1254. URL: https://doi.org/10.1109/FOCS46700.2020.00118 (cit. on p. 4).

[LP21a] Yanyi Liu and Rafael Pass. "Cryptography from sublinear-time average-case hardness of time-bounded Kolmogorov complexity". In: *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing, Virtual Event, Italy, June 21-25, 2021*. 2021, pp. 722–735. URL: https://doi.org/10.1145/3406325.3451121 (cit. on p. 4).

[LP21b] Yanyi Liu and Rafael Pass. “On the Possibility of Basing Cryptography on EXP≠ BPP”. In: *Advances in Cryptology - CRYPTO 2021 - 41st Annual International Cryptology Conference, CRYPTO 2021, Virtual Event, August 16-20, 2021, Proceedings, Part I*. Vol. 12825. Lecture Notes in Computer Science. 2021, pp. 11–40. URL: https://doi.org/10.1007/978-3-030-84242-0%5C_2 (cit. on p. 4).

[LP22] Yanyi Liu and Rafael Pass. “On One-Way Functions from NP-Complete Problems”. In: *37th Computational Complexity Conference, CCC 2022, July 20-23, 2022, Philadelphia, PA, USA*. Vol. 234. LIPIcs. 2022, 36:1–36:24. URL: https://doi.org/10.4230/LIPIcs.CCC.2022.36 (cit. on pp. 4, 5, 12, 14, 15).

[LP23] Yanyi Liu and Rafael Pass. “One-Way Functions and the Hardness of (Probabilistic) Time-Bounded Kolmogorov Complexity w.r.t. Samplable Distributions”. In: *Advances in Cryptology - CRYPTO 2023 - 43rd Annual International Cryptology Conference, CRYPTO 2023, Santa Barbara, CA, USA, August 20-24, 2023, Proceedings, Part II*. Vol. 14082. Lecture Notes in Computer Science. 2023, pp. 645–673. URL: https://doi.org/10.1007/978-3-031-38545-2%5C_21 (cit. on p. 4).

[Lup58] O. B. Lupanov. “A Method of Circuit Synthesis”. In: *Izv. VUZ Radiofiz.* 1 (1958), pp. 120–140 (cit. on pp. 33, 50).

[LV19] Ming Li and Paul M. B. Vitányi. *An Introduction to Kolmogorov Complexity and Its Applications, 4th Edition*. Texts in Computer Science. 2019. ISBN: 978-3-030-11297-4. URL: https://doi.org/10.1007/978-3-030-11298-1 (cit. on pp. 15, 16).

[LW95] Luc Longpré and Osamu Watanabe. “On Symmetry of Information and Polynomial Time Invertibility”. In: *Inf. Comput.* 121.1 (1995), pp. 14–22. URL: https://doi.org/10.1006/inco.1995.1120 (cit. on p. 12).

[Mas79] William J. Masek. “Some NP-complete Set Covering Problems”. Unpublished Manuscript. 1979 (cit. on pp. 5, 15).

[Mic00] Silvio Micali. “Computationally Sound Proofs”. In: *SIAM J. Comput.* 30.4 (2000), pp. 1253–1298. URL: https://doi.org/10.1137/S0097539795284959 (cit. on p. 6).

[MW17] Cody D. Murray and R. Ryan Williams. “On the (Non) NP-Hardness of Computing Circuit Complexity”. In: *Theory Comput.* 13.1 (2017), pp. 1–22. URL: https://doi.org/10.4086/toc.2017.v013a004 (cit. on p. 5).

[Raz87] A. A. Razborov. “Lower bounds on the size of bounded depth circuits over a complete basis with logical addition”. English. In: *Math. Notes* 41 (1987), pp. 333–338. ISSN: 0001-4346 (cit. on p. 4).

[RR97] Alexander A. Razborov and Steven Rudich. “Natural Proofs”. In: *J. Comput. Syst. Sci.* 55.1 (1997), pp. 24–35. URL: https://doi.org/10.1006/jcss.1997.1494 (cit. on p. 4).

[RS21] Hanlin Ren and Rahul Santhanam. “Hardness of KT Characterizes Parallel Cryptography”. In: *36th Computational Complexity Conference, CCC 2021, July 20-23, 2021, Toronto, Ontario, Canada (Virtual Conference)*. Vol. 200. LIPIcs. 2021, 35:1–35:58. URL: https://doi.org/10.4230/LIPIcs.CCC.2021.35 (cit. on p. 4).

[RS22] Hanlin Ren and Rahul Santhanam. “A Relativization Perspective on Meta-Complexity”. In: *39th International Symposium on Theoretical Aspects of Computer Science, STACS 2022, March 15-18, 2022, Marseille, France (Virtual Conference)*. Vol. 219. LIPIcs. 2022, 54:1–54:13. URL: https://doi.org/10.4230/LIPIcs.STACS.2022.54 (cit. on p. 5).

[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*. Vol. 151. LIPIcs. 2020, 68:1–68:26. URL: https://doi.org/10.4230/LIPIcs.ITCS.2020.68 (cit. on p. 4).

[Sha92] Adi Shamir. “IP = PSPACE”. In: *J. ACM* 39.4 (1992), pp. 869–877. URL: https://doi.org/10.1145/146585.146609 (cit. on p. 8).

[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*. 1983, pp. 330–335. URL: https://doi.org/10.1145/800061.808762 (cit. on pp. 4, 16).

[Smo87] Roman Smolensky. “Algebraic Methods in the Theory of Lower Bounds for Boolean Circuit Complexity”. In: *Proceedings of the 19th Annual ACM Symposium on Theory of Computing, 1987, New York, New York, USA*. 1987, pp. 77–82. URL: https://doi.org/10.1145/28395.28404 (cit. on p. 4).

[SS20] Michael E. Saks and Rahul Santhanam. “Circuit Lower Bounds from NP-Hardness of MCSP Under Turing Reductions”. In: *35th Computational Complexity Conference, CCC 2020, July 28-31, 2020, Saarbrücken, Germany (Virtual Conference)*. Vol. 169. LIPIcs. 2020, 26:1–26:13. URL: https://doi.org/10.4230/LIPIcs.CCC.2020.26 (cit. on p. 5).

[SS22] Michael E. Saks and Rahul Santhanam. “On Randomized Reductions to the Random Strings”. In: *37th Computational Complexity Conference, CCC 2022, July 20-23, 2022, Philadelphia, PA, USA*. Vol. 234. LIPIcs. 2022, 29:1–29:30. URL: https://doi.org/10.4230/LIPIcs.CCC.2022.29 (cit. on pp. 9, 11).

[Tra84] Boris A. Trakhtenbrot. “A Survey of Russian Approaches to Perebor (Brute-Force Searches) Algorithms”. In: *IEEE Annals of the History of Computing* 6.4 (1984), pp. 384–400 (cit. on p. 3).

[UVS06] Christopher Umans, Tiziano Villa, and Alberto L. Sangiovanni-Vincentelli. “Complexity of two-level logic minimization”. In: *IEEE Trans. on CAD of Integrated Circuits and Systems* 25.7 (2006), pp. 1230–1246. URL: https://doi.org/10.1109/TCAD.2005.855944 (cit. on pp. 5, 15).

[Zha19] Mark Zhandry. “The Magic of ELFs”. In: *J. Cryptol.* 32.3 (2019), pp. 825–866. URL: https://doi.org/10.1007/s00145-018-9289-9 (cit. on p. 15).

[ZL70] A. K. Zvonkin and L. A. Levin. “The Complexity of Finite Objects and the Development of the Concepts of Information and Randomness by Means of the Theory of Algorithms”. In: *Russian Mathematical Surveys* 25 (1970), pp. 83–124 (cit. on p. 12).

A Lupanov for Multiple Outputs

Here, we show how Lupanov's near-optimal construction of circuits for random Boolean functions with one output bit easily generalizes to multiple output bits. To begin, we recall Lupanov's original construction. Our presentation is based on [BCB13].

Theorem 40 (Lupanov [Lup58])**.** *Let $f : \{0,1\}^n \to \{0,1\}$. Then there is a constant-depth circuit computing f of size $(1+o(1))\frac{2^n}{n}$.*

Proof. Let $k \in [n]$ and $s \in \mathbb{N}$ be parameters we set later. We break up the inputs to f and view $f : \{0,1\}^k \times \{0,1\}^{n-k} \to \{0,1\}$. We also fix a partition of $\{0,1\}^k$ into sets $A_1, \ldots, A_p$ of size at most s where $p \le (2^k/s + 1)$.

Then, we can write

$$f(x,y) = \bigvee_{i\in[p]} \mathbb{1}[x \in A_i] \wedge f(x,y)$$

For any $i \in [p]$ and any string $y \in \{0,1\}^{n-k}$, we write $f(A_i, y)$ as shorthand for the function $g : A_i \to \{0,1\}$ given by $g(x) = f(x,y)$. Then, we can write

$$f(x,y) = \bigvee_{i\in[p]} \bigvee_{g:A_i\to\{0,1\}} \mathbb{1}[x \in A_i \text{ and } g(x) = 1] \wedge \mathbb{1}[f(A_i,y) = g]$$

We now bound the circuit complexity of computing f in this way (which is clearly implementable in constant-depth). First, we compute once and for all the minterms[28] on x and minterms on y. This costs $(1+o(1))(2^k+2^{n-k})$ upfront, but we can reuse these gates in later computations (which will give us savings).

For each $i \in [p]$ and $g : A_i \to \{0,1\}$, the function $\mathbb{1}[x \in A_i \text{ and } g(x) = 1]$ takes k-bit inputs and accepts at most $|A_i| \leq s$ strings. We will compute this function using a DNF. Since we have already computed the minterms on x, the cost of computing each $\mathbb{1}[x \in A_i \text{ and } g(x) = 1]$ is at most s (we just need to pay for the OR gates in the DNF). Summing over all p possibilities for i and all 2^s possibilities for g gives a total cost of at most $p2^s s$

Next, we compute $\mathbb{1}[f(A_i, y) = g]$ also by a DNF. The number of inputs this function accepts is exactly $\sum_{y\in\{0,1\}^{n-k}} \mathbb{1}[f(A_i, y) = g]$. Since this is only a function on the y variable and we have already paid for computing all the y minterms, the cost of computing this function is at most $\sum_{y\in\{0,1\}^{n-k}} \mathbb{1}[f(A_i, y) = g]$. Thus, summing over all $i \in [p]$ and all g, the total cost is at most

$$\begin{aligned}\sum_{i\in[p]} \sum_{g:A_i\to\{0,1\}} \sum_{y\in\{0,1\}^{n-k}} \mathbb{1}[f(A_i, y) = g] &= \sum_{i\in[p]} \sum_{y\in\{0,1\}^{n-k}} \sum_{g:A_i\to\{0,1\}} \mathbb{1}[f(A_i, y) = g] \\ &= \sum_{i\in[p]} \sum_{y\in\{0,1\}^{n-k}} 1 \\ &= p2^{n-k}.\end{aligned}$$

Finally, now that we have subcircuits computing $\mathbb{1}[x \in A_i \text{ and } g(x) = 1]$ and $\mathbb{1}[f(A_i, y) = g]$, we need to compute

$$\bigvee_{i\in[p]} \bigvee_{g:A_i\to\{0,1\}} \mathbb{1}[x \in A_i \text{ and } g(x) = 1] \wedge \mathbb{1}[f(A_i, y) = g].$$

This requires at most p gates for the outer OR, $p2^s$ gates for the inner OR and $p2^s$ gates for the inner AND. Thus, we need at most $3p2^s$ extra gates.

In total, this gives us a bound of

$$(1 + o(1))(2^k + 2^{n-k}) + p2^s s + p2^{n-k} + 3p2^s$$

Setting $k = 3 \log n$ and $s = n - 5 \log n$ yields the theorem (the dominant term is $p2^{n-k}$). □

Theorem 29 (Multi-Output Lupanov)**.** *Let $f : \{0,1\}^n \to \{0,1\}^m$ with $m \geq 2$. Then there is a constant-depth circuit computing f of size $(1 + o(1))\frac{m2^n}{n \log m}$.*

Proof. For each $j \in [m]$, let $f_j : \{0,1\}^n \to \{0,1\}$ be the j'th output of the f function.

Let k and s be parameters we set later. We repeat the construction in Theorem 40 using this k and s to compute each f_j for all $j \in [m]$. Doing this naively would give a cost of

$$m[(1 + o(1))(2^k + 2^{n-k}) + p2^s s + p2^{n-k} + 3p2^s].$$

However, note that we only need to compute the minterms on x and y once (they will be the same no matter which f_j we are computing). Similarly, we only need to compute $\mathbb{1}[x \in A_i \text{ and } g(x) = 1]$ once for each $i \in [p]$ and $g : A_i \to \{0,1\}$ (these functions do not depend on f_j).

Thus, we instead get the bound

$$(1 + o(1))(2^k + 2^{n-k}) + p2^s s + mp2^{n-k} + 3mp2^s.$$

Now we set $s = n \log(m) - 5 \log(n \log m)$ and $k = 3 \log(n \log m)$. In this case, $p \leq 2^k/s + 1 \leq (1 + o(1))2^k/s$. We have

$$\begin{aligned}&(1 + o(1))(2^k + 2^{n-k}) + p2^s s + mp2^{n-k} + 3mp2^s \\ \leq&(1 + o(1))(2^k + 2^{n-k}) + (1 + o(1))2^{s+k} + m2^n/s + 3m2^{k+s}/s \\ \leq&(1 + o(1))m2^n/(n \log m)\end{aligned}$$

[28] A minterm of x is just the AND of some subset of variables in x.

□