THEORY OF COMPUTING, Volume 22 (2064), 2026, pp. 1–74
www.theoryofcomputing.org

SPECIAL ISSUE: CCC 2021

Hardness of KT Characterizes Parallel Cryptography

Hanlin Ren Rahul Santhanam

Received date; Revised February 26, 2026; Published April 3, 2026

Abstract. A recent breakthrough of Liu and Pass (FOCS'20) shows that one-way functions exist if and only if the (polynomial-)time-bounded Kolmogorov complexity, K^t, is bounded-error hard on average to compute. In this paper, we strengthen this result and extend it to other complexity measures:

- We show, perhaps surprisingly, that the KT complexity is bounded-error average-case hard if and only if there exist one-way functions in *constant parallel time* (i.e. NC^0). This result crucially relies on the idea of *randomized encodings*. Previously, a seminal work of Applebaum, Ishai, and Kushilevitz (FOCS'04; SICOMP'06) used the same idea to show that NC^0-computable one-way functions exist if and only if logspace-computable one-way functions exist.
- Inspired by the above result, we present randomized average-case reductions among the NC^1-versions and logspace-versions of K^t complexity, and the KT

A conference version of this paper appeared in the Proceedings of the 36th Computational Complexity Conference, 2021 [84]. A preprint of this article appeared in the Electronic Colloquium on Computational Complexity (ECCC) as Tech Report TR 21-057.

ACM Classification: F.1.3

AMS Classification: 68Q17

Key words and phrases: one-way functions, meta-complexity, KT complexity, parallel cryptography, randomized encodings

© 2026 Hanlin Ren and Rahul Santhanam
Licensed under a Creative Commons Attribution License (CC-BY)

complexity. Our reductions preserve both bounded-error average-case hardness and zero-error average-case hardness. To the best of our knowledge, this is the first reduction between the KT complexity and a variant of $\mathrm{K}^{\mathrm{poly}}$ complexity.

- We prove tight connections between the hardness of K^t complexity and the hardness of (the hardest) one-way functions. In analogy with the Exponential-Time Hypothesis and its variants, we define and motivate the *Perebor Hypotheses* for complexity measures such as K^t and KT. We show that a Strong Perebor Hypothesis for K^t implies the existence of (weak) one-way functions of near-optimal hardness $2^{n-o(n)}$. To the best of our knowledge, this is the first construction of one-way functions of near-optimal hardness based on a natural complexity assumption about a search problem.
- We show that a Weak Perebor Hypothesis for MCSP implies the existence of one-way functions, and establish a partial converse. This is the first unconditional construction of one-way functions from the hardness of MCSP over a natural distribution.
- Finally, we study the average-case hardness of MKtP. We show that it characterizes cryptographic pseudorandomness in one natural regime of parameters, and complexity-theoretic pseudorandomness in another natural regime.

1 Introduction

1.1 Background and motivation

1.1.1 Meta-complexity

Let μ be a complexity measure, such as the circuit size of a Boolean function or the time-bounded Kolmogorov complexity of a string. Traditional complexity theory studies the complexity measure on fixed functions, e.g. the AC^0 complexity of the Parity function. In contrast, we study the *meta-complexity* problem associated with μ: given an input function, what is its μ value?

Meta-complexity problems are fundamental to theoretical computer science and have been studied since the very beginning of the discipline [91]. They have connections to several areas of theoretical computer science, including circuit lower bounds, learning, meta-mathematics, average-case complexity, and cryptography. However, our knowledge about them is still very limited compared to our knowledge of other fundamental problems such as the Satisfiability problem.

Some of the basic complexity questions about meta-complexity include:

- Is computing a given measure μ complete for some natural complexity class? For example, is the Minimum Circuit Size Problem (MCSP, [65]) NP-complete?
- Can we show unconditional circuit lower bounds for computing μ, at least for weak circuit classes? Can we distinguish truth tables with $2^{o(n)}$-size circuits from random truth tables by a small $\mathsf{AC}^0[2]$ circuit?

- Is deciding whether μ is at least some parameter k robust to the choice of the parameter k? Let MCSP$[s(n)]$ denote the problem of whether an input function (represented as a truth table) has circuit complexity at most $s(n)$; are MCSP$[2^{n/2}]$ and MCSP$[2^{n/3}]$ computationally equivalent?

- How do low-level definitional issues affect the complexity of μ? Does the complexity of the time-bounded version of Kolmogorov complexity ("K^t") depend on the universal Turing machine that defines it?

- For which pairs of measures μ and μ' can we show that the problem of computing μ reduces to the problem of computing μ'? Can we reduce computing the time-bounded version of Kolmogorov complexity to computing circuit complexity?

There has been much interest in recent years in these questions. While there has been some progress on answering these questions for specific measures [4, 55, 3, 6, 81, 48, 38, 44, 54, 53], there are also barriers to understanding these questions better, such as our inability to prove circuit lower bounds [65, 75] and the magnification phenomenon [82, 80, 74, 25]. Many of the above questions such as the NP-completeness of MCSP remain wide open.

1.1.2 Cryptography

A fundamental question in cryptography is whether one-way functions exist. We have been quite successful at basing one-way functions on the hardness of specific problems such as factoring [85], discrete logarithm [28], and some lattice problems [1]. One problem with this approach, however, is that we have little complexity-theoretic evidence for the hardness of these problems (for example, they are unlikely to be NP-hard). The most compelling evidence for their hardness so far is simply that *we have not been able to find efficient algorithms for them*.

Can we base the existence of one-way functions on firm complexity-theoretic assumptions? A "holy grail" in this direction would be to construct one-way functions assuming (only) $\mathsf{NP} \not\subseteq \mathsf{BPP}$ [28]. This goal remains elusive, and there are several obstacles to its resolution:

- Unless PH collapses, non-adaptive "black-box" reductions cannot transform worst-case hardness of NP into average-case hardness of NP [19]. As the latter is necessary for one-way functions, this barrier result demonstrates limits of such "black-box" reductions on basing one-way functions from worst-case assumptions such as $\mathsf{NP} \not\subseteq \mathsf{BPP}$. For the task of constructing one-way functions (instead of just a hard-on-average problem in NP), stronger barrier results are known [2, 76].

- Even the seemingly easier task of basing one-way functions from *average*-case hardness of NP remains elusive. Indeed, Impagliazzo [57] called a world "Pessiland" where NP is hard on average but one-way functions do not exist. It is not hard to construct a relativized Pessiland [98], therefore a relativization barrier exists even for this "easier" task.

1.1.3 The Liu–Pass result

Very recently, in a breakthrough result, Liu and Pass [70] showed an *equivalence* between the existence of one-way functions and the *bounded-error average-case hardness* of computing the K^t complexity (the Kolmogorov complexity of a string with respect to a given polynomial time bound t) over the uniform distribution. This result is significant for several reasons.

- From the perspective of cryptography, it establishes the first equivalence between the existence of one-way functions and the average-case complexity of a natural problem over a natural distribution. Such an *equivalence* result bases cryptography on firmer complexity-theoretic foundations.
- From the perspective of meta-complexity, it enables *robustness* results for the complexity of K^t in the average-case setting. Indeed, [70] proved that *approximating* the K^t complexity of a string or *finding* an optimal description for a string are both equivalent to the problem of computing the K^t complexity.
- More generally, such connections suggest the possibility of new and non-trivial *average-case* reductions between natural problems on natural distributions, which is by itself an important goal in average-case complexity theory. Several of the most basic questions in this area remain open: Is random 3-SAT as hard as random 4-SAT (or vice versa)? Is the decision version of Planted Clique as hard as its search version?[1]

Given these motivations, it is natural to ask if the main result of [70] can be extended to other meta-complexity problems. For example, is the average-case hardness of MCSP also equivalent to the existence of one-way functions? There is a "Kolmogorov-version" of circuit complexity, named KT, which is more "fine-grained" than circuit complexity [3, 4]. Maybe this problem is also closely related to the existence of one-way functions? What about Levin's Kt complexity [67]?[2]

1.2 Our contributions

We give strong positive answers to the above questions. We show somewhat surprisingly that the average-case hardness of KT complexity is also equivalent to a natural cryptographic primitive, namely the existence of one-way functions computable in *fast parallel time*.[3] For MCSP, we obtain weaker results: exponential hardness of computing circuit size over the uniform distribution implies the existence of one-way functions, and there is a partial converse. Bounded-error average-case complexity of Kt complexity turns out to be equivalent to the existence of one-way

[1] The decision version of Planted Clique is to distinguish Erdős–Rényi random graphs from graphs with a planted clique; the search version is to find the planted clique. Three years after the conference version of our paper (CCC'21), Hirahara and Shimizu [50] showed that the two versions are equivalent.

[2] See Definition 3.3 for the precise definitions of K^t, KT, and Kt.

[3] Due to a result in [11], the "fast parallel time" here can be interpreted as either NC^0 or NC^1. We also point the reader to Benny Applebaum's book *Cryptography in Constant Parallel Time* [8], which inspired the title of the current paper.

functions in one natural setting of parameters (despite the fact that computing Kt in the worst case is EXP-hard [4]), and equivalent to the existence of complexity-theoretic pseudorandom generators in another natural setting of parameters.

We also extend the connection between the hardness of K^t complexity and one-way functions to the high end of the parametric regime — this yields one-way functions of almost optimal hardness from plausible assumptions about the hardness of K^t complexity. We define and motivate the Perebor Hypotheses[4], which are average-case analogues of the Exponential-Time Hypothesis and its variants for meta-complexity problems, stating that there is no better way to solve meta-complexity problems than brute force search. This is a conceptual contribution of this work, and we expect these hypotheses to have further applications to cryptography, average-case complexity, and fine-grained complexity.

We now describe our results in more detail.

1.2.1 Connections between meta-complexity and one-way functions

Our main result is an equivalence between "parallel cryptography" and the average-case hardness of MKTP:

Theorem 1.1 (Main Result; Informal)**.** *There is a one-way function computable in uniform* NC^1 *if and only if* KT *is bounded-error hard on average.*

The class "uniform NC^1" in the above theorem is somewhat arbitrary and can be replaced by any class between NC^0 and $\oplus\mathsf{L}$.[5] This is because [11] proved that the existence of one-way functions in $\oplus\mathsf{L}$ implies the existence of one-way functions in NC^0.

For comparison, Liu and Pass [70] showed an equivalence between ("sequential") cryptography and the average-case hardness of time-bounded Kolmogorov complexity (K^t).

Theorem 1.2 (Main Result of [70])**.** *There is a one-way function if and only if for some polynomial* t*,* K^t *is bounded-error hard on average.*

Theorem 1.2 shows that the one-way function defined based on hardness of K^t is a natural *universal* one-way function.[6] Similarly, Theorem 1.1 shows that the one-way function we define based on the hardness of KT is a natural universal one-way function in NC^1.

As a corollary, the classical open question of whether polynomial-time computable one-way functions imply one-way functions in NC^0 is equivalent to the question of whether average-case hardness of K^t implies average-case hardness of KT.

[4] Our terminology is inspired by Trakhtenbrot's survey [91] on work in the former Soviet Union aiming to show that various meta-complexity problems require brute force search to solve. "Perebor" roughly means "by exhaustive search" in Russian.

[5] $\oplus\mathsf{L}$ is the class of problems solvable by a *parity* Turing machine with $O(\log n)$ space. This class contains both NC^1 and L (log-space).

[6] A *universal one-way function* is a polynomial-time computable function f such that if any one-way function exists, then f is a one-way function. An artificial universal one-way function can be defined by enumerating uniform algorithms and concatenating their outputs [68, 34].

Results for MCSP**.** The KT complexity was defined as a variant of Kolmogorov-complexity that resembles circuit complexity [3, 4]. Therefore, it is natural to ask whether our equivalence also holds for circuit complexity.

It turns out that circuit complexity is less convenient to deal with. Nevertheless, we still proved non-trivial analogues of Theorem 1.1 as follows:

Theorem 1.3 (Informal)**.** *The following are true:*

- *If* MCSP *is exponentially*[7] *hard on average, then there is a (super-polynomially hard) one-way function.*
- *If there is an exponentially hard weak one-way function in* NC^0*, then* MCSP *is (exponentially) hard on average.*

For the technical difficulties of handling circuit complexity, the reader is referred to Section 6 (and in particular Remark 6.9).

Results for MKtP**.** We also observe that the existence of (polynomial-time computable) one-way functions can be characterized by the bounded-error average-case complexity of Kt.

Theorem 1.4. *There is a one-way function if and only if* Kt *is bounded-error hard on average.*

This result may seem surprising as computing Kt is EXP-hard under polynomial-size reductions [4]. This is true even for any oracle that is a *zero-error* heuristic for computing Kt. In contrast, we show that the *bounded-error* average-case complexity of Kt is captured by one-way functions, a notion that seems much "easier" than EXP.

The harder direction in Theorem 1.4 is to construct a one-way function from the hardness of Kt. How could we construct a one-way function from merely a hard problem *in exponential time*? The crucial insight is as follows: For *most* strings $x \in \{0,1\}^n$ whose optimal Kt complexity is witnessed by a machine d and a time bound t where $\mathrm{Kt}(x) = |d| + \log t$, we have $t \leq \mathrm{poly}(n)$. We refer the reader to Section 2.1.2 and Section 7 for more details.

Note that Theorem 1.4 can also be seen as a characterization of cryptographic pseudorandomness, by the known equivalence between one-way functions and cryptographic pseudorandomness [42]. In a different regime of parameters, average-case hardness of Kt turns out to capture the existence of *complexity-theoretic* pseudorandom generators, which are pseudorandom generators with non-trivial seed length computable in *exponential* time. Thus the average-case complexity of a single problem (Kt) can be used to capture both cryptographic pseudorandomness and complexity-theoretic pseudorandomness!

The result below is essentially a corollary of known results from [4, 78].

Theorem 1.5 (Informal)**.** *For each* $\epsilon > 0$*, there is a pseudo-random generator from* n^ϵ *bits to* n *bits computable in time* $2^{n^\epsilon}\mathrm{poly}(n)$ *secure against* $\mathrm{poly}(n)$ *size circuits iff for each* $c > 1/2$ *there are no polynomial-size circuits solving* Kt *on more than* $1 - 2^{-cn}$ *fraction of inputs of length* n*.*

[7] In this paper, *exponential* hardness always means hardness $2^{\Omega(n)}$.

1.2.2 Application in meta-complexity: Robustness theorems

We exploit the connection between MKTP and parallel cryptography to establish more robustness results for meta-complexity. It is known that parallel cryptography is *extremely* robust: L-computable one-way functions exist, if and only if NC^1-computable one-way functions exist, if and only if NC^0-computable one-way functions exist [11]. We define L- and NC^1-variants of K^t complexity, and translate the result in [11] to the following robustness theorem:

Theorem 1.6 (Bounded-Error Robustness of Meta-Complexity; Informal)**.** *The following statements are equivalent:*

- *For $t_1(n) := n^{10}$, the search version of NC^1-K^{t_1} is bounded-error hard on average.*
- *For $t_2(n) := 5n$, L-K^{t_2} is bounded-error hard on average to approximate, within an additive error of $100 \log n$.*
- KT *is bounded-error hard on average.*

It turns out that the above theorem can be interpreted as a *reduction*! In particular, we discover the following average-case reduction from L-K^t to MKTP:

Theorem 1.7 (Informal)**.** *Let n, t be parameters, $m := \mathrm{poly}(n, t)$. There is a randomized reduction $\mathsf{Red}(x)$ running in $\mathrm{poly}(m)$ time that maps a length-n input to a length-m input, and satisfies the following property:*

- *Given a uniform random input x of length n, $\mathsf{Red}(x)$ produces a uniform random string of length m.*
- *Given a string x such that L-$\mathrm{K}^t(x)$ is small, for every possible randomness used in Red, the* KT *complexity of $\mathsf{Red}(x)$ is also small.*

To the best of our knowledge, this is the first reduction from a variant of K^t complexity to a variant of KT complexity. The only special property of L that we use is that L-computable functions have *perfect randomized encodings* [11]. If polynomial-time computable functions have such perfect randomized encodings, then our techniques imply an average-case reduction from the (standard) K^t complexity to the KT complexity.

We have focused on the *bounded-error* average-case complexity of meta-complexity problems so far. However, Theorem 1.7 also implies robustness in the *zero-error* regime. Here, let $\mathrm{MKTP}[s]$ be the problem of determining whether the input x satisfies $\mathrm{KT}(x) \leq s(|x|)$, and let $\mathrm{MINK}^t[s]$ be the problem of determining whether the input x satisfies $\mathrm{K}^t(x) \leq s(|x|)$.

Theorem 1.8 (Zero-Error Robustness of Meta-Complexity; Informal)**.** *Let $t_1(n) := n^{10}$ and $t_2(n) := 5n$. Among the following items, we have (1) $\iff$ (2), and both items are implied by (3).*

1. *There is a constant $c > 0$ such that NC^1-$\mathrm{MINK}^{t_1}[n - c \log n]$ is zero-error easy on average.*
2. *There is a constant $c > 0$ such that L-$\mathrm{MINK}^{t_2}[n - c \log n]$ is zero-error easy on average.*
3. *There is a constant $c > 0$ such that $\mathrm{MKTP}[n - c \log n]$ is zero-error easy on average.*

1.2.3 Application in cryptography: Maximally hard one-way functions

How hard can a one-way function be? The standard definition of one-way functions only requires that no polynomial-time adversary inverts a random output except with negligible probability. However, it is conceivable that some one-way function requires $2^n/\mathrm{poly}(n)$ time to invert (say, on a constant fraction of inputs)!

When studying the fine-grained complexity of inverting one-way functions, it is useful to draw an analogy to the theory of Satisfiability, where the Exponential Time Hypothesis (ETH) and Strong Exponential Time Hypothesis (SETH) [59, 60, 22] have gained prominence recently. ETH states that CNF-SAT requires time $2^{\Omega(n)}$ to solve, and SETH states that CNF-SAT requires time $2^{n-o(n)}$ to solve. Analogously, we define the *Weak* and *Strong* Perebor Hypotheses for any given complexity measure μ. The Weak Perebor Hypothesis for μ states that μ requires time $2^{\Omega(n)}$ to compute on average, and the Strong Perebor Hypothesis for μ states that μ requires time $2^{n-o(n)}$ to compute on average. Note that the Strong Perebor Hypothesis for μ essentially states that brute-force search is close to optimal for computing μ on average. While the Perebor Hypotheses refer by default to hardness against uniform algorithms, we can define non-uniform analogues in a natural way. (Unfortunately, due to the lack of a uniform hardcore lemma with good enough parameters, the results here only hold for the non-uniform Perebor Hypotheses.)

We suggest that the Perebor Hypotheses might find similar applicability to ETH and SETH in the theory of fine-grained complexity.

In this work, we tighten the connection between *weak* one-way functions (for which it is hard to invert a random instance w.p. $1 - 1/\mathrm{poly}(n)$) and the hardness of K^t complexity. We managed to show a very tight result:

Theorem 1.9 (Informal)**.** *For every constant $\alpha > 0$, there exists a weak one-way function with non-uniform hardness $2^{(1-o(1))\alpha n}$ if and only if K^t complexity is hard on average for non-uniform algorithms of size $2^{(1-o(1))\alpha n}$.*

Note that the two α's in the exponent $(1 - o(1))\alpha n$ are the *same*. That is, we essentially construct the best (weak) one-way functions from the hardness of K^t complexity. In particular, almost maximally hard one-way functions exist if the (non-uniform version of the) Strong Perebor Hypothesis holds for K^t.

We also attempted to strengthen the relationship between one-way functions in NC^0 and the hardness of KT complexity. Our result is that exponentially-hard weak one-way functions in NC^0 imply exponential hardness of KT.

Theorem 1.10 (Informal)**.** *If there is a weak one-way function in NC^0 with non-uniform hardness $2^{\Omega(n)}$, then* KT *requires $2^{\Omega(n)}$ circuit size to solve on average.*

In other words, the existence of an exponentially hard weak one-way function in NC^0 implies the non-uniform version of the Weak Perebor Hypothesis for KT.

1.3 Related work

There have been several previous works connecting meta-complexity to cryptography. Impagliazzo and Levin [58] show that the existence of one-way functions is equivalent to the hardness of

a certain learning task related to time-bounded Kolmogorov complexity, and follow-up results in this direction are shown by Blum, Furst, Kearns, and Lipton [18]. Oliveira and Santhanam [81] show a dichotomy between learnability and cryptographic pseudorandomness in the non-uniform setting: there is a non-trivial non-uniform learner for polynomial-size Boolean circuits iff there is no exponentially secure distribution on functions computable by polynomial-size circuits. Santhanam [86] proves an equivalence between the existence of one-way functions and the non-existence of natural proofs under a certain universality assumption about succinct pseudorandom distributions. We note here that the non-existence of natural proofs is equivalent to the zero-error average-case hardness of MCSP [48].

None of the above results gives an unconditional equivalence between the average-case hardness of a natural decision problem over a natural distribution and the existence of one-way functions. This was finally achieved by Liu and Pass [70], who showed that the weak hardness of $\mathrm{K}^{\mathrm{poly}}$ over the uniform distribution is equivalent to the existence of one-way functions. [70] leaves open whether there are similar connections between one-way functions and the hardness of other meta-complexity problems such as KT and MCSP over the uniform distribution. In this work, we show such connections to *parallel* cryptography, i.e., to the existence of one-way functions in NC^1, which by [11] is equivalent to the existence of one-way functions in NC^0.

There is an extensive literature on parallel cryptography, beginning with the work of [11]. We refer to [8] and [9] for further information.

Our work also relates to average-case meta-complexity, which was studied in [83, 48]. [48] essentially observe that the identity reduction trivially reduces μ to μ' over the uniform distribution in a zero-error average-case sense, where μ and μ' are any two meta-complexity measures such that $\mu'(x) \leq \mu(x) \leq |x| + O(\log(|x|))$ for all x. In this work (particularly Sections 4.3 and 4.4), we give several *non-trivial* examples of zero-error and bounded-error average-case reductions between meta-complexity problems.

Concurrent works of [71] and [5]. We now discuss the relationship of our work with the concurrent works of [71] and [5], which overlap in some respects with ours.

Liu and Pass [71] show an equivalence between the bounded-error weak average-case hardness of Kt over the uniform distribution and the existence of one-way functions — this is essentially the same as our Theorem 1.4. They also show that the *zero-error* average-case hardness of Kt over the uniform distribution is equivalent to $\mathsf{EXP} \neq \mathsf{BPP}$. In contrast, our Theorem 1.5 gives an equivalence between the bounded-error average-case hardness of Kt over the uniform distribution in a different parametric regime and the worst-case hardness of EXP, where the hardness in each case is with respect to non-uniform adversaries. The somewhat surprising message of both sets of results is the same: a minor variation on an average-case complexity assumption that is equivalent to the worst-case hardness of EXP implies the existence of one-way functions.

[71] also give characterizations of parallel cryptography but they do this using space-bounded Kolmogorov complexity and the conditional version thereof. Their work does not contain any results relating to the hardness of KT or MCSP.

Allender, Cheraghchi, Myrisiotis, Tirumala, and Volkovich [5] relate the average-case

hardness of the conditional version of KT complexity (McKTP) over the uniform distribution to the existence of one-way functions. They show that if McKTP is hard on a polynomial fraction of instances then one-way functions exist. They also give a weak converse: if one-way functions exist, then McKTP is hard on an exponential fraction of instances. Finally, using our techniques, they show that the average-case hardness of McKTP characterizes parallel cryptography as well.

Further work on the Perebor Hypotheses. There are several works on the Perebor Hypotheses after our paper becomes available online. Carmosino, Hoover, Impagliazzo, Kabanets, and Kolokolova [24] demonstrated barriers against obtaining "natural" reductions between different variants of AC^0-MCSP, under a weak form of Perebor Hypothesis for AC^0-MCSP. More recently, Mazor and Pass [73], and independently Hirahara, Ilango, and Williams [47], disproved the non-uniform version of the Strong Perebor Hypothesis for $\mathrm{K}^{\mathrm{poly}}$ (Hypothesis 5.19), thereby confirming the speculations we raised in Section 5.5.

1.4 Organization

Section 2 presents some of our main ideas and techniques. Section 3 provides basic definitions and preliminaries.

The equivalence between the existence of NC^0-computable one-way functions and the hardness of KT complexity is proved in Section 4. We prove our robustness results in Section 4.3 and 4.4. In Section 5, we present the tight connection between the hardness of K^t complexity and maximally hard one-way functions. To motivate future study, we put forward a few Perebor Hypotheses in Section 5.5, which are closely related to the existence of maximally-hard one-way functions. The results related to MCSP are proved in Section 6, and the results related to MKtP are proved in Section 7. Finally, we leave a few open questions in Section 8.

2 Intuitions and techniques

For strings $s_1, s_2, \ldots, s_n$, we use $s_1 \circ s_2 \circ \cdots \circ s_n$ to denote their concatenation.

2.1 Parallel cryptography and the hardness of KT

Our proof of Theorem 1.1 builds on [70]. However, it turns out that we need new ideas for both directions of the equivalence.

2.1.1 Hardness of KT from one-way functions in NC^0

We first review how Liu and Pass [70] proved that one-way functions imply average-case hardness of K^t.

Any cryptographically-secure PRG G implies *zero-error* hardness of K^t [83, 65, 4]. Roughly speaking, the outputs of G have "non-trivial" K^t complexity, but random strings are likely to

have "trivial" K^t complexity.[8] If there is a polynomial-time (zero-error) heuristic for K^t, this heuristic will recognize most random strings as "trivial", but recognize every output of G as "non-trivial". Thus, we can use it as a distinguisher for G, contradicting the security of G.

It is crucial in the above argument that our heuristic does not make mistakes. If the outputs of G are "sparse" and our heuristic has two-sided error, our heuristic could also recognize the outputs of G as "non-trivial". (Here, a PRG G with output length n is sparse if the number of possible outputs of G is significantly smaller than 2^n.) In this case, the heuristic may still be correct on most length-n strings but fail to distinguish the outputs of G from true random strings.

Why not *make G dense*? This is the core idea of Liu and Pass. In particular, from an arbitrary one-way function f, they constructed a *dense* PRG G,[9] and used G to argue that K^t is bounded-error average-case hard. Roughly speaking, if the outputs of G occupy a $1/\mathrm{poly}(n)$ fraction of $\{0,1\}^n$, then any bounded-error heuristic for K^t with error probability $1/n^{\omega(1)}$ is a distinguisher for G. It follows from the security of G that K^t is bounded-error hard on average.

What about KT**?** Recall that the KT complexity of a string x is the minimum of $|d| + t$ over programs d and integers t such that x can be generated *implicitly* from d in at most t steps, i.e., the universal machine computes the i-th bit x_i of x correctly in at most t steps with oracle access to d.

When we use the above framework to analyze the hardness of KT complexity, there is a problem: the outputs of G might have "trivial" KT complexity. Indeed, if G is a *dense* PRG, then it has at least $2^n/\mathrm{poly}(n)$ many different outputs and hence its seed length is at least $n - O(\log n)$. Furthermore, if G runs in $t(n)$ time, then we can only upper bound the KT complexity of its outputs by $n - O(\log n) + t(n)$. For most PRGs G, $t(n)$ would be a large polynomial, hence this upper bound is even worse than the trivial upper bound (which is $n + O(\log n)$). However, if G is computable in *constant parallel time*[10], then our upper bound on the KT complexity is still non-trivial. In other words, a dense PRG computable in NC^0 still suffices for the bounded-error average-case hardness of KT.

We can construct an NC^0-computable PRG from NC^0-computable one-way functions as follows.[11] We first use [70] to construct a dense PRG G. This PRG is not necessarily in NC^0, as [70] needs some more complex primitives (e.g. extractors). Nevertheless, we can apply the randomized encodings in [11] to compile G into a PRG in NC^0.

[8] Here, the K^t (or KT) complexity of a length-n string is "non-trivial", if it is at most $n - \Omega(\log n)$. Most length-n strings have complexity at least $n - \Omega(\log n)$; every length-n string has complexity at most $n + O(\log n)$ (justifying the word "trivial").

[9] The input distribution of their PRG is not the uniform distribution, which is different from standard PRGs; see Definition 3.13. We ignore this difference in the informal exposition.

[10] Due to low-level issues in the computational models, the "constant time" in [8] actually corresponds to $O(\log n)$ time in this paper. See Section 3.1 for details.

[11] Note that this is different from [42, 41]. The PRG we construct is dense, but its input distribution is not uniform. In contrast, [42, 41] constructs a PRG (on uniformly random inputs) from an arbitrary one-way function, but the PRG is not necessarily dense.

2.1.2 One-way functions in NC^0 from the hardness of KT

It is straightforward to construct a one-way function from the hardness of KT using techniques of [70, Section 4]. Roughly speaking, the one-way function f receives two inputs d, t, where d is the description of a machine and t is a time bound. Let x be the string such that for each $i \in [n]$, x_i is equal to the output bit of $d(i)$ for t steps. We define $f(d,t) := (|d| + t, x)$. An inverter, on input (ℓ, x), is required to find a description of x with complexity at most ℓ, thus it needs to solve MKTP. (All one-way functions in this section are *weak*, meaning that no adversary can efficiently invert them on a $1 - 1/\mathrm{poly}(n)$ fraction of inputs.)

There is one problem: f is not in NC^0. By [11], it suffices to construct a one-way function in $\oplus\mathsf{L}$, but f is also not in $\oplus\mathsf{L}$ (unless $\oplus\mathsf{L} = \mathsf{P}$).

Our idea is to only consider *typical* inputs and throw away the atypical ones. In particular, for most strings x, the values of t in the optimal description of $\mathrm{KT}(x) = |d| + t$ are small. (We have $t = O(\log n)$ for every string x with Kolmogorov complexity at least $n - O(\log n)$.) We call an input *typical* if its value of t is at most $O(\log n)$. If KT is (bounded-error) hard on average, then it is also hard on average conditioned on the input being typical.

Therefore, we place the restriction that $t \le c \log n$ in our one-way function f, where c is a constant depending on the hardness of KT. We can still base the hardness of f on the hardness of KT. More importantly, f is computable in space complexity $O(c \log n)$ and we obtain a one-way function in NC^0 by [11].

2.2 Applebaum–Ishai–Kushilevitz as a reduction

For any "reasonable" circuit class $\mathscr{C}$, we can use [70] to show that the existence of one-way functions computable in $\mathscr{C}$ is equivalent to the hardness of $\mathscr{C}$-K^t. (The precise definition of $\mathscr{C}$-K^t is beyond the scope of this paper, but NC^1-K^t and L-K^t are defined in Definition 3.4.) Now, let us review the main results of [11]: $\oplus\mathsf{L}$-computable one-way functions exist if and only if NC^0-computable one-way functions exist. In other words, $\oplus\mathsf{L}$-K^t is hard on average if and only if NC^0-K^t is hard on average![12]

It is natural to ask whether there is a reduction between $\oplus\mathsf{L}$-K^t and NC^0-K^t. It turns out that the answer is *yes*! In this section, we describe this reduction without using the language of one-way functions. This reduction is randomized, reduces any string with non-trivial $\oplus\mathsf{L}$-K^t complexity to a string with non-trivial NC^0-K^t complexity, and reduces a random string to a random string. Although it may not be a worst-case reduction, it establishes non-trivial equivalence results between average-case complexities of $\oplus\mathsf{L}$-K^t and NC^0-K^t.

The property that enables our reduction is *resamplability* [29]. For now, think of "easy" as being NC^0-computable and "hard" as otherwise. A hard function f is *resamplable* if given an input x and random coins r, there is an easy procedure (the "resampler") that produces a uniformly random input of f whose answer is the same as x.

Example 2.1. The parity function $\mathrm{PARITY}(x) = x_1 \oplus x_2 \oplus \cdots \oplus x_n$ is hard (i.e., not computable in NC^0). Given n input bits $x_1, x_2, \ldots, x_n$ and $n-1$ random bits $r_1, r_2, \ldots, r_{n-1}$, we can produce a

[12] NC^0 may not be "reasonable" in the above sense, but the reduction we present in this section is correct.

uniformly random input whose answer is the same as x, as follows:

$$(x_1 \oplus r_1, x_2 \oplus r_1 \oplus r_2, x_3 \oplus r_2 \oplus r_3, x_4 \oplus r_3 \oplus r_4, \ldots, x_{n-1} \oplus r_{n-2} \oplus r_{n-1}, x_n \oplus r_{n-1}).$$

Note that the resampler is easy (i.e., in NC^0), thus parity is resamplable.

The reduction. We will use a $\oplus\mathsf{L}$-complete problem named DCMD that is resamplable (see Section 3.7). Our reduction is very simple: given an input $x \in \{0,1\}^n$, we choose a large enough $N = \mathrm{poly}(n)$ and replace every bit x_i by a random length-N instance of DCMD whose answer is x_i. Our reduction outputs the concatenation of these n instances.

Since DCMD is balanced (i.e., the number of 0-instances and 1-instances are the same), our reduction maps a random instance to a random instance.

Now assume that $\oplus\mathsf{L}\text{-}\mathrm{K}^t(x) = n - \gamma$ is non-trivial, and d is a $\oplus\mathsf{L}$ machine of description length $n - \gamma$ that "computes" x. Since DCMD is $\oplus\mathsf{L}$-complete (under NC^0-reductions), for each i, the computation of x_i can be reduced to a DCMD-instance s_i of length N such that $\mathrm{DCMD}(s_i) = x_i$. Moreover, given the description d, we can produce $s_1 \circ s_2 \circ \cdots \circ s_n$ in NC^0.

We use the *resamplability* of DCMD. The resampler for DCMD only uses $N - 1$ random bits (which is optimal). Consider the following NC^0 circuit. It receives d and $r_1, r_2, \ldots, r_n$ as inputs where each r_i is a random string of length $N - 1$. It computes $s_1, s_2, \ldots, s_n$ from d, and for each i, feeds s_i and r_i to the resampler to obtain a uniformly random DCMD instance whose answer is the same as s_i. When r_i are random bits, the output distribution of this NC^0 circuit is identical to the distribution of $\mathsf{NC}^0\text{-}\mathrm{K}^t$ instances we reduced x to. Moreover, the $\mathsf{NC}^0\text{-}\mathrm{K}^t$ complexity of *every* string in this distribution is at most $(n - \gamma) + (N - 1)n + O(\log n) = Nn - \gamma + O(\log n)$, which is non-trivial.[13]

As a consequence, we also obtain an (average-case) reduction from $\oplus\mathsf{L}\text{-}\mathrm{K}^t$ to KT.

2.3 Tighter connections

To obtain a tight relationship between the hardness of K^t and the hardness of weak one-way functions, we optimize the construction from one-way functions to PRGs in [70]. Suppose that given a one-way function f with input length n, we could construct a PRG with output length m'. Then solving K^t on length m' is (roughly) as hard as inverting f on length n. Therefore, we need m' to be as close to n as possible. As the PRG is dense, its output length m' is close to its input length m, thus we only need m to be close to n.

It turns out that the input of the PRG consists of the input of f and the seeds of a few pseudorandom objects.

- One object is an *extractor* $\mathsf{Ext}(\mathcal{X}, r)$ [79, 77], which given a "somewhat random" distribution $\mathcal{X}$ and a truly random seed r, outputs a distribution that is statistically close to the uniform distribution.

  We use the near-optimal explicit extractors with $O(\log^2 n)$ seed length [40].

[13] The additive factor here is $O(\log n)$ since in our computational model, each memory access requires $\Theta(\log n)$ time. See Section 3.1 for details.

- Another object is a *hardcore function* $\mathsf{HC}(x, r)$ [36]. Let f be a one-way function, x be a random input, and r be a random seed. Given $f(x) \circ r$, it should be infeasible to distinguish between $\mathsf{HC}(x, r)$ and a uniformly random string. Note that $\mathsf{HC}(x, r)$ needs to have multiple output bits; in contrast, a *hardcore predicate* (also defined in [36]) only has one output bit.

  We use the observation, implicit in [92, 89], that any seed-extending "black-box" pseudorandom generator is a good hardcore function. We use the *direct product* generator [46, 45] as our hardcore function, which has $O(\log^2 n)$ seed length, and very small "advice complexity." The advice complexity turns out to be related to the overhead of our reduction.

There is another problem: [70] needs a *strong* one-way function to start with, but we only have a *weak* one-way function. (A strong one-way function is infeasible to invert on *almost every* input, but a weak one-way function is only infeasible to invert on *a non-trivial fraction* of inputs.) Yao [102] showed how to "amplify" a weak one-way function to a strong one-way function, but the overhead of this procedure is too large. In particular, Yao's hardness amplification does not preserve *exponential* hardness, and it is open whether exponentially-hard weak one-way functions imply exponentially-hard strong one-way functions.

Our idea is to use *Impagliazzo's hardcore lemma* [56] instead. The hardcore lemma states that for any weak one-way function f, there is a "hardcore" distribution on which f becomes a strong one-way function. We (and [70]; see Footnote 9) allow the input distribution of our PRG to be *arbitrary*, as long as the output distribution is pseudorandom. Such "PRGs" still imply hardness of K^t. The hardcore lemma only has a small complexity overhead, which allows us to prove tight results.

Now, from a weak one-way function of input length n, we can construct a PRG with output length $n + O(\log^2 n)$. This construction allows us to transform the hardness of one-way functions to the hardness of K^t at almost no cost.

Tighter connections between MKTP **and one-way functions in** NC^0**.** Here, the problem becomes to construct NC^0-computable PRGs from NC^0-computable one-way functions. We use a construction of universal hash functions in NC^0 with linear seed length by Applebaum [10]. Such hash functions are both good extractors (by the leftover hash lemma) and good hardcore functions (proved in [16, 51]). As the hash functions require linear seed length, from an NC^0-computable one-way function with input length n, we obtain an NC^0-computable PRG with output length $O(n)$. It follows that if the one-way function is hard against $2^{\Omega(n)}$-size adversaries, then MKTP is also hard against $2^{\Omega(n)}$-size algorithms.

2.4 MCSP-related results

One-way functions from hardness of MCSP**.** We use the straightforward construction: our one-way function receives a circuit C and outputs $|C|$ and $tt(C)$, where $|C|$ is the size of C and $tt(C)$ is the truth table of C. The inverter, on input (s, tt), is required to find a size-s circuit whose truth table is tt, thus needs to solve MCSP.

One problem with this construction is that if we sample a uniform circuit (according to some distribution), the induced distribution over truth tables may not be uniform. In the case of K^t (and KT), we can show that for every string of length n, its optimal description is sampled (in the one-way function experiment) w.p. at least $2^{-n}/\mathrm{poly}(n)$, therefore we can "transfer" the hardness of K^t over a random truth table to the hardness of inverting the one-way function over a random description. Using the best bounds on the maximum circuit complexity of n-bit Boolean functions [31], we can still prove that for every truth table of length N, its optimal circuit is sampled w.p. at least $2^{-N}/2^{\eta}$, where $\eta < o(N)$. This means that starting from *exponential* hardness of MCSP, we can still obtain non-trivial one-way functions.

We conjecture that hardness of MCSP actually implies one-way functions in NC^0; see Remark 6.9 for details.

Hardness of MCSP **from one-way functions in** NC^0**.** To argue about the hardness of MCSP, we need a PRG whose outputs have non-trivial circuit complexity. As before, we use the hash functions in [10] to construct an exponentially-hard PRG. We would like to argue that all outputs of the PRG have non-trivial circuit complexity. In order to do this, we use the mass production theorem of Uhlig [93, 94] to generate a circuit of size $(1 + o(1))2^n/n$ that evaluates a given function on *multiple* inputs. (If our PRG has locality d, i.e., each output bit depends on d input bits, then we need a size-$(1 + o(1))2^n/n$ circuit that evaluates d inputs in parallel.) However, Uhlig's theorem only gives us non-trivial circuit size if our PRG has *linear* stretch, i.e., stretch εn for some constant $\varepsilon > 0$. This is why we need the hardness of the one-way function in our assumption to be at least $\mathrm{poly}(2^{\varepsilon n})$.

2.5 Using hardness of Kt to capture cryptographic and complexity-theoretic pseudorandomness

To show Theorem 1.4, we use ideas similar to those in Section 2.1.2. Suppose we try to define a one-way function by computing the string corresponding to an optimal description with respect to Kt complexity. An obvious issue is that such strings might require exponential time to compute, while the one-way function needs to be evaluated efficiently. However, we observe that *typical* inputs only require polynomial time to generate from their optimal descriptions. Here, the typical inputs are those with Kolmogorov complexity $n - O(\log n)$. In their optimal descriptions $\mathrm{Kt}(x) = |d| + \log t$, we have $t \leq \mathrm{poly}(n)$. Our one-way function receives two inputs d, t, where d is the description of a machine, and $t \leq \mathrm{poly}(n)$ is a time bound. We simply simulate the machine d for t steps and output what it outputs. The proof that this gives a one-way function is closely analogous to the proof of the reverse implication in Theorem 1.1. The proof that one-way functions imply the average-case hardness of Kt complexity mimics the proof of the corresponding implication in Theorem 1.2, since the outputs of a cryptographic PRG with stretch $\lambda \log n$ have non-trivial Kt complexity when λ is large enough compared to the time required to compute the PRG.

To show Theorem 1.5, we use the Nisan–Wigderson generator [78] in a way similar to how it is used by [4] to show that Kt is complete for exponential time under polynomial-size reductions.

The interesting direction is to show that the Nisan–Wigderson generator implies the average-case hardness of Kt for the range of parameters in the statement of Theorem 1.5. We use the fact that the Nisan–Wigderson generator can be made seed-extending without loss of generality. We truncate the output of the generator so that the stretch is $(1+\epsilon)n$ for some small $\epsilon > 0$ — this implies that the outputs of the generator on all seeds have non-trivial Kt complexity. Since the generator is seed-extending, the output has high entropy, hence a strong enough average-case algorithm for Kt can distinguish random strings (which have trivial Kt complexity) from the outputs of the PRG. Here we take advantage of the stretch being *small* rather than *large*: this gives us better parameters for our average-case hardness result.

3 Preliminaries

We use $\mathcal{U}_n$ to denote the uniform distribution over length-n binary strings. For a distribution $\mathcal{D}$, we use $\mathbf{x} \leftarrow \mathcal{D}$ to denote that $\mathbf{x}$ is a random variable drawn from $\mathcal{D}$. A function $\mathrm{negl} : \mathbb{N} \to [0,1]$ is *negligible* if for every constant c, $\mathrm{negl}(n) \leq 1/n^c$ for large enough integers n.

Let $D : \{0,1\}^n \to \{0,1\}$ be a function and X, Y be two random variables over $\{0,1\}^n$. For $\varepsilon > 0$, we say D *ε-distinguishes* X from Y if

$$|\Pr[D(X) = 1] - \Pr[D(Y) = 1]| \geq \varepsilon.$$

Otherwise we say X and Y are *ε-indistinguishable* by D.

We often consider ensembles of functions in this paper. For example, a function $f : \{0,1\}^\star \to \{0,1\}^\star$ can be interpreted as an ensemble $f = \{f_n : \{0,1\}^n \to \{0,1\}^\star\}$; we call each f_n the *n-th slice* of f. Similarly, we also consider ensemble of distributions $\mathcal{D} = \{\mathcal{D}_n\}$ as input distributions for a function f where each $\mathcal{D}_n$ is a distribution over $\{0,1\}^n$.

3.1 Computational model and uniformity

We need a computational model with random access to inputs. We consider a Turing machine that accesses the length-n input x via an "address" tape and a length-$O(1)$ "answer" tape. Whenever the machine enters a particular "address" state, let i be the binary number written in the address tape. After one step, the content of the answer tape becomes x_i, and the address tape is cleared. (In other words, the Turing machine treats x as the truth table of an *oracle*.)

We also assume that the address tape has length $\lceil \log n \rceil$. In particular, there are two special markers at the address tape, and there are $\lceil \log n \rceil$ cells strictly between them. The machine can only modify this portion of $\lceil \log n \rceil$ cells; the rest of the address tape is read-only. For sub-linear time Turing machines, this can be viewed as a mechanism to provide information about n (i.e., the length of x; up to a factor of 2). We also require that whenever the machine enters the "address" state, all the $\lceil \log n \rceil$ cells between the two markers are non-empty, hence we can interpret the concatenation of these cells as a (binary) address.

Every bit operation takes one step. Therefore, it takes $\Theta(\log n)$ time to write down an address. Note that we clear the address tape after each access, which means when we access another input bit, we have to spend another $\Theta(\log n)$ time to write down the address from

scratch. This definition ensures that in $O(\log n)$ time we can only access a constant number of input bits, so DLOGTIME becomes a natural uniform analog of NC^0.

In addition to the address tape and the answer tape, we also have a constant number of work tapes. In the case that our Turing machine computes a multi-output function f, we also provide an input tape that contains an index i (note that our *real* input is the "oracle" x), which means our Turing machine outputs the i-th bit of $f(x)$. We use $M^x(i)$ to denote the output of the machine M on input i given oracle access to the string x. To measure the space complexity of our Turing machine, we assume the input tape is read-only and we only count the total length of work tapes.

Definition 3.1. Let $c > 0$ be a constant, $p(\cdot)$ be a polynomial, and $F = \{F_n : \{0,1\}^n \to \{0,1\}^{p(n)}\}$ be an ensemble of functions. We say $F \in \mathsf{DTIME}[c \log n]$ if there is a Turing machine M with running time $c \log n$ such that for every sufficiently large n, every $x \in \{0,1\}^n$, and every $1 \le i \le p(n)$, $M^x(n,i)$ outputs the i-th bit of $F_n(x)$.[14]

Let $\mathsf{DLOGTIME} = \bigcup_{c \ge 1} \mathsf{DTIME}[c \log n]$.

Definition 3.2. Let $c > 0$ be a constant, $p(\cdot)$ be a polynomial, and $F = \{F_n : \{0,1\}^n \to \{0,1\}^{p(n)}\}$ be an ensemble of functions.

We say F is in $\mathsf{ATIME}[c \log n]$, if there is an *alternating* Turing machine M of $c \log n$ running time such that for every large enough n, every $x \in \{0,1\}^n$, every $1 \le i \le p(n)+1$, and every $b \in \{0,1,\star\}$, $M^x(n,i,b) = 1$ if the i-th bit of $F_n(x)$ is b.

We say F is in $\mathsf{SPACE}[c \log n]$, if there is a Turing machine M of space complexity $c \log n$ that satisfies the above requirement. We say F is in uniform $\oplus\mathsf{SPACE}[c \log n]$, if there is a *parity* Turing machine M of space complexity $c \log n$ that satisfies the above requirement.

Let $\mathsf{ALOGTIME} = \mathsf{NC}^1 = \bigcup_{c \ge 1} \mathsf{ATIME}[c \log n]$. (That is, in this paper, we use ALOGTIME and NC^1 interchangeably.) We also define $\mathsf{L} = \bigcup_{c \ge 1} \mathsf{SPACE}[c \log n]$ and $\oplus\mathsf{L} = \bigcup_{c \ge 1} \oplus\mathsf{SPACE}[c \log n]$.

For the readers not familiar with parity Turing machines, we provide an alternative definition of $\oplus\mathsf{L}$ by its complete problems; see Section 3.7.

3.2 Resource-bounded Kolmogorov complexity

We define some variants of resource-bounded Kolmogorov complexity. In particular, we define the plain Kolmogorov complexity K, the KT complexity [3], the time-bounded Kolmogorov complexity K^t [66], and Levin's Kt complexity [67]. Then we define the NC^1- and L-versions of K^t.

Definition 3.3. Let U be a Turing machine and x be a string. Artificially let $x_{|x|+1} = \star$.

- $\mathrm{K}_U(x)$ is the minimum $|d|$ over the description $d \in \{0,1\}^\star$ such that for every $1 \le i \le |x|+1$ and $b \in \{0,1,\star\}$, $U^d(i,b)$ accepts if and only if $x_i = b$.
- $\mathrm{KT}_U(x)$ is the minimum value of $|d|+t$ over the pairs (d,t) such that for every $1 \le i \le |x|+1$ and $b \in \{0,1,\star\}$, $U^d(i,b)$ accepts in t steps if and only if $x_i = b$.

[14] Note that n and i are encoded in binary using $O(\log n)$ bits here.

- $\mathrm{Kt}_U(x)$ is the minimum value of $|d| + \log t$ over the pairs (d, t) such that for every $1 \le i \le |x| + 1$ and $b \in \{0, 1, \star\}$, $U^d(i, b)$ accepts in t steps if and only if $x_i = b$.
- Let $t : \mathbb{N} \to \mathbb{N}$ be a resource bound. $\mathrm{K}^t_U(x)$ is the minimum value of $|d|$ such that for every $1 \le i \le |x| + 1$ and $b \in \{0, 1, \star\}$, $U^d(i, b)$ accepts in $t(|x|)$ steps if and only if $x_i = b$.

Definition 3.4. Let U_a be an alternating Turing machine, and U_s be a (space-bounded) Turing machine. Let x be a string and artificially let $x_{|x|+1} = \star$.

- Let $t : \mathbb{N} \to \mathbb{N}$ be a resource bound. NC^1-$\mathrm{K}^t_{U_a}(x)$ is the minimum value of $|d|$ such that for every $1 \le i \le |x| + 1$ and $b \in \{0, 1, \star\}$, $U^d_a(i, b)$ accepts in alternating time $\log t(|x|)$ if and only if $x_i = b$.
- Let $t : \mathbb{N} \to \mathbb{N}$ be a resource bound. L-$\mathrm{K}^t_{U_s}(x)$ is the minimum value of $|d|$ such that for every $1 \le i \le |x| + 1$ and $b \in \{0, 1, \star\}$, $U^d_s(i, b)$ accepts in space $\log t(|x|)$ if and only if $x_i = b$.

Fix the number of tapes $k \ge 2$, then there is a universal k-tape Turing machine U that simulates every k-tape Turing machine with only a constant factor blow-up in both time complexity and space complexity [32]. Our results hold for this efficient universal Turing machine, therefore in what follows, we drop the subscript U and simply write KT, K^t, etc.

We also define the *circuit complexity* of a truth table:

Definition 3.5. Let $N = 2^n$, $tt \in \{0, 1\}^N$ be the truth table that corresponds to a function $f : \{0, 1\}^n \to \{0, 1\}$. We define $\mathsf{Size}(tt)$ as the size (number of gates) of the smallest circuit that computes f.

Given a complexity measure μ, the Minimum μ Problem is the language $\{(x, 1^k) : \mu(x) \le k\}$. In particular:

Definition 3.6. We define the following problems:

- (Minimum KT Problem) MKTP := $\{(x, 1^k) : \mathrm{KT}(x) \le k\}$.
- (Minimum Time-Bounded Kolmogorov Complexity Problem) MINKT := $\{(x, 1^t, 1^s) : \mathrm{K}^t(x) \le s\}$.
- (Minimum Circuit Size Problem) MCSP := $\{(tt, 1^s) : \mathsf{Size}(tt) \le s\}$.

There are natural search versions for the problems above. The search version for MKTP is to find a description d such that x can be generated from d implicitly in time at most $k - |d|$. The search version for MINKT is to find a description d of size at most s for x such that x can be generated from d in time at most t. The search version for MCSP is to find a circuit of size at most s for the Boolean function whose truth table is tt.

We need a "trivial" upper bound on these complexity measures. We only state the upper bound for KT complexity.

Fact 3.7 ([4, Proposition 13])**.** *There is an absolute constant $c' > 0$ such that $\mathrm{KT}(x) \leq |x| + c' \log |x|$ for every string x.*

We need the fact that most strings have large Kolmogorov complexity.

Fact 3.8. *Let n be an integer, $s \leq n - 1$, then*

$$\Pr_{\mathbf{x} \leftarrow \mathcal{U}_n}[\mathrm{K}(\mathbf{x}) \leq s] \leq 2^{-(n-s-1)}.$$

Proof Sketch. The number of strings x such that $\mathrm{K}(x) \leq s$ is at most $\sum_{i=0}^{s} 2^i = 2^{s+1} - 1$. □

3.3 Basic Information Theory

We also need some basic concepts in information theory. The *Shannon entropy* of a random variable X, denoted as $\mathrm{H}(X)$, is defined as

$$\mathrm{H}(X) := \mathop{\mathbb{E}}_{\mathbf{x} \leftarrow X}[-\log \Pr[X = \mathbf{x}]].$$

The *min-entropy* of a random variable X, denoted as $\mathrm{H}_\infty(X)$, is the largest real number k such that for every x in the support of X,

$$\Pr[X = x] \leq 2^{-k}.$$

Let X, Y be two random variables defined over a set $\mathcal{S}$. The *statistical distance* between X and Y, denoted as $\mathsf{SD}(X, Y)$, is defined as

$$\mathsf{SD}(X, Y) := \frac{1}{2} \sum_{s \in \mathcal{S}} |\Pr[X = s] - \Pr[Y = s]|.$$

An equivalent definition is as follows: $\mathsf{SD}(X, Y)$ is the maximum value of ε such that there is a (possibly unbounded) distinguisher $\mathcal{D}$ that ε-distinguishes X from Y:

$$\mathsf{SD}(X, Y) := \max_{\mathcal{D}: \mathcal{S} \to \{0,1\}} |\Pr[\mathcal{D}(X) = 1] - \Pr[\mathcal{D}(Y) = 1]|.$$

3.4 Bounded-Error Average-Case Hardness

We define the (bounded-error) average-case hardness of a function f. (Think of $f = \mathrm{KT}$ or K^t.) In the cryptographic setting, we require that any algorithm with an *arbitrary polynomial* run time fails to solve a *fixed-polynomial* fraction of inputs.

Definition 3.9. Let $f : \{0,1\}^\star \to \mathbb{N}$ be a function.

- We say that f is *(bounded-error) hard on average* if the following is true. There is a constant $c > 0$ such that for every PPT[15] machine $\mathcal{A}$ and every large enough input length n,

$$\Pr_{\mathbf{x} \leftarrow \mathcal{U}_n}[\mathcal{A}(\mathbf{x}) = f(\mathbf{x})] \leq 1 - \frac{1}{n^c}.$$

[15] PPT stands for probabilistic polynomial-time.

- Let d be a constant. We say that f is (bounded-error) hard on average to $(d \log n)$-*approximate* if the following is true. There is a constant $c > 0$ such that for every PPT machine $\mathcal{A}$ and every large enough input length n,

$$\Pr_{\mathbf{x} \leftarrow \mathcal{U}_n} \left[f(\mathbf{x}) \leq \mathcal{A}(\mathbf{x}) \leq f(\mathbf{x}) + d \log n \right] \leq 1 - \frac{1}{n^c}.$$

3.5 One-Way Functions

We recall the standard definition of one-way functions and weak one-way functions. Note that by a standard padding trick (see e.g., [34]), every (weak or strong) one-way function can be made *length-preserving*, i.e. for every input $x \in \{0,1\}^\star$, $|f(x)| = |x|$. In this paper, we will implicitly assume that *every candidate one-way function we consider is length-preserving*.

Definition 3.10 (One-Way Functions)**.** Let $f : \{0,1\}^\star \to \{0,1\}^\star$ be a polynomial-time computable function. We say f is a *one-way function* if for every PPT adversary $\mathcal{A}$, it inverts a random output of f with negligible probability. That is, for every $n \in \mathbb{N}$,

$$\Pr_{\mathbf{x} \leftarrow \mathcal{U}_n} [\mathcal{A}(f(\mathbf{x})) \in f^{-1}(f(\mathbf{x}))] \leq \mathrm{negl}(n).$$

One-way functions are also called *strong* one-way functions, as no PPT adversary could invert it non-trivially. We also consider *weak* one-way functions, where no PPT adversary could invert it on a $1 - 1/p(n)$ fraction of inputs for some *fixed* polynomial $p(\cdot)$.

Definition 3.11 (Weak One-Way Functions)**.** Let $f : \{0,1\}^\star \to \{0,1\}^\star$ be a polynomial-time computable function. We say f is a *weak one-way function* if there is a polynomial $p(\cdot)$ such that the following holds. For every PPT adversary $\mathcal{A}$, it inverts a random output of f with probability at most $1 - 1/p(n)$. That is, for every $n \in \mathbb{N}$,

$$\Pr_{\mathbf{x} \leftarrow \mathcal{U}_n} [\mathcal{A}(f(\mathbf{x})) \in f^{-1}(f(\mathbf{x}))] \leq 1 - \frac{1}{p(n)}.$$

Yao showed that every weak one-way function can be *amplified* into a strong one-way function.

Theorem 3.12 ([102, 34])**.** *If there exists a weak one-way function, then there exists a strong one-way function.*

In particular, let f be a weak one-way function. Then there is a polynomial $k(\cdot)$ such that the following function f^k is a strong one-way function.

$$f^k(x_1, x_2, \ldots, x_{k(n)}) = f(x_1) \circ f(x_2) \circ \cdots \circ f(x_{k(n)}),$$

where $x_1, x_2, \ldots, x_{k(n)}$ are length-n inputs.

3.6 Conditionally Secure Entropy-Preserving PRGs

Here we define conditionally secure entropy-preserving PRGs (condEP-PRGs), introduced in [70].

A *pseudorandom generator*, according to the standard definition, is a polynomial-time computable function $G : \{0,1\}^n \to \{0,1\}^m$ (where $m > n$) such that $G(\mathcal{U}_n)$ and $\mathcal{U}_m$ are computationally indistinguishable. Compared with standard PRGs, a *condEP-PRG* $G : \{0,1\}^n \to \{0,1\}^m$ has three differences:

- The input distribution of G is not $\mathcal{U}_n$. Instead, it is the uniform distribution over a subset of inputs $\mathcal{E}_n$, called the *condition*. (We will use $\mathcal{E}_n$ to denote both the subset and the uniform distribution over this subset.)
- G is *entropy-preserving*, meaning that $G(\mathcal{E}_n)$ has large (information-theoretic) entropy. (Note that $\log|\mathcal{E}_n| \le n \le m$. As a consequence, $\log|\mathcal{E}_n|$ cannot be too small compared to m.)
- Finally, G only $(1/p(n))$-fools PPT adversaries for a fixed polynomial $p(\cdot)$. For comparison, a standard PRG is required to $(1/p(n))$-fool PPT adversaries *for every polynomial* $p(\cdot)$. This difference is mostly technical.

Definition 3.13 (Conditionally Secure Entropy-Preserving PRG, abbr. condEP-PRG, [70])**.** Let $\gamma > 0$ be a constant and $p(\cdot)$ be a polynomial. Consider a polynomial-time computable ensemble of functions $G = \{G_n : \{0,1\}^n \to \{0,1\}^{n+\gamma\log n}\}$. We say G is a *condEP-PRG*, if there is a family of subsets $\mathcal{E} = \{\mathcal{E}_n \subseteq \{0,1\}^n\}$ (called the "events" or "conditions") such that the following are true.

(Pseudorandomness) $G_n(\mathcal{E}_n)$ is $(1/p(n))$-indistinguishable from $\mathcal{U}_{n+\gamma\log n}$ by PPT adversaries. That is, for every PPT $\mathcal{A}$ and every integer n,

$$\left|\Pr_{\mathbf{x}\leftarrow\mathcal{U}_{n+\gamma\log n}}[\mathcal{A}(\mathbf{x})=1] - \Pr_{\mathbf{x}\leftarrow G_n(\mathcal{E}_n)}[\mathcal{A}(\mathbf{x})=1]\right| < 1/p(n).$$

(Entropy-Preservation) There is a constant d such that for every large enough n, $\mathrm{H}(G_n(\mathcal{E}_n)) \ge n - d\log n$.

We say the *stretch* of G is $\gamma\log n$ and the *security* of G is $1/p(n)$.

Theorem 3.14 ([70])**.** *There is a function* EP-PRG *computable in* ALOGTIME *such that the following holds. For any one-way function $f : \{0,1\}^\star \to \{0,1\}^\star$ and any constant $\gamma > 0$, let $G(x,z) =$ EP-PRG$(\gamma, x, f(x), z)$, then G is a condEP-PRG with stretch $\gamma\log n$ and security $1/n^\gamma$.*

Remark 3.15. It is important that the machine EP-PRG is fixed and does not depend on the constant γ. Suppose there is an absolute constant $c > 0$ such that for every $\gamma > 0$, there is a PRG G_γ that runs in DTIME$[c\log n]$ and stretches n bits into $n + \gamma\log n$ bits. The outputs of G_γ will always have KT complexity at most $n + c\log n + O(1) < n + \gamma\log n$, hence a heuristic for MKTP can always distinguish the outputs of G_γ from truly random strings. It follows that we can use such G_γ to argue about the hardness of MKTP. On the other hand, if the time complexity of G_γ depends on γ, it does not necessarily imply any hardness of MKTP.

3.7 Complete Problems for ⊕L

We introduce the ⊕L-complete problems, called Connected Matrix Determinant (CMD) and Decomposed Connected Matrix Determinant (DCMD), which will be crucial to us. Originally motivated by secure multi-party computation [61, 62], these problems have found surprisingly many applications in cryptography and complexity theory [11, 37, 29, 48, 26].

Let n be any integer, define $\ell_{\mathrm{CMD}}(n) := n(n+1)/2$ and $\ell_{\mathrm{DCMD}}(n) := n^3(n+1)/2$.

Definition 3.16 (See e.g., [26])**.** An instance of CMD is an $n \times n$ matrix over GF(2) where the main diagonal and above may contain either 0 or 1, the second diagonal (i.e., the one below the main diagonal) contains 1, and other entries are 0. In other words, the matrix is of the following form (where $*$ represents any element in GF(2)):

$$\begin{pmatrix} * & * & * & \cdots & * & * \\ 1 & * & * & \cdots & * & * \\ 0 & 1 & * & \cdots & * & * \\ 0 & 0 & 1 & \cdots & * & * \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 & * \end{pmatrix}.$$

The instance is an $(n(n+1)/2)$-bit string specifying elements on and above the main diagonal. We define $x \in \mathrm{CMD}$ if and only if the determinant (over GF(2)) of the matrix corresponding to x is 1.

An instance of DCMD is a string of length $n^3(n+1)/2$. For an input x, $\mathrm{DCMD}(x)$ is computed as follows: we partition x into blocks of length n^2, let $y_i (1 \le i \le n(n+1)/2)$ be the parity of the i-th block, and define $\mathrm{DCMD}(x) := \mathrm{CMD}(y_1 \circ y_2 \circ \cdots \circ y_{n(n+1)/2})$.

In fact, the precise definitions of CMD and DCMD are not important here, but we need the following important facts about them.

Theorem 3.17 ([11])**.** *Let n be an integer. There is a function $P_{\mathrm{CMD}} : \{0,1\}^{\ell_{\mathrm{CMD}}(n)} \times \{0,1\}^{\ell_{\mathrm{DCMD}}(n)-1} \to \{0,1\}^{\ell_{\mathrm{DCMD}}(n)}$ computable in* DLOGTIME *such that the following holds. For any input $x \in \{0,1\}^{\ell_{\mathrm{CMD}}(n)}$, the distribution of $P_{\mathrm{CMD}}(x, \mathcal{U}_{\ell_{\mathrm{DCMD}}(n)-1})$ is equal to the uniform distribution over $\{y \in \{0,1\}^{\ell_{\mathrm{DCMD}}(n)} : \mathrm{DCMD}(y) = \mathrm{CMD}(x)\}$.*

Note that P_{CMD} only uses $\ell_{\mathrm{DCMD}}(n) - 1$ random bits, which is optimal. It also implies:

Corollary 3.18. DCMD *is* balanced. *In other words, for every integer n, the number of Yes instances and No instances of* DCMD *on input length $\ell_{\mathrm{DCMD}}(n)$ are the same.*

Proof. Fix any Yes instance $x \in \{0,1\}^n$ of CMD, then $\{P_{\mathrm{CMD}}(x,r) : r \in \{0,1\}^{\ell_{\mathrm{DCMD}}(n)-1}\}$ contains every Yes instance of DCMD. It follows that there are at most $2^{\ell_{\mathrm{DCMD}}(n)-1}$ Yes instances of DCMD on input length $\ell_{\mathrm{DCMD}}(n)$. The same upper bound can also be obtained for No instances. Since there are $2^{\ell_{\mathrm{DCMD}}(n)}$ strings of length $\ell_{\mathrm{DCMD}}(n)$, there must be exactly $2^{\ell_{\mathrm{DCMD}}(n)-1}$ Yes instances and exactly $2^{\ell_{\mathrm{DCMD}}(n)-1}$ No instances of length $\ell_{\mathrm{DCMD}}(n)$. □

Theorem 3.19 ([61, 62])**.** CMD *is* $\oplus\mathsf{L}$*-complete under projections.*[16]

In other words, a language L *is in* $\oplus\mathsf{L}$ *if and only if there is a polynomial* $t(\cdot)$ *and a* DLOGTIME*-computable projection* $p : \{0,1\}^n \to \{0,1\}^{\ell_{\mathrm{CMD}}(t(n))}$ *such that for every input* $x \in \{0,1\}^n$, $x \in L$ *if and only if* $\mathrm{CMD}(p(x)) = 1$.

Remark 3.20. A proof of Theorem 3.19 can be found in [26, Section B.1]. However, the proof in [26] does not show that the projections are DLOGTIME-uniform. In particular, the reduction needs to calculate the topological order of the underlying (parity) branching program ($\sigma_1, \sigma_2, \ldots, \sigma_m$ in [26, Section B.1]), which may not be computable in DLOGTIME.

We can fix this issue by adding a clock to the log-space Turing machine; a state of the Turing machine appears earlier in the topological order if its clock value is smaller. Equivalently, let $G = (V, E)$ be the old branching program. The new branching program G^{new} has a vertex (i, v) for every $0 \le i \le |V|$ and $v \in V$, and has edges from (i, u) to $(i+1, v)$ for every edge $(u, v) \in G$ and every $0 \le i < |V|$. Let $V = \{v_1, \ldots, v_n\}$, then

$$(0, v_1), \ldots, (0, v_n), (1, v_1), \ldots, (1, v_n), \ldots, (|V|, v_1), \ldots, (|V|, v_n)$$

is a valid topological ordering of G^{new}. Now we can use [26, Section B.1] to reduce the computation of G^{new} to CMD by a DLOGTIME-uniform projection.

We thank Yanyi Liu for pointing out this issue.

Theorem 3.17 and 3.19 implies the following beautiful result in [11].

Theorem 3.21 ([11])**.** *Suppose there is a one-way function computable in* $\oplus\mathsf{L}$*. Then there is a one-way function computable in* DLOGTIME*.*

Proof Sketch. Let f be a one-way function in $\oplus\mathsf{L}$. There is a DLOGTIME-computable function $p(\cdot, i)$ that maps n input bits to $\mathrm{poly}(n)$ output bits such that for every integer i and every string x, the i-th output bit of $f(x)$ is $\mathrm{CMD}(p(x, i))$. Consider the following function:

$$g(x, y, i) := P_{\mathrm{CMD}}(p(x, i), y).$$

It turns out that the function $g(x, y_1, \ldots, y_n) = g(x, y_1, 1) \circ \cdots \circ g(x, y_n, n)$ is still one-way. □

4 KT Complexity and Parallel Cryptography

In this section, we characterize the existence of one-way functions in DLOGTIME by the average-case hardness of MKTP. Recall that the seminal work of [11] showed that the existence of one-way functions in DLOGTIME is also equivalent to the existence of one-way functions in uniform NC^1, L, or $\oplus\mathsf{L}$.

Theorem 1.1 (Main Result; Informal)**.** *There is a one-way function computable in uniform* NC^1 *if and only if* KT *is bounded-error hard on average.*

[16] A *projection* is a (multi-output) function where each output bit either is a constant or only depends on one input bit.

4.1 One-Way Functions in NC^0 from Hardness of MKTP

Theorem 4.1. *Suppose that the search version of* KT *is bounded-error hard on average. Then there is a one-way function computable in* DLOGTIME.

Proof. We show that there is a weak one-way function computable in logarithmic space. Then by Theorem 3.12, there is a one-way function in logarithmic space, and by Theorem 3.21, there is a one-way function in DLOGTIME.

Suppose KT is bounded-error hard on average. By Definition 3.9, there is a constant $c > 0$ such that for every PPT algorithm $\mathcal{A}$ and every large enough n, the probability that $\mathcal{A}$ solves the search version of KT on a random length-n input is at most $1 - 1/n^c$.

For a string x, we define $t(x)$ to be the parameter t in the definition of $\mathrm{KT}(x)$ (Definition 3.3). Formally, $t(x)$ is the smallest integer t such that there is a description d of length $\mathrm{KT}(x) - t$ such that for every $1 \le i \le |x|$ and $b \in \{0, 1, \star\}$, $U^d(i, b)$ accepts in t steps if and only if $x_i = b$. We can see that most strings have small $t(x)$. In what follows, let c_1 be the absolute constant in Fact 3.7 such that for every $x \in \{0,1\}^n$, $\mathrm{KT}(x) \le |x| + c_1 \log |x|$.

Claim 4.2. *For all but an $1/n^{c+1}$ fraction of strings $x \in \{0,1\}^n$, we have $t(x) \le (c + c_1 + 2) \log n$.*

Proof of Claim 4.2. By Fact 3.7, for every $x \in \{0,1\}^n$, $\mathrm{KT}(x) \le n + c_1 \log n$. By Fact 3.8, all but a $1/n^{c+1}$ fraction of strings $x \in \{0,1\}^n$ satisfies that $\mathrm{K}(x) > n - (c+1)\log n - 1$. For such strings x, we have $t(x) \le \mathrm{KT}(x) - \mathrm{K}(x) \le (c + c_1 + 2) \log n$. ◇

For convenience, we say a pair (d, t) *outputs* the string x, if (d, t) is a valid "witness" for $\mathrm{KT}(x)$, i.e. for every $1 \le i \le |x| + 1$ and $b \in \{0, 1, \star\}$, $U^d(i, b)$ accepts in time t if and only if $x_i = b$. Let $\mathsf{Output}(d, t)$ be the (unique) string that (d, t) outputs; if (d, t) does not output any (finite) string, let $\mathsf{Output}(d, t) = \perp$.

We define a weak one-way function f as in Algorithm 1.

Algorithm 1 Weak OWF in L from Average-Case Hardness of MKTP

```
function f(ℓ, t, M)
    The input consists of integers ℓ ∈ [n + c_1 log n], t ∈ [(c + c_1 + 2) log n], and a string M ∈ {0,1}^{n + c_1 log n}.
    M' ← the first ℓ bits of M
    out ← Output(M', t)
    if |out| = n then
        return the concatenation of ℓ, t, and out
    else
        return ⊥
    end if
end function
```

Since $t \le O(\log n)$, we can always compute $\mathsf{Output}(M', t)$ in logarithmic space. It follows that f is computable in logarithmic space.

Let $\mathcal{D}_{\mathsf{owf}}$ be the output distribution of f on uniform inputs. In other words, to sample from $\mathcal{D}_{\mathsf{owf}}$, we sample two integers $\ell \leftarrow [n + c_1 \log n]$, $t \leftarrow [(c + c_1 + 2) \log n]$ and a string M of length $n + c_1 \log n$, and output $f(\ell, t, M)$. We prove that $\mathcal{D}_{\mathsf{owf}}$ *almost dominates* the uniform distribution over $\{0,1\}^n$, in the following sense.

Claim 4.3. *Let n be a large enough integer. For every string x such that $t(x) \leq (c + c_1 + 2) \log n$, the probability that a random sample from $\mathcal{D}_{\mathsf{owf}}$ is equal to $(\mathrm{KT}(x) - t(x), t(x), x)$ is at least $\frac{1}{2^n n^{2+c_1}}$.*

Proof of Claim 4.3. For a large enough n, with probability at least $\frac{1}{n^2}$, the sampler for $\mathcal{D}_{\mathsf{owf}}$ samples $t = t(x)$ and $\ell = \mathrm{KT}(x) - t(x)$. Then, with probability $\frac{1}{2^\ell} \geq \frac{1}{2^n n^{c_1}}$, the sampler samples a description M' such that $\mathsf{Output}(M', t) = x$. It follows that w.p. at least $\frac{1}{2^n n^{2+c_1}}$ the sampler outputs $(\mathrm{KT}(x) - t(x), t(x), x)$. ◇

Now, we can prove the security of the weak OWF f. Let $\mathcal{A}_{\mathsf{owf}}$ be a candidate PPT adversary trying to invert f. We construct a polynomial-time algorithm $\mathcal{A}_{\mathsf{KT}}$ that attempts to solve (the search version of) MKTP in Algorithm 2.

Algorithm 2 Bounded-Error Heuristic $\mathcal{A}_{\mathsf{KT}}$ for MKTP from Inverter $\mathcal{A}_{\mathsf{owf}}$ for f

```
function A_KT(x)
    n ← |x|; Opt ← +∞; Witness ← ⊥
    for ℓ ∈ [n + c_1 log n] and t ∈ [(c + c_1) log n] do
        (ℓ', t', M) ← A_owf(ℓ, t, x)
        M' ← the first ℓ' bits of M
        if Output(M', t') = x and Opt > |M'| + t' then
            Opt ← |M'| + t'
            Witness ← (M', t')
        end if
    end for
    return (Opt, Witness)
end function
```

Note that for a fixed input x, $\mathcal{A}_{\mathsf{KT}}(x)$ fails to output a valid witness for $\mathrm{KT}(x)$ only if $\mathcal{A}_{\mathsf{owf}}$ fails to invert the output $(\mathrm{KT}(x) - t(x), t(x), x)$. Let $p_{\mathsf{fail}}(x)$ be the probability (over the internal randomness of $\mathcal{A}_{\mathsf{owf}}$) that $\mathcal{A}_{\mathsf{owf}}$ fails to invert $(\mathrm{KT}(x) - t(x), t(x), x)$, then we have

$$\mathop{\mathbb{E}}_{\mathbf{x} \leftarrow \mathcal{U}_n} [p_{\mathsf{fail}}(\mathbf{x})] \geq \Pr_{\mathbf{x} \leftarrow \mathcal{U}_n} [\mathcal{A}_{\mathsf{KT}}(\mathbf{x}) \text{ fails on input } \mathbf{x}] \geq \frac{1}{n^c}. \tag{4.1}$$

Let p be the probability that $\mathcal{A}_{\mathsf{owf}}$ fails to invert a random input of f. Then

$$
\begin{aligned}
p &\geq \sum_{\substack{x\in\{0,1\}^n \\ t(x)\leq(c+c_1+2)\log n}} \Pr_{\mathbf{y}\leftarrow\mathcal{D}_{\mathsf{owf}}}[\mathbf{y} = (\mathrm{KT}(x)-t(x), t(x), x)] \cdot p_{\mathsf{fail}}(x) \\
&\geq \sum_{\substack{x\in\{0,1\}^n \\ t(x)\leq(c+c_1+2)\log n}} \frac{1}{2^n n^{2+c_1}} \cdot p_{\mathsf{fail}}(x) && \text{(Claim 4.3)} \\
&\geq \frac{1}{2^n n^{2+c_1}}\left(\left(\sum_{x\in\{0,1\}^n} p_{\mathsf{fail}}(x)\right) - \frac{2^n}{n^{c+1}}\right) && \text{(Claim 4.2)} \\
&\geq \frac{1}{n^{2+c_1}}\left(\mathop{\mathbb{E}}_{\mathbf{x}\leftarrow\mathcal{U}_n}[p_{\mathsf{fail}}(\mathbf{x})]\right) - \frac{1}{n^{c+c_1+3}} \\
&\geq \frac{1}{n^{2+c_1+c}} - \frac{1}{n^{c+c_1+3}}. && \text{By (4.1)}
\end{aligned}
$$

Let $c' = c + c_1 + 4$, then every PPT adversary $\mathcal{A}_{\mathsf{owf}}$ fails to invert a random input of f w.p. at least $\frac{1}{n^{c'}}$. It follows that f is a weak OWF. □

4.2 Hardness of MKTP from One-Way Functions in ⊕L

In this section, we prove the following theorem.

Theorem 4.4. *Suppose there is a one-way function computable in* ⊕L*. Then for every constant* $\lambda > 0$*,* KT *is bounded-error hard on average to approximate within an additive factor of* $\lambda \log n$*.*

Let f be a one-way function in ⊕L. The proof consists of three steps:

- First, we use f to build a condEP-PRG G. If f is computable in ⊕L, then G is also computable in ⊕L. This step is the same as in [70] and follows directly from Theorem 3.14.
- Second, we construct the randomized encoding $\tilde{G}$ of G. We argue that $\tilde{G}$ is also a condEP-PRG. Moreover, $\tilde{G}$ is computable in DLOGTIME. This step is implemented in Lemma 4.5.
- Last, as every output of $\tilde{G}$ has small KT complexity, we use the security of $\tilde{G}$ to show that MKTP is bounded-error hard on average. This step is implemented in Lemma 4.7.

4.2.1 CondEP-PRG in DLOGTIME

In this section, we prove the following lemma that constructs a DLOGTIME-computable condEP-PRG from a ⊕L-computable condEP-PRG.

Lemma 4.5. *Suppose there is a constant* $c > 0$ *such that for every constant* $\lambda > 0$*, there is a condEP-PRG* G *with stretch* $\lambda \log n$ *and security* $1/n^\lambda$ *that is computable in* $\oplus\mathsf{SPACE}[c \log n]$*.*

Then there is a constant $c' > 0$ *such that for every constant* $\lambda > 0$*, there is a condEP-PRG* $\tilde{G}$ *with stretch* $\lambda \log n$ *and security* $1/n^\lambda$ *that is computable in* $\mathsf{DTIME}[c' \log n]$*.*

Proof. Fix an input length n. Let λ' be a constant that depends on λ, we will fix λ' later. Let G be a condEP-PRG with stretch $\lambda' \log n$ and security $1/n^{\lambda'}$ that is computable in $\oplus\mathsf{SPACE}[c \log n]$. We denote the n-th slice of G as $G_n : \{0,1\}^n \to \{0,1\}^\ell$, where $\ell := n + \lambda' \log n$.

Let $N := n^c$. Since $G \in \oplus\mathsf{SPACE}[\log N]$, there are projections

$$G_1^{\mathsf{proj}}, G_2^{\mathsf{proj}}, \ldots, G_\ell^{\mathsf{proj}} : \{0,1\}^n \to \{0,1\}^{\ell_{\mathrm{CMD}}(N)},$$

such that the i-th output bit of $G(x)$ is equal to $\mathrm{CMD}(G_i^{\mathsf{proj}}(x))$. Let $r_1, r_2, \ldots, r_\ell$ be random strings of length $\ell_{\mathrm{DCMD}}(N) - 1$. Let P_{CMD} be the $\mathsf{DLOGTIME}$-computable function defined in Theorem 3.17, then the i-th output bit of $G(x)$ is equal to $\mathrm{DCMD}(P_{\mathrm{CMD}}(G_i^{\mathsf{proj}}(x), r_i))$. We define

$$\tilde{G}(x, r_1, \ldots, r_\ell) = P_{\mathrm{CMD}}(G_1^{\mathsf{proj}}(x), r_1) \circ P_{\mathrm{CMD}}(G_2^{\mathsf{proj}}(x), r_2) \circ \cdots \circ P_{\mathrm{CMD}}(G_\ell^{\mathsf{proj}}(x), r_\ell).$$

($\tilde{G}$ is the "randomized encoding" of G in the sense of [11].)

It is easy to see that there is a constant c_g depending only on c such that $\tilde{G} \in \mathsf{DTIME}[c_g \log n]$. Note that the input length of $\tilde{G}$ is $n_{\mathsf{in}} := n + \ell \cdot (\ell_{\mathrm{DCMD}}(N) - 1)$, the output length of $\tilde{G}$ is $n_{\mathsf{out}} := \ell \cdot \ell_{\mathrm{DCMD}}(N)$, and $n_{\mathsf{out}} = n_{\mathsf{in}} + (\ell - n) = n_{\mathsf{in}} + \lambda' \log n$. Here, we fix λ' large enough such that $\lambda' \geq \lambda \frac{\log n_{\mathsf{in}}}{\log n}$.

Claim 4.6. *$\tilde{G}$ is a condEP-PRG with stretch $\lambda \log n_{\mathsf{in}}$ and security $1/(n_{\mathsf{in}})^\lambda$.*

Proof. Clearly, the stretch of $\tilde{G}$ is $\lambda' \log n \geq \lambda \log n_{\mathsf{in}}$.

Suppose $\mathcal{E} = \{\mathcal{E}_n \subseteq \{0,1\}^n\}$ is a sequence of events such that G and $\mathcal{E}$ satisfy Definition 3.13. Let $\tilde{\mathcal{E}} := \{\tilde{\mathcal{E}}_{n_{\mathsf{in}}}\}$ where $\tilde{\mathcal{E}}_{n_{\mathsf{in}}} := \mathcal{E}_n \times \{0,1\}^{\ell \cdot (\ell_{\mathrm{DCMD}}(N) - 1)}$. We verify that $\tilde{G}$ and $\tilde{\mathcal{E}}$ satisfy Definition 3.13.

Pseudorandomness. Suppose, for the sake of contradiction, that there is a PPT adversary $\mathcal{A}'$ such that

$$\Pr[\mathcal{A}'(\tilde{G}(\tilde{\mathcal{E}}_{n_{\mathsf{in}}}))] - \Pr[\mathcal{A}'(\mathcal{U}_{n_{\mathsf{out}}})] \geq 1/(n_{\mathsf{in}})^\lambda.$$

Consider an adversary $\mathcal{A}$ that distinguishes $G(\mathcal{E}_n)$ from $\mathcal{U}_\ell$ as follows. On input y, for every $1 \leq i \leq \ell$, we sample $\mathbf{r}_i \leftarrow P_{\mathrm{CMD}}(y_i, \mathcal{U}_{\ell_{\mathrm{DCMD}}(N)-1})$, then $\mathbf{r}_i$ is a uniformly random length-$\ell_{\mathrm{DCMD}}(N)$ input of DCMD such that $\mathrm{DCMD}(\mathbf{r}_i) = y_i$. We concatenate them as $\mathbf{r} = \mathbf{r}_1 \circ \mathbf{r}_2 \circ \cdots \circ \mathbf{r}_\ell$ and let $\mathcal{A}(y) = \mathcal{A}'(\mathbf{r})$.

Suppose $\mathbf{y} \leftarrow G(\mathcal{E}_n)$, then the distribution of $\mathbf{r}$ is exactly $\tilde{G}(\tilde{\mathcal{E}}_{n_{\mathsf{in}}})$. On the other hand, suppose $\mathbf{y} \leftarrow \mathcal{U}_\ell$, then the distribution of $\mathbf{r}$ is exactly $\mathcal{U}_{n_{\mathsf{out}}}$. As $\mathcal{A}'$ distinguishes $\tilde{G}(\tilde{\mathcal{E}}_{n_{\mathsf{in}}})$ from $\mathcal{U}_{n_{\mathsf{out}}}$ with advantage $\geq 1/(n_{\mathsf{in}})^\lambda$, we can see that $\mathcal{A}$ also distinguishes $G(\mathcal{E}_n)$ from $\mathcal{U}_\ell$ with advantage $\geq 1/(n_{\mathsf{in}})^\lambda \geq 1/n^{\lambda'}$, contradicting the security of G.

Entropy-preservation. Consider the above experiment, where we first sample $\mathbf{y} \leftarrow G(\mathcal{E}_n)$, then sample a uniform string $\mathbf{r}_i$ of length $\ell_{\mathrm{DCMD}}(N)$ such that $\mathrm{DCMD}(\mathbf{r}_i) = \mathbf{y}_i$ for every $1 \leq i \leq \ell$,

and finally concatenate them as $\mathbf{r} = \mathbf{r}_1 \circ \mathbf{r}_2 \circ \cdots \circ \mathbf{r}_\ell$. The distribution of $\mathbf{r}$ is exactly $\tilde{G}(\tilde{\mathcal{E}}_{n_{\mathsf{in}}})$. Therefore,

$$\begin{aligned} \mathrm{H}(\tilde{G}(\tilde{\mathcal{E}}_{n_{\mathsf{in}}})) &= \mathrm{H}(G(\mathcal{E}_n)) + \ell \cdot (\ell_{\mathrm{DCMD}}(N) - 1) \\ &\geq n - \Omega(\log n) + \ell \cdot (\ell_{\mathrm{DCMD}}(N) - 1) \\ &\geq n_{\mathsf{in}} - \Omega(\log n_{\mathsf{in}}). \end{aligned}$$ ◇

We have only defined $\tilde{G}$ and $\tilde{\mathcal{E}}$ on input lengths of the form $n_{\mathsf{in}}(n) = n+(n+\lambda' \log n)(\ell_{\mathrm{DCMD}}(n^c)+1)$. However, it is straightforward to define $\tilde{G}$ and $\tilde{\mathcal{E}}$ on every input length. Let m be an input length, $m' = n_{\mathsf{in}}(n)$ be the largest number of the form $n_{\mathsf{in}}(n)$ such that $m' \leq m$. On input $x \in \{0,1\}^m$, let x_1 be the length-m' prefix of x and x_2 be the rest of x (i.e., $x = x_1 \circ x_2$), and we can define $\tilde{G}(x) = \tilde{G}(x_1) \circ x_2$. Similarly, we could define $\tilde{\mathcal{E}}_m = \tilde{\mathcal{E}}_{m'} \times \{0,1\}^{m-m'}$. □

4.2.2 Hardness of MKTP

Lemma 4.7. *Suppose there is a constant $c > 0$ such that for every constant $\lambda > 0$, there is a condEP-PRG G with stretch $\lambda \log n$ and security $1/n^\lambda$ that is computable in $\mathsf{DTIME}[c \log n]$.*

Then for every constant $\lambda > 0$, KT is bounded-error hard on average to approximate within an additive error of $\lambda \log n$.

Proof. Let $\lambda' := \lambda + c_1 + 2$ for a constant c_1 defined later, and G be a condEP-PRG with stretch $\lambda' \log n$ and security $1/n^{\lambda'}$ that is computable in $\mathsf{DTIME}[c \log n]$. Fix an input length n, and let $\ell := n + \lambda' \log n$.

We note that the KT complexity of every output of G is nontrivial. Let c_1 be a large enough constant that only depends on c. Since $G \in \mathsf{DTIME}[c \log n]$, there is a description d of constant length such that the following holds: For every input $x \in \{0,1\}^n$, every $1 \leq i \leq \ell + 1$, and every $b \in \{0,1,\star\}$, $U^{d,x}(i,b)$ accepts in $(c_1 - 1)\log n$ time if and only if the i-th bit of $G(x)$ is equal to b. It follows that for every $x \in \{0,1\}^n$,

$$\mathrm{KT}(G(x)) \leq n + (c_1 - 1)\log n + O(1) < \ell - (\lambda' - c_1)\log n.$$

Suppose, for the sake of contradiction, that KT is bounded-error easy on average to approximate within an additive factor of $\lambda \log n$. For a large constant c_{kt} that we fix later, there is a PPT machine $\mathcal{A}$ such that

$$\Pr_{\mathbf{y} \leftarrow \mathcal{U}_\ell}[\mathrm{KT}(\mathbf{y}) \leq \mathcal{A}(\mathbf{y}) \leq \mathrm{KT}(\mathbf{y}) + \lambda \log n] \geq 1 - \frac{1}{n^{c_{\mathrm{kt}}}}. \tag{4.2}$$

It is natural to consider the following adversary $\mathcal{A}'$: On input $y \in \{0,1\}^\ell$, $\mathcal{A}'$ outputs 1 if $\mathcal{A}(y) \geq \ell - 2\log n$, and outputs 0 otherwise. We will prove the following two lemmas, showing that $\mathcal{A}'$ distinguishes $G(\mathcal{E}_n)$ from $\mathcal{U}_\ell$ with good advantage.

Lemma 4.8. $\Pr_{\mathbf{y} \leftarrow \mathcal{U}_\ell}[\mathcal{A}'(\mathbf{y}) = 1] \geq 1 - \frac{1}{n^2} - \frac{1}{n^{c_{\mathrm{kt}}}}$.

Proof. By Fact 3.8, all but a $\frac{1}{n^2}$ fraction of strings $y \in \{0,1\}^\ell$ satisfies that $\mathrm{K}(y) \geq \ell - 2\log n$. Therefore, for all but a $\left(\frac{1}{n^2} + \frac{1}{n^{c_{\mathrm{kt}}}}\right)$ fraction of strings $y \in \{0,1\}^\ell$, we have $\mathcal{A}(y) \geq \mathrm{KT}(y) \geq \mathrm{K}(y) \geq \ell - 2\log n$. On these strings y we have $\mathcal{A}'(y) = 1$. ◇

Lemma 4.9. $\Pr_{\mathbf{y}\leftarrow G(\mathcal{E}_n)}[\mathcal{A}'(\mathbf{y}) = 1] \leq 1 - \frac{1}{n}$.

Proof. Let $H := \mathrm{H}(G(\mathcal{E}_n))$. There is a constant d such that $\ell - H \leq d\log n$. Note that d does not depend on c_{kt}, which means we can set $c_{\mathrm{kt}} := d + 15$.

Consider the set of outputs of G that is outputted with probability at most 2^{1-H}. We say these inputs are *good*. Let Good be the set of good inputs, i.e.,

$$\mathsf{Good} := \left\{y \in \{0,1\}^\ell : 0 < \Pr[G(\mathcal{E}_n) = y] \leq 2^{1-H}\right\}.$$

We can see that there are many good strings. Actually, let $p := \Pr_{\mathbf{y}\leftarrow G(\mathcal{E}_n)}[\mathbf{y} \in \mathsf{Good}]$, then

$$H = \mathrm{H}(G(\mathcal{E}_n)) \leq p \cdot n + (1-p)\cdot(H-1),$$

which implies that $p \geq \frac{1}{n-H+1}$.

Let Err be the subset of Good on which $\mathcal{A}$ fails to produce a good approximation of KT. (In case that $\mathcal{A}$ is a randomized algorithm, it fails w.p. at least $1/n^4$.) That is,

$$\mathsf{Err} := \left\{y \in \mathsf{Good} : \Pr[\mathrm{KT}(y) \leq \mathcal{A}(y) \leq \mathrm{KT}(y) + \lambda\log n] \leq 1 - 1/n^4\right\}.$$

By Eq. (4.2), $|\mathsf{Err}| \leq 2^\ell/n^{c_{\mathrm{kt}}-4}$. Therefore,

$$\Pr_{\mathbf{y}\leftarrow G(\mathcal{E}_n)}[\mathbf{y} \in \mathsf{Err}] \leq (2^\ell/n^{c_{\mathrm{kt}}-4})\cdot 2^{1-H} \leq 2\cdot n^{d+4-c_{\mathrm{kt}}} \leq 1/n^4.$$

Note that for every y in the range of $G(\mathcal{E}_n)$, if $\mathcal{A}$ is correct on y, we have $\mathcal{A}(y) < \ell - (\lambda' - c_1)\log n + \lambda\log n = \ell - 2\log n$. Therefore for every $y \in \mathsf{Good} \setminus \mathsf{Err}$, we have $\mathcal{A}'(y) = 0$ w.p. at least $1 - 1/n^4$ over the internal randomness of $\mathcal{A}'$. It follows that

$$\begin{aligned}
\Pr_{\mathbf{y}\leftarrow G(\mathcal{E}_n)}[\mathcal{A}'(\mathbf{y}) = 1] &\leq (1-p) + \Pr_{\mathbf{y}\leftarrow G(\mathcal{E}_n)}[\mathbf{y} \in \mathsf{Err}] + \frac{1}{n^4} \\
&\leq 1 - \frac{1}{n-H+1} + \frac{1}{n^4} + \frac{1}{n^4} \\
&\leq 1 - \frac{1}{n}.
\end{aligned}$$

◇

From the pseudorandomness of the condEP-PRG G, we conclude that KT is hard on average to approximate within an additive error of $\lambda\log n$.

Note that we have only proved the hardness of MKTP on input lengths of the form $n + \lambda'\log n$, but it is straightforward to extend the argument to every input length m. Let m' be the largest number of the form $m' = n + \lambda'\log n$ such that $m' \leq m$, then $m - m' \leq O(1)$. For every $x \in \{0,1\}^m$, let x_1 be the length-m' prefix of x. There is an absolute constant d such that $\mathrm{KT}(x_1) - d\log m \leq \mathrm{KT}(x) \leq \mathrm{KT}(x_1) + d\log m$. It follows that if we can approximate MKTP on input length m', then we can also approximate MKTP on input length m. □

4.2.3 Proof of Theorem 4.4

Theorem 4.4. *Suppose there is a one-way function computable in* $\oplus\mathsf{L}$. *Then for every constant* $\lambda > 0$, KT *is bounded-error hard on average to approximate within an additive factor of* $\lambda \log n$.

Proof. Let c be a constant such that there is a one-way function f computable in $\oplus\mathsf{SPACE}[c \log n]$. Let $\mathsf{EP\text{-}PRG}$ be the Turing machine guaranteed in Theorem 3.14. For every constant $\lambda > 0$, let $G(x, z) = \mathsf{EP\text{-}PRG}(\lambda, x, f(x), z)$. Then there is a constant c_1 only depending on c (not on λ) such that G is computable in $\oplus\mathsf{SPACE}[c_1 \log n]$. Moreover, G is a condEP-PRG with stretch $\lambda \log n$ and security $1/n^\lambda$.

By Lemma 4.5, there is a constant c_2 only depending on c such that for every constant $\lambda > 0$, there is a condEP-PRG with stretch $\lambda \log n$ and security $1/n^\lambda$ that is computable in $\mathsf{DTIME}[c_2 \log n]$. By Lemma 4.7, for every constant $\lambda > 0$, KT is bounded-error hard on average to approximate within an additive error of $\lambda \log n$. □

4.3 Bounded-Error Average-Case Robustness of Meta-Complexity

Our techniques also show that the meta-complexity of (resource-bounded) Kolmogorov complexity is "robust", i.e. a slight change in the underlying computation model has little effect on their hardness. Actually, for many resource-bounded variants of Kolmogorov complexity, such as KT, NC^1-K^t, and L-K^t, either all of them admit bounded-error polynomial-time heuristics, or none of them do. (See Section 3.2 for their definition.)

Theorem 4.10. *The following are equivalent:*

1. *There is a one-way function computable in* $\oplus\mathsf{L}$.
2. *There is a one-way function computable in* $\mathsf{DLOGTIME}$.
3. *The search version of* KT *is hard on average.*
4. *For every constant* $\lambda > 0$, KT *is hard on average to approximate within an additive error of* $\lambda \log n$.
5. *There is a polynomial* $t(\cdot)$ *such that the search version of* NC^1-K^t *is hard on average.*
6. *For every constant* $\lambda > 0$ *and polynomial* $t(\cdot)$ *such that* $t(n) > 2n$, NC^1-K^t *is hard on average to approximate within an additive error of* $\lambda \log n$.
7. *There is a polynomial* $t(\cdot)$ *such that the search version of* L-K^t *is hard on average.*
8. *For every constant* $\lambda > 0$ *and polynomial* $t(\cdot)$ *such that* $t(n) > 2n$, L-K^t *is hard on average to approximate within an additive error of* $\lambda \log n$.

Proof Sketch. (2) $\implies$ (1), (4) $\implies$ (3), (6) $\implies$ (5), and (8) $\implies$ (7) are trivial.

(3) $\implies$ (2): Directly from Theorem 4.1.

(5) $\implies$ (2) and (7) $\implies$ (2): The construction from [70, Section 4] gives a one-way function computable in $\mathsf{ALOGTIME}$ (i.e., uniform NC^1) based on the hardness of NC^1-K^t. By Theorem 3.21, there is a one-way function computable in $\mathsf{DLOGTIME}$. The same argument works for L-K^t.

(1) $\implies$ (4): Directly from Theorem 4.4.

(1) $\implies$ (6): Consider the condEP-PRG G computable in $\mathsf{DTIME}[c \log n]$ that we constructed in the proof of Theorem 4.4, where c is some constant. Let $t'(n) := n^{O(c)}$, then for every $x \in \{0,1\}^n$ that in the range of G, NC^1-$\mathrm{K}^{t'}(x) \leq n - \Theta(\log n)$. It follows that there is a polynomial t' such that NC^1-$\mathrm{K}^{t'}$ is hard on average to approximate.

To prove that NC^1-K^t is hard on average to approximate for every polynomial t, we use a padding trick. (See also [70, Theorem 5.6].) Let $\varepsilon > 0$ be a small enough constant, and $n_1 = n^\varepsilon$. Consider the generator $G'(x, r) = G(x) \circ r$, where $|x| = n_1$ and $|r| = n - n_1$. It is easy to see that if G is a condEP-PRG, then G' is also a condEP-PRG. For every $x \in \{0,1\}^n$ that is in the range of G', if we take ε to be a small enough constant, we have NC^1-$\mathrm{K}^t(x) \leq n - \Theta(\log n)$. Since G' is pseudorandom, NC^1-K^t is hard on average to approximate.

(1) $\implies$ (8): The same argument as in (1) $\implies$ (6) also works for L-K^t. □

4.4 Zero-Error Average-Case Reductions

Our techniques actually imply reductions among MKTP, NC^1-MINKT, and L-MINKT. A closer look at these reductions reveals that they are not only *two-sided error* average-case reductions, but also *zero-error* ones! This allows us to prove new relations between the *zero-error* average-case complexity of variants of MINKT and MKTP.

The standard definition of an average-case complexity class, such as AvgZPP, is a class of pairs $(L, \mathcal{D})$ where L is a language, and $\mathcal{D}$ is a distribution ensemble over inputs. (See, e.g., [12, Chapter 18].) In this section, we only deal with the uniform distribution as the input distribution. Therefore, for simplicity, we define AvgZPP as a class of languages rather than (language, distribution) pairs.

Definition 4.11. Let L be a language and $\delta < 1$ be a constant. We say $L \in \mathsf{Avg}_\delta\mathsf{ZPP}$ if there is a zero-error PPT heuristic $\mathcal{H}$ such that the following are true: (To emphasize that $\mathcal{H}$ is a randomized heuristic, we use $\mathcal{H}(x; r)$ to denote the output of $\mathcal{H}$ on input x and randomness r.)

- For every input $x \in \{0,1\}^\star$ and $r \in \{0,1\}^{\mathrm{poly}(|x|)}$, $\mathcal{H}(x; r) \in \{L(x), \perp\}$.
- For every integer n, $\Pr_{\mathbf{x} \leftarrow \mathcal{U}_n, \mathbf{r} \leftarrow \mathcal{U}_{\mathrm{poly}(n)}}[\mathcal{H}(\mathbf{x}; \mathbf{r}) = \perp] \leq \delta$.

Let $\mathsf{AvgZPP} := \bigcup_{\delta<1} \mathsf{Avg}_\delta\mathsf{ZPP}$.

We consider the parameterized versions of MKTP and MINKT in this section. Let $t(n) \leq \mathrm{poly}(n)$ be a time bound and $s(n) \leq n$ be a size parameter. We define $\mathrm{MKTP}[s] := \{x : \mathrm{KT}(x) \leq s(|x|)\}$, and $\mathrm{MINK}^t[s] := \{x : \mathrm{K}^{t(|x|)}(x) \leq s(|x|)\}$. The problems NC^1-$\mathrm{MINK}^t[s]$ and L-$\mathrm{MINK}^t[s]$ are defined similarly.

A language L is *ε-sparse* if for every integer n, $\Pr_{\mathbf{x} \leftarrow \mathcal{U}_n}[\mathbf{x} \in L] \leq \varepsilon$. From Fact 3.8, for every unbounded function $f(n) = \omega(1)$, $\mathrm{MKTP}[n - f(n)]$ and $\mathrm{MINK}^{\mathrm{poly}(n)}[n - f(n)]$ are $o(1)$-sparse. In general, to solve an $o(1)$-sparse problem L on average, it suffices to design a heuristic that distinguishes every instance in L from the random instances. Therefore, the following notion of reductions will be convenient for studying the zero-error average-case complexity of sparse problems:

Definition 4.12. Let L_1, L_2 be two problems. We say there is a *one-sided mapping reduction* from L_1 to L_2, if there are polynomials $p(\cdot)$, $m(\cdot)$ and a randomized polynomial-time mapping $\mathsf{Red} : \{0,1\}^n \times \{0,1\}^{p(n)} \to \{0,1\}^{m(n)}$ such that the following holds for every large enough n.

- For every $x \in L_1 \cap \{0,1\}^n$ and $r \in \{0,1\}^{p(n)}$, it holds that $\mathsf{Red}(x;r) \in L_2$.
- The distribution of $\mathsf{Red}(\mathcal{U}_n; \mathcal{U}_{p(n)})$ is equal to $\mathcal{U}_{m(n)}$.

Remark 4.13. Here we require that the reduction maps the uniform distribution to the uniform distribution exactly. In some cases, this requirement is too strong, and we only need that $\mathcal{U}_{m(n)}$ *dominates* $\mathsf{Red}(\mathcal{U}_n; \mathcal{U}_{p(n)})$. (See [12, Definition 18.6].) Nevertheless, thanks to the perfect randomized encodings [11], we are able to design reductions as strong as Definition 4.12.

In short, a one-sided mapping reduction (among sparse problems) maps a Yes instance to a Yes instance and maps a random instance to a random instance. It is easy to see that such reductions preserve the property of being in AvgZPP.

Fact 4.14. *Let L_1, L_2 be two $o(1)$-sparse problems. Suppose that there is a one-sided mapping reduction Red from L_1 to L_2. If there is a constant $\delta_2 < 1$ such that $L_2 \in \mathsf{Avg}_{\delta_2}\mathsf{ZPP}$, then there is a constant $\delta_1 < 1$ such that $L_1 \in \mathsf{Avg}_{\delta_1}\mathsf{ZPP}$.*

For every $s_1(n) \le s_2(n)$, there is a one-sided mapping reduction from $\mathrm{MKTP}[s_1(n)]$ to $\mathrm{MKTP}[s_2(n)]$. (The identity mapping is a valid reduction [48].) Similarly, for every s_1, t_1, s_2, t_2 such that an alternating machine of description length s_1 and (alternating) time $\log t_1$ can be compiled into a deterministic machine of description length s_2 and space $\log t_2$, there is a one-sided mapping reduction from $\mathsf{NC}^1\text{-MINK}^{t_1}[s_1]$ to $\mathsf{L}\text{-MINK}^{t_2}[s_2]$. (Again, the identity mapping is a valid reduction.)

Now we present a one-sided mapping reduction from L-MINKT to MKTP. Actually, the reduction we present is from L-MINKT to $\mathrm{MINK}^{t'}$, where $t'(n) = \lambda \log n$ for some absolute constant $\lambda > 0$.

Theorem 4.15. *For every polynomial $t(\cdot)$ and integer $c > 0$, there is a constant $c' > 0$ such that there is a one-sided mapping reduction Red from $\mathsf{L}\text{-MINK}^t[n - c' \log n]$ to $\mathrm{MINK}^{t'}[n - c \log n]$.*

Proof. For convenience, denote $s(n) := n - c' \log n$. Let $x \in \{0,1\}^n$ be an input to $\mathsf{L}\text{-MINK}^t[s]$.

The reduction is simple. It fixes $N := \mathrm{poly}(t(n))$ and reduces a length-n input to a length-$\tilde{N}$ input, where $\tilde{N} := n \cdot \ell_{\mathrm{DCMD}}(N)$. For every bit x_i ($1 \le i \le n$), it samples a uniformly random string $s_i \in \{0,1\}^{\ell_{\mathrm{DCMD}}(N)}$ conditioned on that $\mathrm{DCMD}(s_i) = x_i$. Finally, it outputs the concatenation of $s_1, s_2, \ldots, s_n$.

Since DCMD is balanced (Corollary 3.18), the reduction maps a random instance to a random instance. Now it remains to show that it maps a Yes instance to a Yes instance.

Suppose x is a Yes instance. Denote $\mathsf{Red}(x;r) := s_1 \circ s_2 \circ \cdots \circ s_n$, where r is the random coins that our reduction uses. For $t'(n) = \lambda \log n$, we want to prove that $\mathrm{K}^{t'}(\mathsf{Red}(x;r)) \le \tilde{N} - c \log \tilde{N}$.

Let U be the universal Turing machine we consider. Then there is a description d of length at most $s(n)$ such that for every $1 \le i \le n+1$ and every $b \in \{0,1,\star\}$, $U^d(i,b)$ accepts in

space $\log t(n)$ if and only if $x_i = b$. Since CMD is L-hard under projections (Theorem 3.19), for $N = \text{poly}(t(n))$, there is a DLOGTIME-computable projection

$$p_x : \{0,1\}^{s(n)} \times [n+1] \times \{0,1,\star\} \to \{0,1\}^{\ell_{\text{CMD}}(N)}$$

such that for every $1 \le i \le n+1$ and $b \in \{0,1,\star\}$, $x_i = b$ if and only if $\text{CMD}(p_x(d,i,b)) = 1$.

The description of $\mathsf{Red}(x;r)$ contains the string d and n strings $s'_1, s'_2, \ldots, s'_n$ of length $\ell_{\text{DCMD}}(N) - 1$ each. Let P_{CMD} be the DLOGTIME-computable projection in Theorem 3.17. The string s'_i is chosen such that $P_{\text{CMD}}(p_x(d,i,1), s'_i) = s_i$. (Note that $\text{CMD}(p_x(d,i,1)) = \text{DCMD}(s_i)$, so each s'_i exists and is unique.)

Let $1 \le i \le |\mathsf{Red}(x;r)|$. To compute the i-th bit of $\mathsf{Red}(x;r)$, we first "locate" i by computing $k := \lfloor \frac{i-1}{\ell_{\text{DCMD}}(N)} \rfloor + 1$ and $j := i - \ell_{\text{DCMD}}(N)(k-1)$. Now, the i-th bit of $\mathsf{Red}(x;r)$ is the j-th bit of s_k. We can simply calculate the j-th bit of $P_{\text{CMD}}(p(d,i,1), s'_i)$, which takes $\lambda \log \tilde{N}$ time for some absolute constant $\lambda > 0$.

It follows that whenever $\mathsf{L}\text{-}\mathrm{K}^t(x) \le n - c' \log n$, regardless of the random bits r we choose, there is a description that allows us to quickly retrieve each bit of $\mathsf{Red}(x;r)$. Moreover, the description has length $n - c' \log n + n(\ell_{\text{DCMD}}(N) - 1) = \tilde{N} - c' \log n$. If the constant c' is big enough compared with c, then $\mathsf{Red}(x;r)$ is a Yes instance of $\text{MINK}^{t'}[\tilde{N} - c \log \tilde{N}]$. □

Note that for $c > \lambda$, $\text{MINK}^{t'}[n - c \log n]$ reduces to $\text{MKTP}[n - (c - \lambda) \log n]$ via the identity mapping. (See the proof of Theorem 4.16.) Therefore, Theorem 4.15 shows a one-sided mapping reduction from some version of MINKT to some version of MKTP. To the best of our knowledge, this reduction is the first result of its kind.

Moreover, Theorem 4.15 demonstrates the *robustness* of meta-complexity w.r.t. the zero-error average-case complexity. In particular:

Theorem 4.16. *Let $t(\cdot)$ be a fixed polynomial such that $t(n) > 2n$. The following are equivalent:*

1. *There is a constant $c > 0$ such that $\mathsf{NC}^1\text{-}\text{MINK}^t[n - c \log n] \in \mathsf{AvgZPP}$.*
2. *There is a constant $c > 0$ such that $\mathsf{L}\text{-}\text{MINK}^t[n - c \log n] \in \mathsf{AvgZPP}$.*
3. *There is a constant $c > 0$ such that $\text{MINK}^{t'}[n - c \log n] \in \mathsf{AvgZPP}$, where $t'(n) = \lambda \log n$ is defined above.*

Moreover, the above items are implied by the following items:

4. *There is a constant $c > 0$ such that $\text{MKTP}[n - c \log n] \in \mathsf{AvgZPP}$.*

Proof. (4) $\implies$ (3): It suffices to show that for every $c > 0$, the identity mapping reduces $\text{MINK}^{t'}[n - c' \log n]$ to $\text{MKTP}[n - c \log n]$ where $c' = c + \lambda$. Let $x \in \text{MINK}^{t'}[n - c' \log n] \cap \{0,1\}^n$, and d be a description of length $n - c' \log n$ witnessing the fact that $\mathrm{K}^{t'}(x) \le n - c' \log n$. Since $(d, t'(n))$ is also a witness that $\mathrm{KT}(x) \le n - c \log n$. we have $x \in \text{MKTP}[n - c \log n]$.

(3) $\implies$ (2): By Theorem 4.15 and Fact 4.14.

(3) $\implies$ (1): Note that the only property of L used in the proof of Theorem 4.15 is that CMD is hard for L. (In other words, $\mathsf{L} \subseteq \oplus\mathsf{L}$.) As CMD is also hard for NC^1, the proof of Theorem 4.15 is also true for L-MINKT replaced by NC^1-MINKT.

(2) $\implies$ (3): Every machine that runs in $t'(n)$ time also runs in $t'(n)$ space. Therefore, for every $c > 0$, the identity mapping reduces $\text{MINK}^{t'}[n - c' \log n]$ to $\mathsf{L}\text{-MINK}^{t_1}[n - c \log n]$, where $t_1(n) = 2^{O(t'(n))}$. We can use a padding trick [70, Theorem 5.6] to reduce $\mathsf{L}\text{-MINK}^{t_1}$ to $\mathsf{L}\text{-MINK}^{t}$.

(1) $\implies$ (3): The same argument as (2) $\implies$ (3) also works for $\mathsf{NC}^1\text{-K}^t$. □

5 Tighter Connections between Meta-Complexity and One-Way Functions

In this section, we present a tighter connection between the hardness of MINKT (or MKTP) and the maximum security of *weak* one-way functions. We first define the *security* of weak one-way functions.

Definition 5.1. Let $f : \{0,1\}^n \to \{0,1\}^n$ be a polynomial-time computable function. We say f is a weak one-way function *with security* $S(n)$ if there is a polynomial $p(\cdot)$ such that for every circuit C of size $S(n)$,

$$\Pr_{\mathbf{x} \leftarrow \mathcal{U}_n}[C(f(\mathbf{x})) \in f^{-1}(f(\mathbf{x}))] \leq 1 - \frac{1}{p(n)}.$$

Our main results are as follows.

Theorem 5.2. *Let $S(n)$ be any monotone function such that $S(n + O(\log^2 n)) \leq S(n) \cdot n^{O(\log n)}$. The following are equivalent:*

(a) There is a weak one-way function with security $S(n) \cdot n^{\Theta(\log n)}$.

(b) There are polynomials p, t such that the search version of K^t requires $S(n) \cdot n^{\Theta(\log n)}$ size to compute on a $1 - 1/p(n)$ fraction of inputs.

(c) For every constant $\lambda > 0$, there are polynomials p, t such that K^t requires $S(n) \cdot n^{\Theta(\log n)}$ size to $(\lambda \log n)$-approximate on a $1 - 1/p(n)$ fraction of inputs.

Theorem 5.3. *Suppose there is a weak one-way function f with security $2^{\Omega(n)}$ computable in* DLOGTIME. *Then there is a polynomial p such that* KT *requires $2^{\Omega(n)}$ size to compute on a $1 - 1/p(n)$ fraction of inputs.*

Remark 5.4. A few remarks are in order.

- We only consider non-uniform adversaries in this section. The reason is that we will use Impagliazzo's hardcore lemma (Lemma 5.21), which only works for non-uniform adversaries. We remark that there are hardcore lemmas that also work for uniform adversaries: if there is no time-t' algorithm that inverts a weak one-way function on a

$1 - o(1)$ fraction of inputs, then there is no time-t algorithm that non-trivially inverts every hardcore of the same one-way function. However, we do not know whether the dependence of t' on t is tight. Theorem 4.5 of [96] achieves $t' = \mathrm{poly}(t)$, but we need $t' = t \cdot \mathrm{polylog}(t)$. We leave this issue for future work.

- Our equivalence only holds for *weak* one-way functions. Indeed, it is an open problem whether the existence of *exponentially*-hard weak one-way functions is equivalent to the existence of *exponentially*-hard strong one-way functions [35]. Yao's hardness amplification theorem (Theorem 3.12) blows up the input length by a polynomial factor. Therefore given a $2^{\Omega(n)}$-hard weak one-way function, it only produces a $2^{n^{\Omega(1)}}$-hard strong one-way function.

- Our result for KT (Theorem 5.3) is weaker than our result for K^t. In particular, suppose the one-way function has security $2^{\alpha n}$, we can only show that KT requires $2^{\beta n}$ size on average, for some constant β that is much smaller than α.

- The $n^{\Theta(\log n)}$ overhead in Theorem 5.2 arises from the use of *extractors* and *hardcore functions* in our construction. Specifically, we rely on explicit extractors that extract all min-entropy, and the best known such construction has seed length $O(\log^2 n)$ [40]. In addition, the hardcore function used in Theorem 5.11 requires a seed length of $O(k \log(n/\varepsilon))$, which is also $O(\log^2 n)$ under our parameter setting. These result in the $n^{\Theta(\log n)}$ factor in our main theorem.

We rely on the construction of condEP-PRGs from weak one-way functions in [103, 70], thus we structure this section as follows. In Section 5.1, we define *extractors* and *hardcore functions*, which are technical building blocks of the construction. In Section 5.2, we describe the construction in [103, 70]. (The correctness of this construction is proved in Section 5.6.) The proofs of Theorem 5.2 and 5.3 appear in Section 5.3 and 5.4 respectively.

5.1 Technical Building Blocks

5.1.1 Extractors

Definition 5.5. A function $\mathsf{Ext} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ is a (k, ε)-extractor if for every random variable X over $\{0,1\}^n$ such that $\mathrm{H}_\infty(X) \geq k$, the statistical distance between $\mathsf{Ext}(X, \mathcal{U}_d)$ and $\mathcal{U}_m$ is at most ε.

Moreover, Ext is a *strong* (k, ε)*-extractor* if for every random variable X as above, the statistical distance between $\mathsf{Ext}(X, \mathcal{U}_d)$ and $\mathcal{U}_m$ is at most ε, even *conditioned on the seed*. That is, the statistical distance between the following two distributions is at most ε:

$$\mathcal{D}_1 := (\mathbf{r} \circ \mathsf{Ext}(\mathbf{x}, \mathbf{r}) \mid \mathbf{r} \leftarrow \mathcal{U}_d, \mathbf{x} \leftarrow X), \text{ and } \mathcal{D}_2 := \mathcal{U}_{d+m}.$$

5.1.2 Hardcore Functions

Definition 5.6. Let $\varepsilon = \varepsilon(n) > 0$, $L = L(n) \leq \mathrm{poly}(n)$, $\mathsf{HC} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ be a function, and R be a probabilistic oracle algorithm. We say HC is a *hardcore function* with reconstruction algorithm R, distinguishing probability ε, and list size L, if the following holds.

- On every oracle O, R^O outputs a list of L strings of length n.
- For every string x and every oracle O that ε-distinguishes $\mathcal{U}_d \circ \mathsf{HC}(x, \mathcal{U}_d)$ from $\mathcal{U}_{d+m}$, x is in the list outputted by R^O w.p. $\geq 1/2$.

Our definition of hardcore functions indeed implies the standard definition in [36]:

Fact 5.7. *Let $\mathsf{HC} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ be a hardcore function with a $\mathrm{poly}(n)$-time reconstruction algorithm, distinguishing probability $\varepsilon = 1/\mathrm{poly}(n)$, and list size $L \leq \mathrm{poly}(n)$.*

Let f be any one-way function, $\mathbf{x} \leftarrow \mathcal{U}_n$, and $\mathbf{r} \leftarrow \mathcal{U}_d$. No polynomial-size adversary can 2ε-distinguish the distribution $f(\mathbf{x}) \circ \mathbf{r} \circ \mathsf{HC}(\mathbf{x}, \mathbf{r})$ from the distribution $f(\mathbf{x}) \circ \mathbf{r} \circ \mathcal{U}_m$.

Proof. Let $\mathcal{A}$ be an adversary of size $\mathrm{poly}(n)$ that 2ε-distinguishes the distribution $f(\mathbf{x}) \circ \mathbf{r} \circ \mathsf{HC}(\mathbf{x}, \mathbf{r})$ from $f(\mathbf{x}) \circ \mathbf{r} \circ \mathcal{U}_m$. Say $x \in \{0,1\}^n$ is *good* if $\mathcal{A}$ can ε-distinguish $f(x) \circ \mathbf{r} \circ \mathsf{HC}(x, \mathbf{r})$ from $f(x) \circ \mathbf{r} \circ \mathcal{U}_m$. Then by a Markov bound, at least an ε fraction of inputs x are good. We will use $\mathcal{A}$ to invert $f(x)$ on every good input x in probabilistic polynomial time. Our inversion algorithm will have success probability $1/2$ on a good x; as $(\varepsilon/2) > 1/\mathrm{poly}(n)$, this contradicts the one-wayness of f.

On input $y = f(x)$ where x is good, define the oracle

$$O(z) := \mathcal{A}(y, z).$$

Then O can ε-distinguish $\mathcal{U}_d \circ \mathsf{HC}(x, \mathcal{U}_d)$ from $\mathcal{U}_{d+m}$. The reconstruction algorithm R^O outputs a list of size $\mathrm{poly}(n)$ which contains x. We could easily find and output any element x' in this list such that $f(x') = y$. With probability $1/2$ over the internal randomness of R, we invert y successfully. □

5.2 CondEP-PRGs from Weak One-Way Functions

In this section, we present the following construction from weak one-way functions to condEP-PRGs.

Construction 5.8 ([103, 70])**.** Let $0 < \varepsilon < \frac{1}{10n^2}$ be the desired security parameter of the condEP-PRG (i.e., it should be $O(\varepsilon)$-indistinguishable from uniformly random strings). Let $\delta > 0$ and f be a weak one-way function that is hard to invert on a $(1-\delta)$ fraction of inputs. Let $\alpha > 0$ be the desired stretch of our condEP-PRG. Suppose we have the following objects:

- For every k, a strong (k, ε)-extractor $\mathsf{Ext} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ with optimal output length, where $d := d_{\mathsf{Ext}}(n, \varepsilon)$ and $m := k - 2\log(1/\varepsilon) - O(1)$. We write the extractor as $\mathsf{Ext}^{(k)}$ if we need to emphasize the min-entropy parameter k.
- For $k_{\mathsf{HC}} := \alpha + \log(n/\delta) + 4\log(1/\varepsilon) + O(1)$, a hardcore function $\mathsf{HC} : \{0,1\}^n \times \{0,1\}^{d'} \to$

$\{0,1\}^{k_{\mathsf{HC}}}$ with $\mathrm{poly}(n/\varepsilon)$-time reconstruction algorithm R, distinguishing probability ε, and list size $L \le \mathrm{poly}(n/\varepsilon)$, where $d' := d_{\mathsf{HC}}(n, k_{\mathsf{HC}}, \varepsilon)$.

Let $G_{n,r} : \{0,1\}^n \times \{0,1\}^d \times \{0,1\}^d \times \{0,1\}^{d'} \to \{0,1\}^{n+2d+d'+\alpha}$ be the following construction:

$$G_{n,r}(x, z_1, z_2, z_3) := z_1 \circ \mathsf{Ext}^{(r-1)}(x, z_1) \circ z_2 \circ \mathsf{Ext}^{(\lfloor n-r-\log(2n/\delta)\rfloor)}(f(x), z_2) \circ z_3 \circ \mathsf{HC}(x, z_3).$$

Theorem 5.9. *Let $\varepsilon, \delta, \alpha, f$ be defined as in Construction 5.8. If $\varepsilon \ge 1/\mathrm{poly}(n)$ and $L \le \mathrm{poly}(n)$, then there is a function $r : \mathbb{N} \to \mathbb{N}$ such that $G = \{G_{n,r(n)}\}_{n\in\mathbb{N}}$ is a condEP-PRG with stretch α and security 4ε.*

More precisely, let $\tilde{n} = n + 2d + d'$. Suppose that for every subset $\mathcal{D} \subseteq \{0,1\}^{\tilde{n}}$ such that $\mathrm{H}(G(\mathcal{D})) \ge \tilde{n} - \Omega(\log(\frac{n}{\delta\varepsilon}))$ and every k, there is an adversary of size s that 4ε-distinguishes $G_{n,k}(\mathcal{D})$ from the uniform random distribution. Then there is an adversary of size $s \cdot \mathrm{poly}(nL/\varepsilon)$ that inverts f on a $1-\delta$ fraction of inputs.

The proof basically follows from [70], and we present a self-contained proof in Section 5.6. However, there are two major differences between our proof and the proof in [70]:

- We replace the extractors and hardcore functions with better constructions. In particular, our extractors and hardcore functions in Section 5.3 only require $O(\log^2 n)$ random bits.
- More importantly, in the very beginning, we need to transform the weak one-way function into a strong one. [70] uses hardness amplification (Theorem 3.12) to implement this step. However, Theorem 3.12 does not preserve *exponential* security, therefore we use *Impagliazzo's hardcore lemma* [56] instead. We only obtain a strong one-way function on a "hardcore" distribution of inputs (instead of the uniform distribution), but this already suffices for our purpose.

5.2.1 Warm-Up: Proof of Theorem 3.14

Theorem 5.9 immediately implies Theorem 3.14.

Theorem 3.14 ([70])**.** *There is a function* EP-PRG *computable in* ALOGTIME *such that the following holds. For any one-way function $f : \{0,1\}^\star \to \{0,1\}^\star$ and any constant $\gamma > 0$, let $G(x,z) = \mathsf{EP\text{-}PRG}(\gamma, x, f(x), z)$, then G is a condEP-PRG with stretch $\gamma \log n$ and security $1/n^\gamma$.*

We first introduce the (very simple) extractors and hardcore functions used in [103, 70].

- The extractors are derived from the *leftover hash lemma* [42]. (See also [95, Theorem 6.18].) Let $h : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ be a pairwise independent family of hash functions, where $d = O(n+m)$, then for every k, ε such that $m = k - 2\log(1/\varepsilon)$, h is also a strong (k,ε)-extractor.

We instantiate the pairwise independent hash family by Toeplitz matrices.[17] More precisely, our keys will have length $d := n + m - 1$, and every $key \in \{0,1\}^{n+m-1}$ corresponds to a Toeplitz matrix. For every $1 \le i \le m$ and every input $x \in \{0,1\}^n$, the i-th output of $H(x, key)$ is the inner product of x and $key_{i\sim(i+n-1)}$ (the substring of key from the i-th bit to the $(i+n-1)$-th bit) in GF(2). In other words, $\mathsf{Ext}(x, key)$ is the concatenation of $\langle x, key_{i\sim(i+n-1)}\rangle$ for each i, where $\langle\cdot,\cdot\rangle$ denotes inner product.

- Let $\mathsf{GL} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^k$ be the Goldreich–Levin hardcore function.

  In [36], GL is defined in terms of Toeplitz matrices (again). Let $d := n + k - 1$. For every $x \in \{0,1\}^n$, $r \in \{0,1\}^d$ and $1 \le i \le k$, the i-th output bit of $\mathsf{GL}(x, r)$ is the inner product of x and $r_{i\sim(i+n-1)}$ in GF(2). Also, it is shown in [36] that for every $\varepsilon > 0$, GL is a hardcore function with distinguishing probability ε and list size $\mathrm{poly}(n \cdot 2^k/\varepsilon)$.

Proof Sketch of Theorem 3.14. We can plug the parameters $\varepsilon := \frac{1}{4n^\gamma}$, $\alpha := \gamma \log n$, $\delta := 1/2$ into Theorem 5.9. The list size of GL is $L \le \mathrm{poly}(n)$. Theorem 5.9 gives us a function $r : \mathbb{N} \to \mathbb{N}$ such that $\{G_{n,r(n)}\}$ is a condEP-PRG with stretch $\gamma \log n$ and security $1/n^\gamma$. We can easily construct a uniform condEP-PRG with essentially the same stretch and security: We parse the input as an integer $r \le n$, a string x of length n, and some garbage w. Then we output $G_{n,r}(x) \circ w$.

Now we implement EP-PRG in alternating time $c \log n$, for some absolute constant $c > 0$ independent of γ. On input $(\gamma, x, f(x), z, i)$, we want to compute the i-th output bit of our condEP-PRG. This bit is either equal to some input bit, or the inner product of two length-n sub-strings of the input. It is easy to implement either case in alternating $O(\log n)$ time. □

5.3 Proof of Theorem 5.2

To prove Theorem 5.2, we replace the leftover hash lemma and GL by extractors and hardcore functions with very short seed length:

Theorem 5.10 ([40, Theorem 5.14])**.** *Let $d_{\mathsf{Ext}}(n, \varepsilon) := O(\log n \cdot \log(n/\varepsilon))$, then for every $1 \le k \le n$ and $\varepsilon > 0$, there is a strong (k, ε)-extractor $\mathsf{Ext} : \{0,1\}^n \times \{0,1\}^{d_{\mathsf{Ext}}(n,\varepsilon)} \to \{0,1\}^m$, where $m = k - 2\log(1/\varepsilon) - O(1)$ is optimal.*

We observe that the "k-wise direct product generator" used in [46, 45] is a good hardcore function:

Theorem 5.11. *Let $d_{\mathsf{HC}}(n, k, \varepsilon) := O(k \log(n/\varepsilon))$, then there is a hardcore function $\mathsf{HC} : \{0,1\}^n \times \{0,1\}^{d_{\mathsf{HC}}(n,k,\varepsilon)} \to \{0,1\}^k$ with a $\mathrm{poly}(n2^k/\varepsilon)$-time reconstruction algorithm R, distinguishing probability ε, and list size $L \le 2^k \cdot \mathrm{poly}(k/\varepsilon)$.*

Proof Sketch. Consider the function $\mathsf{DP} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^{d+k}$ defined in [45, Theorem 7.1]. The first d bits of $\mathsf{DP}(x, z)$ is always equal to z, and we let $\mathsf{HC}(x, z)$ be the remaining k bits of $\mathsf{DP}(x, z)$.

[17] An $n \times m$ matrix M is *Toeplitz* if $M_{i,j} = M_{i+1,j+1}$ holds for every $1 \le i < n$, $1 \le j < m$. We can represent a Toeplitz matrix by $n + m - 1$ elements, namely the elements in the first row and the first column.

In [45], the reconstruction algorithm is stated as $R^O : \{0,1\}^a \times \{0,1\}^r \to \{0,1\}^n$. Here, $a \le k + O(\log(k/\varepsilon))$ is the "advice complexity" of DP, the first a input bits correspond to the advice, and the remaining $r = \mathrm{poly}(n/\varepsilon)$ input bits are random coins used by R. For every $x \in \{0,1\}^n$ and every oracle O that ε-distinguishes $\mathsf{DP}(x, \mathcal{U}_d)$ from $\mathcal{U}_{d+k}$, we have

$$\Pr_{\mathbf{w} \leftarrow \mathcal{U}_r}[\exists \alpha \in \{0,1\}^a, R^O(\alpha, \mathbf{w}) = x] \ge 3/4.$$

Our reconstruction algorithm simply samples a random $\mathbf{w} \leftarrow \mathcal{U}_r$ and outputs $R^O(\alpha, \mathbf{w})$ for every $\alpha \in \{0,1\}^a$. It follows that the list size is $L(n,k,\varepsilon) \le 2^a \le 2^k \mathrm{poly}(k/\varepsilon)$. □

Now we use Construction 5.8 to prove Theorem 5.2.

Theorem 5.2. *Let $S(n)$ be any monotone function such that $S(n + O(\log^2 n)) \le S(n) \cdot n^{O(\log n)}$. The following are equivalent:*

(a) There is a weak one-way function with security $S(n) \cdot n^{\Theta(\log n)}$.

(b) There are polynomials p, t such that the search version of K^t requires $S(n) \cdot n^{\Theta(\log n)}$ size to compute on a $1 - 1/p(n)$ fraction of inputs.

(c) For every constant $\lambda > 0$, there are polynomials p, t such that K^t requires $S(n) \cdot n^{\Theta(\log n)}$ size to $(\lambda \log n)$-approximate on a $1 - 1/p(n)$ fraction of inputs.

Proof Sketch. (c) $\implies$ (b) is trivial.

(b) $\implies$ (a): Suppose that the search version of K^t requires $S(n) \cdot n^{\Theta(\log n)}$ size to solve on a $1/p(n)$ fraction of inputs, where p is a polynomial. The construction in [70, Section 4] shows that there is a weak one-way function f such that every adversary of size $S(n) \cdot n^{\Theta(\log n)}$ only inverts an $1 - 1/q(n)$ fraction of inputs, where $q(n) := O(n \cdot p(n)^2)$.

(a) $\implies$ (c): Suppose there is a constant $\lambda > 0$ such that, for every polynomial p, there is an algorithm of size $S(n) \cdot n^{\Theta(\log n)}$ that approximates K^t on a $1 - 1/p(n)$ fraction of inputs, within an additive error of $\lambda \log n$. Let f be a candidate weak one-way function, $\delta := 1/q(n)$ for any polynomial q, and $\varepsilon := 1/n^2$. Let $\alpha := (\lambda + C)\log n$ be the stretch of the condEP-PRG we construct, where C is a large absolute constant. Let $r : \mathbb{N} \to \mathbb{N}$ be any function. Consider the function $G_{n,r(n)}$ in Construction 5.8 where the input length of $G_{n,r(n)}$ is

$$\tilde{n} := n + 2d_{\mathsf{Ext}}(n, \varepsilon) + d_{\mathsf{HC}}(n, O(\log(n/(\delta\varepsilon))), \varepsilon) = n + O(\log^2 n).$$

Suppose G runs in $t(\tilde{n})$ time. Then every output y of G satisfies $\mathrm{K}^{t(|y|)}(y) \le \tilde{n} - (\lambda + 2)\log \tilde{n}$.

Consider any sequence of subsets $\mathcal{E} = \{\mathcal{E}_n \subseteq \{0,1\}^n\}$ such that $\mathrm{H}(G_{n,r(n)}(\mathcal{E}_n)) \ge \tilde{n} - \Omega(\log n)$. The same argument as in Lemma 4.7 shows that there is an adversary of size

$$S(\tilde{n}) \cdot \tilde{n}^{\Theta(\log \tilde{n})} \le S(n) \cdot n^{\Theta(\log n)}$$

that 4ε-distinguishes $G_{n,r(n)}(\mathcal{E}_n)$ from the uniform distribution. It follows from Theorem 5.9 that there is an adversary of size $S(n) \cdot n^{\Theta(\log n)}$ that inverts f on a $1 - \delta$ fraction of inputs. Therefore, there is no weak one-way function with hardness $S(n) \cdot n^{\Theta(\log n)}$. □

5.4 Proof of Theorem 5.3

To prove Theorem 5.3, we need a family of universal hash functions that admit very efficient randomized encodings, as constructed in [63, 10]. In [10], it was also proved that such hash functions are good extractors (by the leftover hash lemma) and hardcore functions (based on previous works [51, 16]).

In the construction of [11], for a (Boolean) function computable by a parity branching program of size S, its randomized encoding needs at least $\Omega(S^2)$ additional random input bits. Even worse, if such a function has m output bits, the randomized encoding requires $\Omega(mS^2)$ random input bits. However, to prove Theorem 5.3, we need to preserve *exponential* hardness of our one-way function, which means our extractors and hardcore functions can only have $O(n)$ random input bits. This is exactly what [10] does. In particular, for a "skew" circuit C of size S and possibly many outputs, the randomized encoding of C in [10] only requires $O(S)$ additional random inputs. Such circuits of linear size can already compute many powerful objects, e.g. universal hash functions [63].

5.4.1 Randomized Encodings for Skew Circuits

We introduce the randomized encodings in [10] in more detail.

We consider circuits that consist of AND and XOR gates of fan-in 2 and has multiple output gates. Let C be such a circuit and X be a subset of input variables. (For example, let $C : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$, we may think of X as the last d input variables.) We say C is *skew* with respect to X, if every AND gate in C has at least one child labeled by a constant or a variable in X. In particular, this implies that if we substitute the variables in X by (arbitrary) constants, the function that C computes is a *linear* function on variables not in X — each output bit is simply the XOR of a subset of these variables.

Let $C : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^{d+m}$ be a skew circuit w.r.t. the last d inputs such that the first d outputs of C is always equal to the last d inputs of C.[18] Let s be the number of internal (i.e. non-input, non-output) gates of C. The randomized encoding of C, denoted by $\tilde{C}$, is a function $\tilde{C} : \{0,1\}^n \times \{0,1\}^{d+s} \to \{0,1\}^{d+m+s}$ defined as follows:

- The inputs of $\tilde{C}$ are $x \in \{0,1\}^n$, $w \in \{0,1\}^d$, and $r \in \{0,1\}^s$.
- For each (input, internal, or output) gate $g \in C$, we associate a bit $r(g)$ with it. Each input gate is associated with its input value (i.e. $r(g) = x_i$ or w_i), the i-th internal gate is associated with $r(g) = r_i$, and every output gate is associated with $r(g) = 0$.
- The first d outputs of $\tilde{C}$ are simply w. The remaining $m + s$ outputs correspond to the internal gates and output gates of C. Let the i-th such gate be $g_i = g_j \triangledown g_k$ (where $\triangledown \in \{\mathsf{AND}, \mathsf{XOR}\}$), then the i-th output is $r(g_i)$ XOR $(r(g_j) \triangledown r(g_k))$.

It was shown in [10] that $\tilde{C}$ is a randomized encoding of C in the following sense. The space of encodings $\{0,1\}^{d+m+s}$ can be partitioned into $\{D_y\}_{y \in \{0,1\}^{d+m}}$ so that for every fixed

[18] That is, we pad the last d inputs at the beginning of our outputs, and the remaining m output bits are the "real" outputs of C. This is a technical restriction on C to ensure its randomized encoding exists.

$x \in \{0,1\}^n$ and $w \in \{0,1\}^d$, letting $y := C(x,w)$, the mapping $\tilde{C}(x,w,-) : \{0,1\}^s \to \{0,1\}^{d+m+s}$ is a bijection between $\{0,1\}^s$ and D_y. Let $\mathsf{Sim}(y)$ denote the uniform distribution over D_y (Sim stands for "simulator"), then Sim can be computed (i.e., sampled) in probabilistic polynomial time, and $\mathsf{Sim}(y)$ is the same distribution as $\tilde{C}(x,w,\mathcal{U}_s)$.

Theorem 5.12. *Let $m := n + \gamma \log n$, $C : \{0,1\}^{n'} \times \{0,1\}^d \to \{0,1\}^{d+m}$ be a circuit with s internal gates that is skew w.r.t. its second argument, and $g : \{0,1\}^n \to \{0,1\}^{n'}$ be an arbitrary function. Suppose that the function $G : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^{d+m}$ defined as*

$$G(x,w) := C(g(x),w)$$

is a condEP-PRG with stretch $\gamma \log n$ and security ε. Let $\tilde{C} : \{0,1\}^{n'} \times \{0,1\}^{d+s} \to \{0,1\}^{d+m+s}$ be the randomized encoding of C. Then the function $\tilde{G}(x,w,r) := \tilde{C}(g(x),w,r)$ is a condEP-PRG with stretch $\gamma \log n$ and security ε.

Proof. The stretch of $\tilde{G}$ is easy to verify.

Let $\mathcal{E} \subseteq \{0,1\}^n$ be the "event" such that (Pseudorandomness) and (Entropy-Preservation) in Definition 3.13 hold. Let $\tilde{\mathcal{E}} := \mathcal{E} \times \{0,1\}^s$. We first observe that $\tilde{G}(\tilde{\mathcal{E}})$ is the same distribution as $\mathsf{Sim}(G(\mathcal{E}))$. In fact, $\tilde{G}(\tilde{\mathcal{E}})$ is the distribution where we choose $(\mathbf{x},\mathbf{w}) \leftarrow \mathcal{E}$, $\mathbf{r} \leftarrow \mathcal{U}_s$, and output $\tilde{C}(g(\mathbf{x}),\mathbf{w};\mathbf{r})$; this is the same distribution as $\mathsf{Sim}(C(g(\mathbf{x}),\mathbf{w})) = \mathsf{Sim}(G(\mathcal{E}))$.

Now we verify that $\tilde{G}$ and $\tilde{\mathcal{E}}$ satisfies Definition 3.13.

Pseudorandomness. Suppose, for the sake of contradiction, that there is a PPT adversary $\mathcal{A}'$ such that

$$\Pr[\mathcal{A}'(\tilde{G}(\tilde{\mathcal{E}}))] - \Pr[\mathcal{A}'(\mathcal{U}_{d+m+s})] \geq \varepsilon.$$

Consider an adversary $\mathcal{A}$ that on input y, samples $\tilde{\mathbf{y}} \leftarrow \mathsf{Sim}(y)$ and returns $\mathcal{A}'(\tilde{\mathbf{y}})$. If $\mathbf{y} \leftarrow G(\mathcal{E})$, then $\tilde{\mathbf{y}}$ is distributed according to $\tilde{G}(\tilde{\mathcal{E}})$. On the other hand, if $\mathbf{y} \leftarrow \mathcal{U}_{d+m}$, since $\{D_y\}$ is a partition of $\{0,1\}^{d+m+s}$, we have that $\tilde{\mathbf{y}}$ is distributed according to $\mathcal{U}_{d+m+s}$. Hence, $\mathcal{A}$ distinguishes $G(\mathcal{E})$ from the uniform distribution with advantage $\geq \varepsilon$ as well, contradicting the security of G.

Entropy-preservation. Note that $\mathrm{H}(\tilde{G}(\tilde{\mathcal{E}})) = \mathrm{H}(\mathsf{Sim}(G(\mathcal{E}))) = \mathrm{H}(G(\mathcal{E})) + s$. Hence if $\mathrm{H}(G(\mathcal{E})) \geq n + d - \alpha$ for some $\alpha \geq 1$, then $\mathrm{H}(\tilde{G}(\tilde{\mathcal{E}})) \geq n + d + s - \alpha$ as well. □

5.4.2 Highly-Uniform Linear-Size Hash Functions

As we are dealing with KT complexity, we will need the randomized encoding to be computable in $\mathsf{DLOGTIME}$. Therefore, our skew circuits need to be *very uniform*. We state our definition of *uniform skew circuits* as follows; it is easy to see that if a family of skew circuits $\{C_n\}$ is uniform, then their randomized encodings can indeed be computed in $\mathsf{DLOGTIME}$.

Definition 5.13 (Uniform Skew Circuits)**.** Let $C = \{C_n : \{0,1\}^n \times \{0,1\}^{d(n)} \to \{0,1\}^{s(n)}\}$ be a family of skew circuits, where $d(n)$ and $s(n)$ are computable in time $O(\log n)$. Moreover, assume

that the fan-out of every gate is at most 2, and the last $s(n)$ gates (i.e., gates with the largest indices) are output gates.

We say that C is a *uniform* family of skew circuits, if there is an algorithm $\mathcal{A}$ with time complexity linear in its input length that, on inputs n, i (in binary), outputs the information about the i-th gate in C_n. This includes the gate type (input, AND, or XOR), indices of its input gates (if they exist), and indices of the (at most 2) gates it feeds to.

Remark 5.14. It may seem strange that we need to output not only predecessors but also successors of each gate. The reason is that in [63], we will need to *reverse* each wire when we transform an encoding circuit to an "exposure resilient function". In particular, after that construction, the predecessors of each gate will become their previous successors. See Section 5.7.4 for details.

We need a family of universal hash functions $\mathcal{H} = \{h_{n,m} : \{0,1\}^n \times \{0,1\}^k \to \{0,1\}^m\}$ in [63], where $k = O(n+m)$. This family has the following important property: $\mathcal{H}$ can be computed by a family of linear-size uniform circuits that are skew w.r.t. the second argument (i.e. the last k bits).

Theorem 5.15. *For every integer n, m where $m = O(n)$, there exists an integer $k = O(n)$ and a family of universal hash functions $\{h_{n,m} : \{0,1\}^n \times \{0,1\}^k \to \{0,1\}^m\}$ such that $h_{n,m}$ can be computed by a uniform family of linear-size circuits that are skew w.r.t. the second argument.*

In [63], the authors showed that $\mathcal{H}$ can be computed by a family of linear-size skew circuits, but they did not show that the circuits are uniform. Therefore, we include a proof sketch of Theorem 5.15 in Section 5.7, with an emphasis on the uniformity of these circuits.

By the leftover hash lemma of [42], $\{h_{n,m}\}$ is a strong (k,ε)-extractor whenever $m = k - 2\log(1/\varepsilon)$. It was proved by [16] (based on [51]) that $\{h_{n,m}\}$ are good hardcore functions:

Lemma 5.16. *For every $\varepsilon > 0$, $h_{n,m}$ is a hardcore function with distinguishing probability ε and a reconstruction algorithm of $\mathrm{poly}(2^m \cdot n/\varepsilon)$ time. (As a result, the list size is also $\mathrm{poly}(2^m \cdot n/\varepsilon)$.)*

5.4.3 Proof of Theorem 5.3

Theorem 5.3. *Suppose there is a weak one-way function f with security $2^{\Omega(n)}$ computable in* DLOGTIME. *Then there is a polynomial p such that* KT *requires $2^{\Omega(n)}$ size to compute on a $1 - 1/p(n)$ fraction of inputs.*

Proof Sketch. Let $\delta = 1/\mathrm{poly}(n)$ such that f is hard to invert on a $(1-\delta)$ fraction of inputs. We plug the hash functions h (which are also extractors and hardcore functions) into Construction 5.8 and build a condEP-PRG $G : \{0,1\}^{n_1} \to \{0,1\}^{n_1+\alpha}$ with stretch $\alpha := O(\log n)$ and security $4\varepsilon \le 1/n^{10}$. Here, since the seed length of h is $O(n)$, we have $n_1 = O(n)$. Moreover, by Theorem 5.9, the condEP-PRG is 4ε-indistinguishable from the uniform distribution by $2^{\Omega(n)}$-size adversaries.

As the hash functions admit a uniform family of skew circuits, the following is true: There is a (uniform) circuit C such that $G(x,z) = C(x, f(x), z)$, and C is a size-$O(n)$ skew circuit w.r.t. the z argument. Let $\tilde{C}$ be the randomized encoding of C and define $\tilde{G}(x,z,r) = \tilde{C}(x, f(x), z, r)$

where $r \in \{0,1\}^{|C|}$. Then by Theorem 5.12, the function $\tilde{G} : \{0,1\}^{n_2} \to \{0,1\}^{n_2+\alpha}$ is also a condEP-PRG. Here, since the size of C is $O(n)$, we have $n_2 = O(n)$. Further note that $\tilde{G}$ is computable in DLOGTIME, hence every output of $\tilde{G}$ has non-trivial KT complexity. Since $\tilde{G}$ is 4ε-indistinguishable from the uniform distribution by $2^{\Omega(n)}$-size adversaries, we can see that KT is hard on average. □

5.5 The Perebor Hypotheses

We mention some Perebor hypotheses as further research directions. Each hypothesis states that to some extent, "Perebor," or brute-force search, is unavoidable to solve a certain meta-complexity problem. In this paper, we only consider the (bounded-error) average-case complexity of these problems, but similar hypotheses for the worst-case or zero-error average-case complexity can also be made. We only state these hypotheses against (uniform) randomized algorithms; the corresponding hypotheses against non-uniform algorithms (i.e., circuits) will be called "non-uniform Perebor hypotheses" accordingly.

These hypotheses are inspired by, and parallel to, the "exponential time hypotheses" for satisfiability [59, 60, 22]. The *exponential time hypothesis* (ETH) asserts that 3-SAT requires $2^{\varepsilon n}$ time to solve, where $\varepsilon > 0$ is some absolute constant and n is the number of variables. The *strong exponential time hypothesis* (SETH) asserts that for *any* constant $\varepsilon > 0$, CNF-SAT requires $2^{(1-\varepsilon)n}$ time to solve. There is a large body of work on these two hypotheses and their variants; in particular, SETH has been a central hypothesis in fine-grained complexity [97].

We believe that the future study of these Perebor hypotheses will bring us more insights into complexity theory, similar to what the study of ETH and SETH has brought us.

The weak Perebor hypotheses. We introduce the following two hypotheses for K^t and KT:

Hypothesis 5.17 (Weak Perebor Hypothesis for K^t)**.** There are a polynomial $t(n) \geq 2n$ and an absolute constant $c \geq 1$ such that the following holds. Every randomized algorithm that runs in $2^{n/c}$ time and attempts to solve K^t fails w.p. at least $1/n^c$ over a uniformly random input.

Hypothesis 5.18 (Weak Perebor Hypothesis for KT)**.** There is an absolute constant $c \geq 1$ such that the following holds. Every randomized algorithm that runs in $2^{n/c}$ time and attempts to solve KT fails w.p. at least $1/n^c$ over a uniformly random input.

Theorem 5.2 shows that the non-uniform version of Hypothesis 5.17 is equivalent to the existence of exponentially-hard weak one-way functions (against non-uniform adversaries). Theorem 5.3 shows that the non-uniform version of Hypothesis 5.18 is implied by the existence of exponentially-hard weak one-way function computable in DLOGTIME (also against non-uniform adversaries).

The strong Perebor hypotheses. We start with the following hypothesis:

Hypothesis 5.19 (Strong Perebor Hypothesis for K^t)**.** There are polynomials $t(n) \geq 2n$ and $p(n)$ such that for every constant $\varepsilon > 0$, every probabilistic algorithm that runs in $2^{(1-\varepsilon)n}$ time and attempts to solve K^t fails w.p. at least $1/p(n)$ over a uniformly random input.

By Theorem 5.2, the non-uniform version of Hypothesis 5.19 is equivalent to the existence of weak one-way functions with hardness $2^{(1-o(1))n}$ (against non-uniform adversaries).

However, Building on Hellman [43], Fiat and Naor [30] showed that no such one-way function exists in the *non-uniform RAM* model. In particular, for *any* function $f : \{0,1\}^n \to \{0,1\}^n$, there is an algorithm that runs in $2^{3n/4}$ time, uses $2^{3n/4}$ bits of advice, and inverts f at *any* point. It is conceivable that a similar attack could also be implemented in circuits, i.e. every function f could be inverted by a circuit of size $2^{99n/100}$ in the worst-case. This gives strong evidence that the non-uniform version of Hypothesis 5.19 is false. To the best of our knowledge, (the uniform version of) Hypothesis 5.19 seems secure.

Following [17, Section 1.1], if we still want (non-uniform) maximum hardness, we can consider *collections* of one-way functions, which corresponds to the *conditional* (time-bounded) Kolmogorov complexity. We believe that this is the correct way to formulate the non-uniform version of strong Perebor hypothesis.

Fix a universal Turing machine U, let x, y be two strings and t be a time bound. Define $\mathrm{cK}^t(x \mid y)$ as the length of the smallest description d such that for every $1 \le i \le |x| + 1$ and $b \in \{0, 1, \star\}$, $U^{d,y}(i, b)$ accepts in time t if and only if $x_i = b$. Note that the universal Turing machine is given random access to y (for free), hence d is a description of x *conditioned on* y. We assume that the default input distribution of cK^t consists of a random string x and a random string y, both of input length n. Hence the hypothesis below actually states that no non-uniform algorithm of $2^{(1-\varepsilon)n}$ size can solve cK^t on input length $2n$.

Hypothesis 5.20 (Strong Perebor Hypothesis for cK^t; Non-uniform Version)**.** There are polynomials $t(n) \ge 2n$ and $p(n)$ such that for every constant $\varepsilon > 0$, every non-uniform algorithm of $2^{(1-\varepsilon)n}$ size that attempts to solve cK^t fails on a $1/p(n)$ fraction of inputs.

5.6 Proof of Theorem 5.9

Theorem 5.9. *Let $\varepsilon, \delta, \alpha, f$ be defined as in Construction 5.8. If $\varepsilon \ge 1/\mathrm{poly}(n)$ and $L \le \mathrm{poly}(n)$, then there is a function $r : \mathbb{N} \to \mathbb{N}$ such that $G = \{G_{n,r(n)}\}_{n\in\mathbb{N}}$ is a condEP-PRG with stretch α and security 4ε.*

More precisely, let $\tilde{n} = n + 2d + d'$. Suppose that for every subset $\mathcal{D} \subseteq \{0,1\}^{\tilde{n}}$ such that $\mathrm{H}(G(\mathcal{D})) \ge \tilde{n} - \Omega(\log(\frac{n}{\delta\varepsilon}))$ and every k, there is an adversary of size s that 4ε-distinguishes $G_{n,k}(\mathcal{D})$ from the uniform random distribution. Then there is an adversary of size $s \cdot \mathrm{poly}(nL/\varepsilon)$ that inverts f on a $1 - \delta$ fraction of inputs.

For convenience, we only consider non-uniform adversaries in this section. (See also Remark 5.4.) Recall that we sometimes use a (multi-)set $\mathcal{S}$ to represent the uniform distribution over $\mathcal{S}$ and we assume that every one-way function is length-preserving.

5.6.1 Impagliazzo's Hardcore Lemma

Lemma 5.21 ([56])**.** *Let f be a (candidate weak one-way) function computable in size $s_f(n)$ and $\varepsilon, \delta > 0$. Suppose for every $\mathcal{E} \subseteq \{0,1\}^n$ with $|\mathcal{E}| \ge \delta \cdot 2^n$, there is a circuit C of size $s(n)$ such that*

$$\Pr_{\mathbf{x}\leftarrow\mathcal{E}}[C(f(\mathbf{x})) \in f^{-1}(f(\mathbf{x}))] \ge \varepsilon.$$

Then there is a circuit of size $O((s(n)+s_f(n))\cdot n\varepsilon^{-1})$ that inverts f on a $1-\delta$ fraction of inputs.

Proof Sketch. In what follows, for a circuit C and an input $x \in \{0,1\}^n$, we say that C *inverts* $f(x)$ if $C(f(x)) \in f^{-1}(f(x))$. Consider a zero-sum game between two players called Inverter and Adversary, where Inverter chooses a subset $\mathcal{E}$ of size at least $\delta \cdot 2^n$, Adversary chooses a circuit C of size $s(n)$, and the amount of money that Inverter pays to Adversary is equal to $\Pr_{\mathbf{x}\leftarrow\mathcal{E}}[C \text{ inverts } \mathbf{x}]$. Our assumption implies that Adversary can always guarantee a payoff of at least ε. By the Minimax theorem, there is a distribution $\mathcal{C}$ of size-$s(n)$ circuits such that no matter which subset $\mathcal{E}$ Inverter chooses, we always have

$$\Pr_{\mathbf{C}\leftarrow\mathcal{C},\mathbf{x}\leftarrow\mathcal{E}}[\mathbf{C} \text{ inverts } \mathbf{x}] \geq \varepsilon. \tag{5.1}$$

Let Bad denote the set of $x \in \{0,1\}^n$ such that $\Pr_{\mathbf{C}\leftarrow\mathcal{C}}[\mathbf{C} \text{ inverts } x] < \varepsilon$. We have $|\mathsf{Bad}| < \delta \cdot 2^n$; otherwise Inverter can take $\mathcal{E}$ to be the uniform distribution over Bad and contradict (5.1).

Consider the following distribution $\mathcal{A}$ of adversaries. Independently sample $t := O(n/\varepsilon)$ circuits $A_1, A_2, \ldots, A_t$ from $\mathcal{C}$. Given $y = f(x)$ as the input where $x \in \{0,1\}^n$, $\mathcal{A}$ uses each A_i to invert y and outputs a preimage of y if any of A_i finds one. For every fixed $x \notin \mathsf{Bad}$, the probability that $\mathcal{A}$ fails to invert x is at most $(1-\varepsilon)^t \leq 4^{-n}$, hence a union bound over every $x \notin \mathsf{Bad}$ shows that with high probability, a random circuit in $\mathcal{A}$ inverts f on every $x \notin \mathsf{Bad}$, which is a $1-\delta$ fraction of inputs. Every circuit in the support of $\mathcal{A}$ is of size $O(s(n)+s_f(n))\cdot t \leq O((s(n)+s_f(n))n\varepsilon^{-1})$, which completes the proof. □

5.6.2 Step I: Making f Strong and Regular

Let f be a weak one-way function. The first step is to transform f into a strong and regular one-way function, but only under a certain input distribution. In particular, we will define a sequence of subsets $\mathcal{X} = \{\mathcal{X}_n\}$ (that is not necessarily easy to sample) such that on the uniform distribution over $\mathcal{X}_n$, f is both *strong* and *regular*. Here:

- We say f is *strong* on $\mathcal{X}$, if every polynomial-size adversary $\mathcal{A}$ fails to invert $f(\mathcal{X})$ except with negligible probability. (For comparison, we are only given that f is a *weak* one-way function on a uniform random input: No PPT adversary inverts f on *a* $(1-1/n^c)$ *fraction of inputs*, for some fixed constant $c > 0$.)
- For a function $r : \mathbb{N} \to \mathbb{N}$, we say f is *r-regular* on $\mathcal{X}$, if for every $n \in \mathbb{N}$ and every $y \in f_n(\mathcal{X}_n)$, we have $|f_{\mathcal{X}}^{-1}(y)| \in [2^{r(n)-1}, 2^{r(n)}]$. Here, $f_{\mathcal{X}}^{-1}(y) = \{x \in \mathcal{X} : f(x) = y\}$, and $|f_{\mathcal{X}}^{-1}(y)|$ denotes the size of the above set.

As discussed in Section 5.2, we use the hardcore lemma to find a subset of inputs on which f is strong. In particular, applying Lemma 5.21, we have:

Claim 5.22. *There is a sequence of subsets $\mathcal{X}' = \{\mathcal{X}'_n \subseteq \{0,1\}^n\}$ with $|\mathcal{X}'_n| \geq 2^n/\mathrm{poly}(n)$ such that for every polynomial-size adversary $\mathcal{A}$,*

$$\Pr_{\mathbf{x}\leftarrow\mathcal{X}'_n}[\mathcal{A}(f(\mathbf{x})) \in f^{-1}(f(\mathbf{x}))] < \mathrm{negl}(n).$$

More precisely, suppose that for every subset $\mathcal{X}'_n \subseteq \{0,1\}^n$ with $|\mathcal{X}'_n| \geq \frac{\delta}{2} \cdot 2^n$, there is an adversary of size s that inverts $f_n(\mathcal{X}'_n)$ w.p. at least ε. Then there is an adversary of size $O(s \cdot n\varepsilon^{-2})$ that inverts f on a $1-\delta$ fraction of inputs. □

Now, for every string $y \in \{0,1\}^n$, let $|f^{-1}_{\mathcal{X}'}(y)|$ denote the number of inputs $x \in \mathcal{X}'_n$ such that $f(x) = y$. For $r \in [1,n]$, let W_r be the number of strings $x \in \mathcal{X}'_n$ such that $|f^{-1}_{\mathcal{X}'}(f(x))| \in [2^{r-1}, 2^r]$. Then we have $\sum_{r=1}^{n} W_r \geq |\mathcal{X}'_n|$. By averaging, there is an integer $r \in [1,n]$ such that $W_r \geq |\mathcal{X}'_n|/n$. We define $r(n)$ to be this integer r, and

$$\mathcal{X}_n := \{x \in \mathcal{X}'_n : |f^{-1}_{\mathcal{X}'}(f(x))| \in [2^{r-1}, 2^r]\}.$$

By definition, f is r-regular on $\mathcal{X} := \{\mathcal{X}_n\}$. Since $|\mathcal{X}_n| \geq |\mathcal{X}'_n|/n$, any adversary that inverts f on an ε fraction of $\mathcal{X}_n$ also inverts f on an ε/n fraction of $\mathcal{X}'_n$. To summarize:

Claim 5.23. *There is a function $r(n) \leq n$ and a sequence of subsets $\mathcal{X} = \{\mathcal{X}_n\}$ with $|\mathcal{X}_n| \geq 2^n/\mathrm{poly}(n)$ such that f is r-regular on $\mathcal{X}$ and that for every polynomial-size adversary $\mathcal{A}$,*

$$\Pr_{\mathbf{x} \leftarrow \mathcal{X}_n}[\mathcal{A}(f(\mathbf{x})) \in f^{-1}(f(\mathbf{x}))] < \mathrm{negl}(n).$$

More precisely, suppose that for every function $r : \mathbb{N} \to \mathbb{N}$ and sequence of subsets $\mathcal{X} = \{\mathcal{X}_n\}$ such that $|\mathcal{X}_n| \geq \frac{\delta}{2n} \cdot 2^n$ and f is r-regular on $\mathcal{X}$, there is an adversary of size s that inverts $f_n(\mathcal{X}_n)$ w.p. at least ε. Then there is an adversary of size $s \cdot \mathrm{poly}(n/\varepsilon)$ that inverts f on a $1-\delta$ fraction of inputs. □

5.6.3 Step II: An Intermediate Function

We define another function ensemble $\tilde{f} = \{\tilde{f}_n\}_{n \in \mathbb{N}}$. Let $k_1 = r - 1$, $k_2 = \lfloor n - r - \log(2n/\delta) \rfloor$, and $d = d_{\mathsf{Ext}}(n, \varepsilon)$. We need the following two extractors:

- a strong (k_1, ε)-extractor $\mathsf{Ext}_1 : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^{m_1}$, where $m_1 := k_1 - 2\log(1/\varepsilon) - O(1)$;
- a strong (k_2, ε)-extractor $\mathsf{Ext}_2 : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^{m_2}$, where $m_2 := k_2 - 2\log(1/\varepsilon) - O(1)$.

The function $\tilde{f}_n : \{0,1\}^n \times \{0,1\}^{2d} \to \{0,1\}^{m_1+m_2+2d}$ is defined as follows.

$$\tilde{f}_n(x, z_1, z_2) := z_1 \circ \mathsf{Ext}_1(x, z_1) \circ z_2 \circ \mathsf{Ext}_2(f(x), z_2).$$

Denote $\ell(n) := m_1 + m_2 + 2d$. The following lemma summarizes the properties of $\tilde{f}_n$ we need:

Lemma 5.24. *For every integer n, the function $\tilde{f}_n$ satisfies the following properties:*

(Uniformity) For every integer n, $\mathsf{SD}(\tilde{f}_n(\mathcal{X}_n, \mathcal{U}_{2d}), \mathcal{U}_{\ell(n)}) \leq 2\varepsilon$.

(Hiding) For every polynomial-size adversary $\mathcal{A}$ and every integer n,

$$\Pr_{\mathbf{x}\leftarrow\mathcal{X}_n}\left[\mathcal{A}(\tilde{f}_n(\mathbf{x},\mathcal{U}_{2d}))=\mathbf{x}\right]\le \mathrm{negl}(n).$$

More precisely, if there is an adversary $\mathcal{A}$ of size s that on input $\tilde{f}_n(\mathcal{X}_n,\mathcal{U}_{2d})$ guesses $\mathcal{X}_n$ with success probability γ, then there is an adversary $\mathcal{A}'$ of size $O(s)$ that inverts $f_n(\mathcal{X}_n)$ w.p. at least $\Omega(\gamma/\varepsilon^2)$.

Proof. (Uniformity) A sample from $\mathcal{X}_n$ can be obtained from two steps. First, we sample a string y_0 with probability $p(y_0) := \Pr_{\mathbf{x}\leftarrow\mathcal{X}_n}[f_n(\mathbf{x})=y_0]$. Then we sample a string x_0 with probability $p(x_0 \mid y_0) := \Pr_{\mathbf{x}\leftarrow\mathcal{X}_n}[\mathbf{x}=x_0 \mid f_n(\mathbf{x})=y_0]$.

Suppose y_0 is fixed. Since f is r-regular, we have $|f_{\mathcal{X}}^{-1}(y_0)|\ge 2^{r-1}$. Therefore, conditioned on y_0, the min-entropy of the distribution of x_0 is at least $r-1\ge k_1$.

Let $\mathbf{x}\leftarrow\mathcal{X}_n$ and $\mathbf{z}_1\leftarrow\mathcal{U}_d$. Since Ext_1 is a strong (k_1,ε)-extractor, we have

$$\mathsf{SD}(\mathbf{z}_1\circ\mathsf{Ext}_1(\mathbf{x},\mathbf{z}_1),\mathcal{U}_{d+m_1}\mid f(\mathbf{x}))\le\varepsilon.$$

(Here, the LHS denotes the statistical distance between $\mathbf{z}_1\circ\mathsf{Ext}_1(\mathbf{x},\mathbf{z}_1)$ and $\mathcal{U}_{d+m_1}$ *conditioned on* the value $f(\mathbf{x})$.)

Now, for every $y_0\in f_n(\mathcal{X}_n)$, since $|f_{\mathcal{X}}^{-1}(y_0)|\le 2^r$, the probability that a sample of $f_n(\mathcal{X}_n)$ is equal to the particular y_0 is at most $2^r/|\mathcal{X}_n|$. It follows that the min-entropy of the distribution of y_0 is at least $\log(|\mathcal{X}_n|/2^r)\ge n-r+\log(\delta/2n)\ge k_2$. Since Ext_2 is a strong (k_2,ε)-extractor, we have

$$\mathsf{SD}(\mathbf{z}_2\circ\mathsf{Ext}_2(f(\mathbf{x}),\mathbf{z}_2),\mathcal{U}_{d+m_2})\le\varepsilon.$$

It follows that

$$\begin{aligned}
&\mathsf{SD}(\mathbf{z}_1\circ\mathsf{Ext}_1(\mathbf{x},\mathbf{z}_1)\circ\mathbf{z}_2\circ\mathsf{Ext}_2(f(\mathbf{x}),\mathbf{z}_2),\mathcal{U}_{\ell(n)})\\
\le\ &\mathsf{SD}(\mathbf{z}_1\circ\mathsf{Ext}_1(\mathbf{x},\mathbf{z}_1)\circ\mathbf{z}_2\circ\mathsf{Ext}_2(f(\mathbf{x}),\mathbf{z}_2),\mathcal{U}_{d+m_1}\circ\mathbf{z}_2\circ\mathsf{Ext}_2(f(\mathbf{x}),\mathbf{z}_2))\\
+\ &\mathsf{SD}(\mathcal{U}_{d+m_1}\circ\mathbf{z}_2\circ\mathsf{Ext}_2(f(\mathbf{x}),\mathbf{z}_2),\mathcal{U}_{\ell(n)})\\
\le\ &\varepsilon+\varepsilon=2\varepsilon.
\end{aligned}$$

(Hiding) Let $\mathcal{A}$ be any adversary that violates the Hiding property. Suppose that

$$\Pr_{\mathbf{x}\leftarrow\mathcal{X}_n}[\mathcal{A}(\tilde{f}_n(\mathbf{x},\mathcal{U}_{2d}))=\mathbf{x}]\ge\gamma.$$

We will use $\mathcal{A}$ to build an algorithm $\mathcal{A}'$ that inverts $f(\mathcal{X}_n)$ w.p. $O(\gamma/\varepsilon^2)$.

Let $\mathbf{x}\leftarrow\mathcal{X}_n$ be a hidden string and $\mathbf{y}=f_n(\mathbf{x})$ be the input of $\mathcal{A}'$. We sample $\mathbf{z}_1,\mathbf{z}_2\leftarrow\mathcal{U}_d$. We also "guess" a string $\mathbf{z}\leftarrow\mathcal{U}_{m_1}$, with the hope that $\mathsf{Ext}_1(\mathbf{x},\mathbf{z}_1)=\mathbf{z}$. Then we output $\mathcal{A}'(\mathbf{y}) := \mathcal{A}(\mathbf{z}_1\circ\mathbf{z}\circ\mathbf{z}_2\circ\mathsf{Ext}_2(\mathbf{y},\mathbf{z}_2))$.

Conditioned on $\mathsf{Ext}_1(\mathbf{x},\mathbf{z}_1)=\mathbf{z}$, the distribution of $\mathbf{z}_1\circ\mathbf{z}\circ\mathbf{z}_2\circ\mathsf{Ext}_2(\mathbf{y},\mathbf{z}_2)$ is exactly $\tilde{f}_n(\mathcal{X}_n,\mathcal{U}_{2d})$. Therefore,

$$\Pr[\mathcal{A}'(\mathbf{y})=\mathbf{x}]\ge\gamma\cdot\Pr[\mathsf{Ext}_1(\mathbf{x},\mathbf{z}_1)=\mathbf{z}]=\gamma\cdot 2^{-m_1}.$$

Note that besides $\mathbf{y} = f(\mathbf{x})$, $\mathcal{A}'$ does not know any information about $\mathbf{x}$. Therefore, for every $x' \in f^{-1}(\mathbf{y})$, the probability that $\mathcal{A}'(\mathbf{y}) = x'$ should also be at least $\gamma \cdot 2^{-m_1}$. We have

$$\begin{aligned}\Pr_{\mathbf{y}=f(\mathcal{X}_n)}[\mathcal{A}'(\mathbf{y}) \in f^{-1}(\mathbf{y})] &\geq \gamma \cdot 2^{-m_1} \cdot |f_{\mathcal{X}}^{-1}(\mathbf{y})| \\ &\geq \gamma \cdot 2^{r-1-m_1} = O(\gamma/\varepsilon^2).\end{aligned}$$

□

5.6.4 Step III: Appending a Hardcore Function

Note that the output length of $\tilde{f}_n$ is $\tau := \log(n/\delta) + 4\log\frac{1}{\varepsilon} + O(1)$ bits shorter than the input length of $\tilde{f}_n$. In this section, we append a hardcore function at the end of $\tilde{f}_n$, making it a pseudorandom generator with stretch $\alpha > 0$. In particular, we need:

- a hardcore function $\mathsf{HC} : \{0,1\}^n \times \{0,1\}^{d'} \to \{0,1\}^k$ with distinguishing probability ε, where $k := \tau + \alpha$, and $d' := d_{\mathsf{HC}}(n, k, \varepsilon)$. Let R be the reconstruction algorithm of this hardcore function, and $L := L(n, k, \varepsilon)$ be the list size.

Let $\tilde{n} := n + 2d + d'$. Recall that $G_{n,r} : \{0,1\}^{\tilde{n}} \to \{0,1\}^{\tilde{n}+\alpha}$ is defined as

$$G_{n,r}(x, z_1, z_2, z_3) := z_1 \circ \mathsf{Ext}_1(x, z_1) \circ z_2 \circ \mathsf{Ext}_2(f(x), z_2) \circ z_3 \circ \mathsf{HC}(x, z_3).$$

Let $\mathcal{E}_{\tilde{n}} := \mathcal{X}_n \times \{0,1\}^{2d+d'}$. In other words, a uniform random string from $\mathcal{E}_{\tilde{n}}$ can be sampled as $\mathbf{x} \circ \mathbf{z}$ where $\mathbf{x} \leftarrow \mathcal{X}_n$ and $\mathbf{z} \leftarrow \mathcal{U}_{2d+d'}$. We will show that $G_{n,r}$ is a condEP-PRG whose "condition" is $\mathcal{E}_{\tilde{n}}$. In particular, Lemma 5.25 shows that $G_{n,r}(\mathcal{E}_{\tilde{n}})$ is pseudorandom, and Lemma 5.26 shows that $G_{n,r}(\mathcal{E}_{\tilde{n}})$ is entropy-preserving.

Lemma 5.25. *Every polynomial-size adversary $\mathcal{A}$ fails to 4ε-distinguish $G_{n,r}(\mathcal{E}_{\tilde{n}})$ from $\mathcal{U}_{\ell(n)+d'+k}$.*

More precisely, if there is an adversary $\mathcal{A}$ of size s that 4ε-distinguishes $G_{n,r}(\mathcal{E}_{\tilde{n}})$ from $\mathcal{U}_{\ell(n)+d'+k}$, then there is an adversary $\mathcal{A}'$ of size $s \cdot \mathrm{poly}(n/\varepsilon)$ that on input $\tilde{f}_n(\mathcal{X}_n, \mathcal{U}_{2d})$, guesses $\mathcal{X}_n$ with success probability $\varepsilon/2L$.

Proof. Suppose $\mathcal{A}$ is an adversary that 4ε-distinguishes $G_{n,r}(\mathcal{E}_{\tilde{n}})$ from $\mathcal{U}_{\ell(n)+d'+k}$.

Since $\mathsf{SD}(\tilde{f}_n(\mathcal{X}_n, \mathcal{U}_{2d}), \mathcal{U}_{\ell(n)}) \leq 2\varepsilon$, it must be the case that $\mathcal{A}$ could 2ε-distinguish $G_{n,r}(\mathcal{E}_{\tilde{n}})$ from $\tilde{f}_n(\mathcal{X}_n, \mathcal{U}_{2d}) \circ \mathcal{U}_{d'+k}$. Equivalently, let $\mathbf{x} \leftarrow \mathcal{X}_n$, then given the information of $\tilde{f}(\mathbf{x}, \mathcal{U}_{2d})$, $\mathcal{A}$ could 2ε-distinguish $\mathcal{U}_{d'} \circ \mathsf{HC}(\mathbf{x}, \mathcal{U}_{d'})$ from $\mathcal{U}_{d'+k}$. We say a string $w := (x, z_1, z_2)$ is *good* if $\mathcal{A}$ could ε-distinguish $\tilde{f}_n(w) \circ \mathcal{U}_{d'} \circ \mathsf{HC}(x, \mathcal{U}_{d'})$ from $\tilde{f}_n(w) \circ \mathcal{U}_{d'+k}$. Then by a Markov bound, a random $\mathbf{w} \leftarrow \mathcal{X}_n \circ \mathcal{U}_{2d}$ is good w.p. at least ε.

The adversary $\mathcal{A}'$ that violates the (Hiding) property of $\tilde{f}_n$ simply follows from the reconstruction algorithm R. In particular, on input $\tilde{f}_n(w) = \tilde{f}_n(x, z_1, z_2)$, $\mathcal{A}'$ constructs the following oracle:

$$O(r) := \mathcal{A}(\tilde{f}_n(w) \circ r),$$

runs the algorithm R^O to obtain a list of size L, and outputs a random element in the list.

We analyze $\mathcal{A}'$. Suppose $\mathcal{A}'$ is given $\tilde{f}_n(w)$ for a good w, then O indeed ε-distinguishes $\mathcal{U}_{d'} \circ \mathsf{HC}(x, \mathcal{U}_{d'})$ from $\mathcal{U}_{d'+k}$. Therefore, w.p. $\geq 1/2$, x is in the list outputted by R^O. If this is the

case, we will correctly output x w.p. $\geq 1/L$. It follows that on input $\tilde{f}_n(\mathbf{x}, \mathcal{U}_{2d})$ where $\mathbf{x} \leftarrow \mathcal{X}_n$, $\mathcal{A}'$ outputs $\mathbf{x}$ w.p. $\geq \varepsilon/2L$. Finally, as R is a polynomial-size oracle circuit (actually a PPT oracle machine), the size of $\mathcal{A}'$ is $s(n) \cdot \mathrm{poly}(n/\varepsilon)$. □

Lemma 5.26. *Suppose that* $\varepsilon < \frac{1}{10n^2}$. *Then* $\mathrm{H}(G_{n,r}(\mathcal{E}_{\tilde{n}})) \geq \tilde{n} - \tau - 2$.

Proof. Since $\mathsf{SD}(\tilde{f}_n(\mathcal{X}_n, \mathcal{U}_{2d}), \mathcal{U}_{\ell(n)}) \leq 2\varepsilon < \frac{1}{\ell(n)^2}$, by [70, Lemma 2.2], we have $\mathrm{H}(\tilde{f}_n(\mathcal{X}_n, \mathcal{U}_{2d})) \geq \ell(n) - 2$. It follows that $\mathrm{H}(G_{n,r}(\mathcal{E}_{\tilde{n}})) \geq (\ell(n) - 2) + d' \geq \tilde{n} - \tau - 2$. □

5.6.5 Putting It Together

Proof of Theorem 5.9. Suppose that for every $\mathcal{X} = \{\mathcal{X}_n\}$ that satisfies the premise of Claim 5.23, and $\mathcal{E}_{\tilde{n}}$ defined above, there is an adversary of size $s(n)$ that 4ε-distinguishes $G_{n,r}(\mathcal{E}_{\tilde{n}})$ from the uniform distribution. Then:

- By Lemma 5.25, there is an adversary of size $s(n) \cdot \mathrm{poly}(n/\varepsilon)$ that on input $\tilde{f}(\mathcal{X}_n, \mathcal{U}_{d_1+d_2})$, guesses $\mathcal{X}_n$ w.p. $\geq \varepsilon/2L$.
- By Lemma 5.24, there is an adversary of size $s(n) \cdot \mathrm{poly}(n/\varepsilon)$ that inverts $f_n(\mathcal{X}_n)$ w.p. $\geq \frac{1}{2\varepsilon L}$.

It follows from Claim 5.23 that there is an adversary of size $s \cdot \mathrm{poly}(nL/\varepsilon)$ that inverts f on a $1 - \delta$ fraction of inputs. □

5.7 Proof of Theorem 5.15

In this section, we briefly review the universal hash functions in [63] that are computable by linear-size circuits, with an emphasis on the uniformity of these circuits. Throughout this section, a circuit family is uniform if it satisfies Definition 5.13. An XOR-circuit is a (multi-output) circuit that only uses XOR gates of fan-in 2. To match Definition 5.13, we also require that every gate in an XOR-circuit has fan-out at most 2.

Outline. Our start point is the strongly explicit family of expanders by [72, 33]. Spielman [88] showed that these expanders imply asymptotically optimal error-correcting codes (i.e., with constant rate and constant relative distance). Using an expander walk trick, for any constant $\varepsilon > 0$, one could construct error-correcting codes with relative distance $1 - \varepsilon$, constant rate, and constant alphabet size. By the construction of [63], such codes imply universal hash functions.

5.7.1 Strongly Explicit Expanders

We use the following construction due to [72, 33]. (See also [52, Construction 8.1].) For every integer n, we have a graph $\mathcal{G}_n$ with n^2 vertices such that every vertex has degree 8. The vertex set of $\mathcal{G}_n$ is $\mathbb{Z}_n \times \mathbb{Z}_n$. Each vertex $v = (x, y)$ is adjacent to the following vertices

$$\gamma_1(v) = (x + 2y, y), \gamma_2(v) = (x + 2y + 1, y), \gamma_3(v) = (x, y + 2x), \gamma_4(v) = (x, y + 2x + 1).$$

Here the additions are modulo n. Note that $\gamma_1, \ldots, \gamma_4$ are bijections and the other four neighbors of v are simply $\gamma_1^{-1}(v), \ldots, \gamma_4^{-1}(v)$. The graph might contain self-loops or parallel edges.

Theorem 5.27 ([33]). *For every integer $n \geq 1$, the second largest eigenvalue of the adjacency matrix of $\mathcal{G}_n$ is at most $5\sqrt{2} < 8$.*

In our construction, we need the degree of the expanders to be a large enough constant. We can simply pick a large enough constant k and take the k-th power of $\mathcal{G}_n$. Let $\mathcal{G}_n^k$ be the k-th power of $\mathcal{G}_n$, i.e., for every $u, v \in V(\mathcal{G}_n)$, the number of (parallel) edges between u and v in $\mathcal{G}_n^k$ is equal to the number of length-k paths between u and v in $\mathcal{G}_n$. Then the degree of $\mathcal{G}_n^k$ is $d := 8^k$, and the second largest eigenvalue of the adjacency matrix of $\mathcal{G}_n^k$ is at most $(5\sqrt{2})^k < d$.

Remark 5.28. It will be convenient to define an explicit mapping (bijection) between $E(\mathcal{G}_n^k)$ and $[dn^2/2]$. Note that each edge $(u, v) \in \mathcal{G}_n^k$ can be represented by a start vertex u and a string $\sigma_1\sigma_2\ldots\sigma_k$ where each $\sigma_i \in \Sigma := \{\gamma_1, \gamma_2, \gamma_3, \gamma_4, \gamma_1^{-1}, \gamma_2^{-1}, \gamma_3^{-1}, \gamma_4^{-1}\}$. The meaning of this representation is that $(\sigma_1 \circ \sigma_2 \circ \cdots \circ \sigma_k)(u) = v$. Each edge has two representations: $(u, \sigma_1\sigma_2\ldots\sigma_k)$ or $(v, \sigma_k^{-1}\sigma_{k-1}^{-1}\ldots\sigma_1^{-1})$. We arbitrarily choose a size-$(d/2)$ subset S of Σ^k such that for each $\sigma_1, \sigma_2, \ldots, \sigma_k \in \Sigma$, exactly one of $\sigma_1\sigma_2\ldots\sigma_k$ and $\sigma_k^{-1}\sigma_{k-1}^{-1}\ldots\sigma_1^{-1}$ is in S. We fix and hardcode a bijection between $[d/2]$ and S. Given an integer $i \in [dn^2/2]$, we interpret i as a pair of $v \in V(\mathcal{G}_n^k)$ and $\sigma_1\sigma_2\ldots\sigma_k \in S$, and the edge corresponding to i is represented as $(v, \sigma_1\sigma_2\ldots\sigma_k)$. This bijection and its inverse are computable in time $O(\log n)$.

5.7.2 Error-Reduction Codes

An intermediate step in [88] is to construct *error-reduction codes*, which are weaker primitives compared to error-correcting codes.

Let $r, \delta, \varepsilon > 0$ be constants. An error-reduction code of rate r, error reduction ε, and reducible distance δ is a function $C : \{0,1\}^{rn} \to \{0,1\}^{(1-r)n}$ mapping rn "message" bits into $(1-r)n$ "check" bits such that the following holds. The codeword of a message x is $x \circ C(x)$. For any message x, if we are given a corrupted codeword that differs from $x \circ C(x)$ with at most $v \leq \delta n$ message bits and at most $t \leq \delta n$ check bits, then we can recover a codeword that differs from $x \circ C(x)$ in at most εt message bits. (We will not be particularly interested in the complexity of recovery or decoding algorithms.)

We need a family of error-reduction codes $\mathcal{R}_n : \{0,1\}^n \to \{0,1\}^{\lfloor n/2 \rfloor}$ (that is, with rate 2/3). The following construction appears in [88, Section V] and we verify its uniformity:

Lemma 5.29. *For some absolute constant $\delta > 0$, there is a family of error-reduction codes $\mathcal{R} = \{\mathcal{R}_n : \{0,1\}^n \to \{0,1\}^{\lfloor n/2 \rfloor}\}$ with error-reduction $1/2$ and reducible distance δ. Moreover, the sequence of functions $\{\mathcal{R}_n\}$ can be computed by a uniform family of linear-size* XOR*-circuits.*

Proof Sketch. Let ε and r be the constants fixed in the proof of [88, Theorem 19], and $d := 8^k$ for some large enough constant k. Let m be the smallest integer such that $dm^2/2 \geq n$. Note that m can be computed in $O(\log n)$ time. Since $r > 4/5$, we have that $dm^2(1-r)/r \leq n/2$. It suffices to construct an error-reduction code with $dm^2/2$ message bits and $dm^2(1-r)/r$ check bits.

The error-reduction code will be of the form $\mathcal{R}(B, \mathcal{S})$ (as in [88, Definition 16]) with B being the edge-vertex incidence graph of $\mathcal{G}_m^k$ and $\mathcal{S}$ being a good (linear) error-correcting code on d-bit messages that has rate r. (Since d is a constant, we can hardcode $\mathcal{S}$ in our algorithm. On

the other hand, since d is large enough, $\mathcal{S}$ exists.) That is, we assign a message bit to every edge of $\mathcal{G}_m^k$. For each vertex $v \in V(\mathcal{G}_m^k)$, let $b_1 b_2 \dots b_d$ be the bits on the d incident edges of v. This vertex outputs $d(1-r)/r$ check bits which are the check bits of $\mathcal{S}$ on message $b_1 b_2 \dots b_d$. Concatenating the outputs of each vertex, we obtain an error-reduction code of $dm^2/2$ message bits and $dm^2(1-r)/r$ check bits. Inspecting the proof of [88, Theorem 19], one can see that for some absolute constant $\delta > 0$, this error-reduction code has error-reduction $1/2$ and reducible distance δ.

Computing the i-th gate of the encoding circuit reduces to computing the indices of the incident edges of a vertex $v \in V(\mathcal{G}_m^k)$. By Remark 5.28, this is computable in $O(\log n)$ time. □

5.7.3 Error-Correcting Codes

The construction in [88, Section II] transforms an error-reduction code into an error-correcting code. Here we only review its encoding algorithm and check that they can be implemented by uniform XOR circuits. The correctness of this error-correcting code is proved in [88].

Lemma 5.30. *There is a constant $n_0 > 1$ and a family $C = \left\{C_k : \{0,1\}^{n_0 2^{k-2}} \to \{0,1\}^{n_0 2^k}\right\}$ of error-correcting codes with constant relative distance. Moreover, C can be encoded by a uniform family of linear-size* XOR *circuits.*

Proof Sketch. We recursively define C_k as follows. First, $C_0 : \{0,1\}^{n_0/4} \to \{0,1\}^{n_0}$ is any good enough error-correcting code. Since n_0 is a constant, our algorithm can hardcode C_0.

Now, for $k \geq 1$, we define C_k as follows. Let $x \in \{0,1\}^{n_0 2^{k-2}}$ be the inputs of C_k.

- The first $n_0 2^{k-2}$ outputs of C_k will always be x itself.[19] Note that we require the fan-out of gates to be at most 2. Therefore we need to make a copy of x. Similarly, we may need to copy the A_k, B_k, C_k defined below. The circuit size is still linear in 2^k.
- We pick an error-reduction code $\mathcal{R}_{k-2} : \{0,1\}^{n_0 2^{k-2}} \to \{0,1\}^{n_0 2^{k-3}}$, and output $A_k := \mathcal{R}_{k-2}(x)$.
- Let $C_{k-1} : \{0,1\}^{n_0 2^{k-3}} \to \{0,1\}^{n_0 2^{k-1}}$ be the error-correcting code we recursively defined. Let $A_k \circ B_k := C_{k-1}(A_k)$, and we output B_k. (Recall that the first $n_0 2^{k-3}$ outputs of C_{k-1} is equal to its inputs, i.e., A_k.)
- We pick an error-reduction code $\mathcal{R}_{k-1} : \{0,1\}^{n_0 2^{k-1}} \to \{0,1\}^{n_0 2^{k-2}}$ and output $C_k := \mathcal{R}_{k-1}(A_k \circ B_k)$.

The required error-reduction codes are constructed in Lemma 5.29. The total number of output bits of C_k is $|x| + |A_k| + |B_k| + |C_k|$ which is indeed $n_0 2^k$.

The i-th gate of the encoding circuit of C_{k-1} can be computed as follows. Let $c2^k$ be the circuit complexity of the first, second, and fourth bullet. (That is, circuit complexity of C_k not counting the recursive part for C_{k-1}.) We may assume c is a power of 2. The encoding circuit for

[19] We can assume this is also true for C_0.

C_k has $|C_0| + \sum_{i=1}^{k} c2^i = |C_0| + c(2^{k+1} - 1)$ gates. Taking the (base-2) logarithm of $(i - |C_0|)/c$, we can find the "level of recursion" that the i-th gate is constructed. Then the problem reduces to computing the encoding circuit of $\mathcal{R}_j$ for some integer j, which is computable in $O(\log n)$ time. □

For every constant $\varepsilon > 0$, the construction of [63] needs a code with relative distance $1 - \varepsilon$ and constant alphabet size. As in [7, 39], we can "amplify" the code in Lemma 5.30 by an expander:

Lemma 5.31. *For every constant $\varepsilon > 0$, there is a constant $D > 0$ and a family of error-correcting codes $\{C'_k : \{0,1\}^{n_0 2^{k-2}} \to [2^D]^{O(2^k)}\}$ that has relative distance $1 - \varepsilon$. Moreover, if we interpret $[2^D]$ as length-D strings, then C'_k can be encoded by a uniform family of linear-size* XOR *circuits.*

Proof Sketch. Recall that $\{\mathcal{G}_n\}$ is the expander family constructed in Theorem 5.27, and $\{C_k : \{0,1\}^{n_0 2^{k-2}} \to \{0,1\}^{n_0 2^k}\}$ is the family of error-correcting codes constructed in Lemma 5.30. Let m be the smallest integer such that $m^2 \geq n_0 2^k$. We pad zeros to the outputs of C_k so that C_k can be regarded as a code that outputs m^2 bits. We assign an output bit of C_k to each vertex in $\mathcal{G}_m$. The relative distance of C_k is still lower bounded by an absolute constant $\delta > 0$.

We will pick a large enough constant p such that $\mathcal{G}_m^p$ has good expansion property: Every subset of $V(\mathcal{G}_m^p)$ with size at least $\delta \cdot m^2$ has at least $(1 - \varepsilon)m^2$ neighbors. (See e.g. [7, Corollary 1].) Let $D := 8^p$, then every vertex in $\mathcal{G}_m^p$ has degree D. On input $x \in \{0,1\}^{n_0 2^{k-2}}$, recall that we assigned each vertex in $V(\mathcal{G}_m^p)$ a bit of the codeword $C_k(x)$. For every $v \in V(\mathcal{G}_m^p)$, the vertex v will output the concatenation of the bits assigned to its neighbor, which can be interpreted as an element in $[2^D]$. The code $C'_k(x)$ simply concatenates the outputs of each vertex $v \in V(\mathcal{G}_m^p)$ together.

Consider the encoding circuits of C'_k. As we need each gate to have fanout at most 2, we make D copies of the encoding circuit of C_k. For every $\sigma = \sigma_1 \sigma_2 \ldots \sigma_k \in \Sigma^k$, we have a copy of C_k denoted as C_k^σ. For each vertex v, and each $\sigma \in \Sigma^k$, let u be the σ-th neighbor of v. The σ-th bit of the output of v is the u-th output of the circuit C_k^σ. We can see this encoding circuit is uniform. □

Remark 5.32. Lijie Chen (personal communication) suggested a similar approach based on expander random walks [90, Proposition 6.6]. As the p-th power of an expander graph G consists of length-p walks in G, the two approaches are essentially the same.

5.7.4 Universal Hash Functions

Finally, we are ready to verify that the universal hash functions in [63] are uniform.

Theorem 5.15. *For every integer n, m where $m = O(n)$, there exists an integer $k = O(n)$ and a family of universal hash functions $\{h_{n,m} : \{0,1\}^n \times \{0,1\}^k \to \{0,1\}^m\}$ such that $h_{n,m}$ can be computed by a uniform family of linear-size circuits that are skew w.r.t. the second argument.*

Proof Sketch. Let $n_1 := cn$ for a large enough constant c, ε be a small enough constant, and D be the constant in Lemma 5.31 depending on ε. We need three ingredients:

- An *ℓ-exposure resilient function* (ERF) $\mathsf{ERF} : \{0,1\}^{n_1 D} \to \{0,1\}^m$ [23]. It is shown in [27] that for any (linear) error-correcting code $C : \{0,1\}^m \to \{0,1\}^n$ with generator matrix G and minimum distance d, the transpose matrix G^T mapping n input bits to m output bits is a perfect ℓ-ERF where $\ell := n - d + 1$.

  For an XOR-circuit C that computes the linear transform G over GF(2), we can obtain a circuit computing the linear transform G^T by exchanging the input gates and output gates and reversing the directions of every wire [20, 63]. In particular, every gate $g \in C$ whose output feeds to the gates $g_1, g_2, \ldots, g_k$ becomes, in the new circuit, an XOR gate g whose *inputs* are $g_1, g_2, \ldots, g_k$.

  Therefore, Lemma 5.30 shows that an ℓ-ERF $\mathsf{ERF} : \{0,1\}^{n_1 D} \to \{0,1\}^m$ is computable by a uniform family of linear-size XOR circuits.

- An error-correcting code $C'_k : \{0,1\}^n \to [2^D]^{n_1}$ with relative distance $1-\varepsilon$, as in Lemma 5.31.

- A hash family $H : \{0,1\}^D \times \{0,1\}^{2D-1} \to \{0,1\}^D$, computable by a skew circuit w.r.t. the second argument. As D is a constant, we can simply hardcode this hash family. (See e.g. Section 5.2.1 for an instantiation based on Toeplitz matrices.)

The construction of [63] goes as follows. On input $x \in \{0,1\}^n$, we first compute $C'(x) \in [2^{h+1}]^{n_1}$. Next, we receive n_1 keys $k_1, k_2, \ldots, k_{n_1} \in \{0,1\}^{2h+1}$ which are the keys for our hash function. Let $t \in [2^{h+1}]^{n_1}$ be the following message: $t_i := H(C'(x)_i, k_i)$. We treat t as a string of length $m(h+1)$ and the output of our hash function is $\mathsf{ERF}(t)$.

It is easy to see that this family is uniform. □

6 MCSP-Related Results

In this section, we generalize Theorem 1.1 to the case of MCSP. Throughout this section, we maintain the convention that our input is the truth table of a function $f : \{0,1\}^n \to \{0,1\}$, and $N = 2^n$ is the input length. We use tt to denote an input truth table. Recall that $\mathsf{Size}(tt)$ is the circuit complexity of tt. The size of a circuit is always measured in *gates*. We consider circuits over the B_2 basis, i.e., a gate can compute any function over its 2 inputs.[20]

Theorem 1.3 (Informal)**.** *The following are true:*

- *If* MCSP *is exponentially hard on average, then there is a (super-polynomially hard) one-way function.*

- *If there is an exponentially hard weak one-way function in* NC^0*, then* MCSP *is (exponentially) hard on average.*

[20] Looking ahead, this is because we need Uhlig's theorem (Theorem 6.5) which is only known to hold for the B_2 basis.

Ideally, we would like to prove that MCSP is bounded-error hard on average if and only if there is a one-way function in DLOGTIME. However, we could only prove weaker results, since we do not have good understandings of the circuit complexity of a random Boolean function.

For KT complexity, we know that a random string x of length N is likely to satisfy that $\mathrm{KT}(x) \in [N - O(\log N), N + O(\log N)]$. That is, N is a good estimate of the KT complexity of a random string, within additive error $\eta := O(\log N)$. It turns out that the overhead of [70] is $2^{O(\eta)}$, which is polynomial in N.

What about MCSP? For the maximum circuit complexity function, we only know that:

Theorem 6.1 ([31])**.** *There is a constant c such that the following is true. Let $C(n)$ be the maximum circuit complexity of any function $f : \{0,1\}^n \to \{0,1\}$, then $C_{\mathsf{lb}}(n) \le C(n) \le C_{\mathsf{ub}}(n)$, where*

$$C_{\mathsf{lb}}(n) = \frac{2^n}{n}\left(1 + \frac{\log n}{n} - \frac{c}{n}\right), \text{ and } C_{\mathsf{ub}}(n) = \frac{2^n}{n}\left(1 + \frac{3\log n}{n} + \frac{c}{n}\right).$$

Therefore, given a random truth table tt of length $N = 2^n$, we could use any value between $C_{\mathsf{lb}}(n)$ and $C_{\mathsf{ub}}(n)$ as an estimate of $\mathsf{Size}(tt)$. However, we could only prove that our additive error is $\eta := (C_{\mathsf{ub}}(n) - C_{\mathsf{lb}}(n)) \cdot O(n) = O(2^n \log n/n)$.[21] The overhead in [70] would be

$$2^{O(\eta)} = 2^{O\left(N\frac{\log\log N}{\log N}\right)}.$$

Nevertheless, as $\eta = o(N)$ is non-trivial, we can still achieve non-trivial results for MCSP.

Remark 6.2. Ilango encountered a similar issue in his search-to-decision reduction for MFSP (Minimum Formula Size Problem) [54]. The additive error for formula complexity is $\eta := O(N/\log\log N)$, thus Ilango only managed to show an (average-case) reduction with time complexity $2^{O(\eta)}$ unconditionally.

Comparing [54] and our work, the $2^{O(\eta)}$ factor comes from different reasons. Ilango's algorithm runs in time $\mathrm{poly}(t)$ where t is the number of "near-optimal" formulas for the input truth table; the current best upper bound of t for a random truth table is $2^{O(\eta)}$. In our case, we need to sample a *uniformly random circuit* (w.r.t. some encoding); let p be the probability that the truth table of a sampled circuit is equal to a given one, then the current best lower bound of p is $2^{-N-O(\eta)}$. (See Section 6.2.) It is an interesting open problem to improve either estimate.

6.1 Preliminaries

6.1.1 Extreme Hardness Amplification for One-Way Functions

We will construct a one-way function f_{MCSP} based on the assumption that MCSP is *exponentially* hard on average. However, we are only able to prove that f_{MCSP} is hard to invert on an inverse-sub-exponential fraction ($2^{-o(N)}$) of inputs. We will need the following variant of Theorem 3.12 that constructs a strong one-way function (of super-polynomial hardness) from such a one-way function that is "exponentially hard" but also "(sub-)exponentially weak".

[21] The extra $O(n)$ factor is because we measure η by *bit*-complexity instead of *gate*-complexity, and every gate in the (maximum) circuit needs $O(n)$ bits to describe.

Theorem 6.3. *Let $p(n) = 2^{o(n)}$ and f be a polynomial-time computable length-preserving function that is exponentially hard to invert on a $1/p(n)$ fraction of inputs. In other words, there is a constant $\varepsilon > 0$ such that for every integer n and every randomized algorithm $\mathcal{A}$ that runs in $2^{\varepsilon n}$ time,*

$$\Pr_{\mathbf{x}\leftarrow\mathcal{U}_n}\left[\mathcal{A}(f(\mathbf{x})) \in f^{-1}(f(\mathbf{x}))\right] \leq 1 - 1/p(n).$$

Then there exists a one-way function.

Proof Sketch. We verify that the standard proof for Theorem 3.12 also works in our setting. We use notations in [34, Theorem 2.3.2]. Let f be a candidate weak one-way function and $m(n) := n^2 \cdot p(n) < 2^{o(n)}$. By [34, Theorem 2.3.2], we can construct a function g on $m(n)$ inputs bits such that the following holds. Given any adversary B that inverts g w.p. $1/q(m)$, we can construct an adversary that makes $a(n) := 2n^2p(n)q(m(n))$ calls to B on input length $m(n)$ and inverts f w.p. $1 - 1/p(n)$.

Suppose that g is not a one-way function. Then there is a polynomial q and an adversary B that runs in $q(m)$ time and inverts g w.p. $1/q(m)$. We can invert f w.p. $1-1/p(n)$ by an adversary of time complexity

$$O(a(n) \cdot q(m(n))) < 2^{o(n)},$$

contradicting the hardness of f. □

6.1.2 Maximum Circuit Complexity

It will be convenient to fix an encoding of circuits into binary strings so that we can sample a uniformly random circuit with a certain description length. Fortunately, such an encoding naturally occurs in the lower bound proofs for the maximum circuit complexity, which usually use a counting argument [87, 31]: If every circuit of size $\mathrm{LB}(n)$ can be encoded as a string of length $2^n - 1$, then there must exist an n-bit Boolean function without size-$\mathrm{LB}(n)$ circuits.

In particular, in the lower bound proof of [31], the authors represented a circuit as a *stack program*. For a detailed description of stack programs, the reader is referred to [31]. We only need the following property of them:

Theorem 6.4. *There is a constant c such that every size-s circuit on n inputs can be encoded into a stack program of bit-length $(s+1)(c+\log(n+s))$.*

We also need the fact that given the description of a stack program, we can compute its truth table (the truth table of the circuit corresponding to it) in polynomial time.

We define

$$C'_{\mathsf{ub}}(n) := (C_{\mathsf{ub}}(n)+1)(\log(n + C_{\mathsf{ub}}(n)) + O(1)) \leq 2^n\left(1 + \frac{2\log n}{n} + \frac{O(1)}{n}\right).$$

By Theorem 6.4, every Boolean function over n inputs has a stack program of bit-length $C'_{\mathsf{ub}}(n)$.

We also need the following theorem, which says that for any Boolean function f on n input bits, there is a circuit of size roughly $2^n/n$ that computes f *simultaneously on multiple inputs*.

Theorem 6.5 ([93, 94]; see also [99, p. 304]). *Let $f : \{0,1\}^n \to \{0,1\}$ be any Boolean function, r be a constant. There is a circuit C of size at most $(1 + o(1))2^n/n$ such that for every $x_1, x_2, \ldots, x_r \in \{0,1\}^n$, $C(x_1, x_2, \ldots, x_r) = f(x_1) \circ f(x_2) \circ \cdots \circ f(x_r)$.*

6.2 One-Way Functions from Hardness of MCSP

In this section, we construct a one-way function assuming MCSP is (exponentially) hard on average.

Theorem 6.6. *Suppose that* MCSP *is exponentially hard on average. In particular, there is a constant $\varepsilon > 0$ and a function $q(N) = 2^{o(N)}$ such that for every randomized algorithm $\mathcal{A}$ running in $2^{\varepsilon N}$ time,*

$$\Pr_{\mathbf{tt} \leftarrow \mathcal{U}_N}[\mathcal{A}(\mathbf{tt}) = \mathsf{Size}(\mathbf{tt})] \leq 1 - 1/q(N).$$

Then there exists a one-way function.

Proof. By Theorem 6.3, it suffices to construct a length-preserving function f that satisfies the following one-wayness property: There is a function $p(\tilde{N}) = 2^{o(\tilde{N})}$ such that for every integer $\tilde{N}$ and every randomized algorithm $\mathcal{A}$ that runs in $2^{\varepsilon \tilde{N}/10}$ time,

$$\Pr_{\mathbf{x} \leftarrow \mathcal{U}_{\tilde{N}}}[\mathcal{A}(f(\mathbf{x})) \in f^{-1}(f(\mathbf{x}))] \leq 1 - 1/p(\tilde{N}). \tag{6.1}$$

We first describe the construction of f. Let $\tilde{N}$ be the input length of f and n be the largest integer such that $n + C'_{\mathsf{ub}}(n) \leq \tilde{N}$. (Recall that every Boolean function over n inputs can be represented by a circuit, or stack program, of bit-length $C'_{\mathsf{ub}}(n)$.) The first n bits of the input denote an integer $s \leq C_{\mathsf{ub}}(n)$, and the next $C'_{\mathsf{ub}}(n)$ bits denote a circuit C of size at most s. If the input is invalid (e.g., if $s > C_{\mathsf{ub}}(n)$ or the size of C is strictly larger than s), our function outputs $\perp$. Otherwise it outputs s and $tt(C)$, where $tt(C)$ is the length-2^n truth table of C. Then we pad the output to make f length-preserving.

In other words, our weak one-way function is defined as follows:

$$f(s, C) = s \circ tt(C).$$

Let $\mathcal{A}_{\mathsf{owf}}$ be any candidate adversary that tries to invert f. We will construct an algorithm $\mathcal{A}_{\mathrm{MCSP}}$ based on $\mathcal{A}_{\mathsf{owf}}$ as in Algorithm 3. In particular, $\mathcal{A}_{\mathrm{MCSP}}$ attempts to solve MCSP on truth tables of length $N := 2^n$, using $\mathcal{A}_{\mathsf{owf}}$ that attempts to invert f on input length $\tilde{N}$. Note that $\tilde{N} \leq 2N$ for large enough n. Hence, if $\mathcal{A}_{\mathsf{owf}}$ runs in $2^{\varepsilon \tilde{N}/10}$ time, then $\mathcal{A}_{\mathrm{MCSP}}$ runs in $2^{\varepsilon N}$ time. By the hardness of MCSP, $\mathcal{A}_{\mathrm{MCSP}}$ does not compute the circuit complexity correctly on a significant fraction of truth tables. Based on that, we can show that $\mathcal{A}_{\mathsf{owf}}$ satisfies Eq. (6.1).

Algorithm 3 Bounded-Error Heuristic $\mathcal{A}_{\text{MCSP}}$ for MCSP from Inverter $\mathcal{A}_{\mathsf{owf}}$ for f

```
function A_MCSP(tt)
    opt ← +∞
    for s ∈ [C_ub(n)] do
        (s', C) ← A_owf(s, tt)
        if tt(C) = tt then
            opt ← min{opt, |C|}
        end if
    end for
    return opt
end function
```

Let Err be the set of truth tables $tt \in \{0,1\}^N$ on which $\mathcal{A}_{\text{MCSP}}$ fails to output the correct answer w.p. $\geq 1/2q(N)$, where the probability is over the internal randomness of $\mathcal{A}_{\text{MCSP}}$ (which depends on the internal randomness of $\mathcal{A}_{\mathsf{owf}}$). By the hardness of MCSP and a Markov bound, we have

$$|\mathsf{Err}|/2^N \geq 1 - \frac{1 - 1/q(N)}{1 - 1/2q(N)} \geq \frac{1}{2q(N) - 1}.$$

We can see that $\mathcal{A}_{\mathsf{owf}}$ fails on every input of the form $(\mathsf{Size}(tt), tt)$ where $tt \in \mathsf{Err}$ also w.p. $\geq 1/2q(N)$. Every such input is generated in the OWF experiment w.p. at least $1/2^{C'_{\mathsf{ub}}(n)+n}$. That is:

$$\Pr[f(\mathcal{U}_{\tilde{N}}) = (\mathsf{Size}(tt), tt)] \geq 1/2^{C'_{\mathsf{ub}}(n)+n}.$$

If, in the OWF experiment, such an input (of the form $(\mathsf{Size}(tt), tt)$ where $tt \in \mathsf{Err}$) is generated, then $\mathcal{A}_{\mathsf{owf}}$ fails w.p. at least $1/2q(N)$. It follows that

$$\begin{aligned}
\Pr_{\mathbf{x} \leftarrow \mathcal{U}_{\tilde{N}}}[\mathcal{A}_{\mathsf{owf}}(f(\mathbf{x})) \notin f^{-1}(f(\mathbf{x}))] &\geq (|\mathsf{Err}|/2^{C'_{\mathsf{ub}}(n)+n}) \cdot (1/2q(N)) \\
&\geq (|\mathsf{Err}|/2^{N + O\left(\frac{N \log\log N}{\log N}\right)}) \cdot (1/2q(N)) \\
&\geq \frac{1}{(2q(N)-1)2q(N)2^{O\left(\frac{N \log\log N}{\log N}\right)}}.
\end{aligned}$$

Let $p(\tilde{N}) := (2q(N) - 1)2q(N)2^{O\left(\frac{N \log\log N}{\log N}\right)}$. Since $N = 2^n \geq \Omega(\tilde{N})$ and $q(N) = 2^{o(N)}$, it is indeed the case that $p(\tilde{N}) = 2^{o(\tilde{N})}$. We can see that every adversary $\mathcal{A}_{\mathsf{owf}}$ that runs in $2^{\varepsilon \tilde{N}/10}$ time fails to invert a random output of f w.p. $\geq p(\tilde{N})$. □

6.3 Hardness of MCSP from DLOGTIME One-Way Functions

We establish a weak converse of Theorem 6.6, namely that if there is a weak one-way function in DLOGTIME with exponential security, then MCSP is (exponentially) hard on average. The first step is to build a condEP-PRG from the one-way function.

Lemma 6.7. *Suppose that there is a weak one-way function f computable in* DLOGTIME *with security $2^{\Omega(N)}$. Then for every small enough constant $\kappa > 0$, the following holds.*

Let n be an integer and $N := 2^n$. There is a DLOGTIME*-computable condEP-PRG $\tilde{G}$ mapping $(1-\kappa)N$ bits to N bits such that for some distribution $\mathcal{E}$:*

- *no adversary of size $2^{o(N)}$ could $1/N^{10}$-distinguish $\tilde{G}(\mathcal{E})$ from the uniform distribution;*
- *the Shannon entropy of $\tilde{G}(\mathcal{E})$ is at least $(1-\kappa-o(1))N$;*
- *the outputs of $\tilde{G}$, when viewed as truth tables, have circuit complexity $\leq (1-\kappa+o(1))N/n$.*

Proof. We start by building a condEP-PRG G with linear-size skew circuits. Let K be a large enough universal constant that we fix later, $\kappa_1 := \frac{K\kappa}{1-\kappa}$. Let $M := \lfloor \frac{N}{K+\kappa_1} \rfloor$ be an input length (of f). Let $\delta := 1/\mathrm{poly}(M)$ and $\kappa_2 > 0$ be a constant such that every adversary of size $2^{\kappa_2 M}$ fails to invert f on a $1-\delta$ fraction of length-M inputs. Let $\alpha := \kappa_1 M$ be the stretch of G. For $\varepsilon := 1/4N^{10}$, the outputs of G should be 4ε-indistinguishable from uniformly random strings. We use the hash functions in Section 5.4 as the extractors and hardcore functions. Plugging in Construction 5.8, for some absolute constant $K_1 > 0$, we obtain a condEP-PRG $G : \{0,1\}^M \times \{0,1\}^{K_1 M} \to \{0,1\}^{(1+K_1+\kappa_1)M}$ with the following properties.

- For some absolute constant $K_2 > 0$, G admits a circuit of size $K_2 M$ that is skew w.r.t. the last $K_1 M$ inputs. Moreover, the first $K_1 M$ outputs of G is equal to the last $K_1 M$ inputs of G. (We can achieve this property by rearranging the output bits of G in Construction 5.8.)
- No adversary of size $2^{\Omega(M)}$ could 4ε-distinguish the outputs of G from the uniform distribution. (Note that the list size of the hardcore function is $2^{O(\alpha)} = 2^{O(\kappa_1 M)}$. So this is true as long as $\kappa_2 > O(\kappa_1) \geq O(\kappa)$.)
- The entropy of the outputs of G is at least $M - \Omega(\log M)$.

Let $\tilde{G}$ denote the randomized encoding of G (as in Section 5.4.1). Then $\tilde{G}$ is a DLOGTIME-computable condEP-PRG that maps KM input bits to $(K+\kappa_1)M$ input bits. We pad $N-(K+\kappa_1)M$ "useless" random bits to both the inputs and the outputs of $\tilde{G}$, so its output length is indeed N. The entropy of the outputs of $\tilde{G}$ is at least $N - \kappa_1 M - O(\log M) \geq (1-\kappa-o(1))N$. Again, no adversary of size $2^{o(M)}$ could 4ε-distinguish the outputs of $\tilde{G}$ from random strings.

Next, we prove that the outputs of $\tilde{G}$, when viewed as truth tables, have non-trivial circuit complexity. Let $tt^{\mathsf{in}} \in \{0,1\}^{(1-\kappa)N}$ be an input, $tt^{\mathsf{out}} := \tilde{G}(tt^{\mathsf{in}})$. Let r be a constant such that $\tilde{G}$ is a non-adaptive function that makes r queries to its input. That is, on input i, $\tilde{G}(tt^{\mathsf{in}})$ computes the indices $q(i,1), q(i,2), \ldots, q(i,r)$, queries $(tt^{\mathsf{in}})_{q(i,j)}$ for each $1 \leq j \leq r$, and computes $(tt^{\mathsf{out}})_i$ based on these answers. Note that every DLOGTIME machine making r *adaptive* queries is equivalent to a DLOGTIME machine making 2^r *non-adaptive* queries, thus it is without loss of generality to assume $\tilde{G}$ is non-adaptive.

By Theorem 6.5, there is a circuit C of size $(1-\kappa+o(1))N/n$ that on input $(x_1, x_2, \ldots, x_r)$, outputs the concatenation of $(tt^{\mathsf{in}})_{x_1}, (tt^{\mathsf{in}})_{x_2}, \ldots, (tt^{\mathsf{in}})_{x_r}$. We design a circuit for tt^{out} as follows.

- On input i, simulate $\tilde{G}(i)$ to obtain the indices $q(i,j)$ for every $1 \le j \le r$. This step takes $O(n)$ time and thus can be implemented in size $\mathrm{poly}(n)$.
- Use the circuit C of size $(1-\kappa+o(1))N/n$ to obtain $(tt^{\mathsf{in}})_{q(i,j)}$ for every $1 \le j \le r$.
- Finally, we can simulate $\tilde{G}(i)$ to obtain $(tt^{\mathsf{out}})_i$. Again, this step can be implemented in size $\mathrm{poly}(n)$.

It follows that the circuit complexity of tt^{out} is at most $(1-\kappa+o(1))N/n$. □

Now we prove the hardness of MCSP using the condEP-PRG constructed above.

Theorem 6.8. *Suppose that there is a weak one-way function f computable in* DLOGTIME *with security $2^{\Omega(N)}$. (See Definition 5.1.) Then, no non-uniform algorithm of size $2^{o(N)}$ can solve* MCSP *on a $1-2^{-o(N)}$ fraction of inputs.*

Proof. Suppose there is a constant $\varepsilon > 0$ and a non-uniform algorithm $\mathcal{A}_{\mathrm{MCSP}}$ that solves MCSP on a $1-2^{-\varepsilon N}$ fraction of inputs. Let $\tilde{G}$ be the condEP-PRG in the conclusion of Lemma 6.7 with parameter $\kappa := \varepsilon/2$ and $\mathcal{E}$ be the corresponding distribution of inputs. We will use $\mathcal{A}_{\mathrm{MCSP}}$ to construct a distinguisher $\mathcal{A}$ for $\tilde{G}(\mathcal{E})$ as follows: On input truth table tt, if $\mathcal{A}_{\mathrm{MCSP}}(tt) \ge (1-\kappa/2)N/n$, then $\mathcal{A}(tt)=1$; otherwise $\mathcal{A}(tt)=0$.

For a uniformly random truth table $\mathbf{tt} \leftarrow \mathcal{U}_N$, we have that

$$\begin{aligned}
&\Pr_{\mathbf{tt}\leftarrow\mathcal{U}_N}[\mathcal{A}(\mathbf{tt})=1] \\
&\ge 1-2^{-\varepsilon N} - \Pr_{\mathbf{tt}\leftarrow\mathcal{U}_N}[\mathsf{Size}(\mathbf{tt}) < (1-\kappa/2)N/n] \\
&\ge 1-2^{-\varepsilon N} - 2^{(s+1)(O(1)+\log(n+s))}/2^N \text{ for } s=(1-\kappa/2)N/n \qquad \text{By Theorem 6.4} \\
&\ge 1-2^{-\varepsilon N} - 2^{-(\kappa/2-o(1))N}.
\end{aligned}$$

On the other hand, consider a pseudorandom truth table $\mathbf{tt} \leftarrow \tilde{G}(\mathcal{E})$. Let $H := \mathrm{H}(\tilde{G}(\mathcal{E}))$, Good be the set of outputs of $\tilde{G}$ that are outputted w.p. at most 2^{1-H}. Then

$$\Pr_{\mathbf{tt}\leftarrow\tilde{G}(\mathcal{E})}[\mathbf{tt}\in\mathsf{Good}] \ge \frac{1}{(1-\kappa)N-H+1} \ge \frac{1}{o(N)}. \text{ (See also the proof of Lemma 4.7.)}$$

Let Err be the subset of Good for which $\mathcal{A}_{\mathrm{MCSP}}$ outputs a wrong answer, then $|\mathsf{Err}| \le 2^{(1-\varepsilon)N}$, which means

$$\Pr_{\mathbf{tt}\leftarrow\tilde{G}(\mathcal{E})}[\mathbf{tt}\in\mathsf{Err}] \le 2^{1-H}\cdot|\mathsf{Err}| \le 2^{(\kappa-\varepsilon+o(1))N}.$$

It follows that

$$\Pr_{\mathbf{tt}\leftarrow\tilde{G}(\mathcal{E})}[\mathcal{A}(\mathbf{tt})=1] \le 1-\frac{1}{o(N)}+2^{(\kappa-\varepsilon+o(1))N}.$$

Therefore, $\mathcal{A}$ is a non-uniform algorithm of size $2^{o(N)}$ such that

$$\left|\Pr_{\mathbf{tt}\leftarrow\mathcal{U}_N}[\mathcal{A}(\mathbf{tt})=1] - \Pr_{\mathbf{tt}\leftarrow\tilde{G}(\mathcal{E})}[\mathcal{A}(\mathbf{tt})=1]\right| > 1/N^2,$$

contradicting the security of $\tilde{G}$. □

Remark 6.9. Theorem 6.6 and 6.8 are not exactly converses of each other, as there are two gaps. First, there is a loss of $2^{O\left(\frac{N \log\log N}{\log N}\right)}$. Second, Theorem 6.6 only produces a (polynomial-time computable) one-way function, but Theorem 6.8 requires a DLOGTIME-computable one-way function to start with.

The first gap seems unavoidable given current knowledge about the maximum circuit complexity. However, we believe that the second gap can be eliminated. In particular, exponential average-case hardness of MCSP should imply a one-way function in DLOGTIME.

If there is a $\oplus$L *heuristic* algorithm for evaluating the truth table of a stack program, then it is indeed true that exponential hardness of MCSP implies a one-way function in DLOGTIME. Note that this heuristic only needs to succeed on *most* stack programs.[22] For example, if the circuit that corresponds to a uniformly random length-ℓ description has depth at most $O(\log \ell)$ with high probability, then a $\oplus$L heuristic can evaluate the circuit up to a particular depth, and still be correct on most inputs. We believe that a random stack program should represent a shallow circuit (w.h.p.), but we are unable to prove it.

Remark 6.10 (Results for MFSP)**.** It is possible to extend Theorem 6.6 to the case of MFSP (Minimum Formula Size Problem). In particular, if MFSP is exponentially hard on average, then there is a (super-polynomially hard) one-way function. Moreover, we only need to compute truth tables of *formulas* to evaluate this one-way function, which is in ALOGTIME [21]. Hence this one-way function is in ALOGTIME and we obtain DLOGTIME-computable one-way functions from Theorem 3.21. We omit the proof here, as it is essentially the same as Theorem 6.6 except that it uses the best bounds for maximum formula complexity (see [64] and references therein). However, we are not aware of any "mass production theorem" (Theorem 6.5) for formulas. Therefore we are not able to prove an MFSP-version of Theorem 6.8.

7 The Average-Case Complexity of MKtP

7.1 Characterizing One-Way Functions Using MKtP

We recall the main result of [70] showing an equivalence between the average-case hardness of K^p for some polynomial p and the existence of one-way functions.

Theorem 7.1 ([70])**.** *The following are equivalent:*

1. *There is a polynomial p such that K^p is bounded-error hard on average.*
2. *One-way functions exist.*
3. *For every polynomial $p(n) \geq 2n$ and constant $\lambda > 0$, K^p is bounded-error hard on average to approximate within an additive factor of $\lambda \log n$.*

[22] If this heuristic is always true (i.e. it is a worst-case algorithm), then $\oplus$L = P.

Somewhat counter-intuitively, we show a similar equivalence between the average-case hardness of Kt and the existence of one-way functions. (Note that Kt is known to be EXP-hard in the worst case under polynomial-size reductions [4].) The proof is closely analogous to the proofs in Section 4, exploiting the fact that "typical" strings of high Kt complexity can be generated from their optimal descriptions in polynomial time. Hence we just provide a sketch.

Theorem 7.2. *The following are equivalent:*

1. Kt *is bounded-error hard on average.*
2. *One-way functions exist.*
3. *For every constant $\lambda > 0$,* Kt *is bounded-error hard on average to approximate within an additive factor of $\lambda \log n$.*

Proof Sketch. (3) $\implies$ (1) is trivial.

(1) $\implies$ (2): the proof closely follows the proof of Theorem 4.1.

Suppose that there is a constant c such that every PPT algorithm computes Kt complexity correctly on at most a $1 - 1/n^c$ fraction of inputs. We observe that for all but a $1/n^{2c}$ fraction of inputs $x \in \{0,1\}^n$, optimal pairs (d,t) such that $d + \log t = \mathrm{Kt}(x)$ have the property that $t \leq O(n^{2c+1})$. Actually, for all but a $1/n^{2c}$ fraction of inputs, $\mathrm{K}(x) \geq n - 2c\log n - 1$, while for all inputs x we have $\mathrm{Kt}(x) \leq n + \log n + O(1)$. Hence for all strings x with $\mathrm{K}(x) \geq n - 2c\log n - 1$, x can be generated from its optimal Kt description in time $O(n^{2c+1})$.

We define a weak one-way function f as follows. It takes as input a triple (ℓ, k, M), where $\ell \in [n + \log n]$, $k \in [(2c+1)\log n]$, and $M \in \{0,1\}^{n+\log n}$, and outputs (ℓ, k, out). Here out is the result of running $U^{M'}$ for at most 2^k steps, where M' is the ℓ-bit prefix of M. Just as in Claim 4.3, the output distribution of f on a uniformly chosen input "almost dominates" the uniform distribution. Assume there is an inverter for f, a heuristic algorithm for the search version of $\mathrm{Kt}(x)$ can cycle over all possible ℓ and k, and find the optimal description of x. As Kt is bounded-error hard on average, our candidate one-way function f is secure.

(2) $\implies$ (3): We use Theorem 3.14 to construct a condEP-PRG G with stretch $\gamma \log n$ and security $1/n^\gamma$ from the presumed one-way function. Here, let c be a constant such that G is computable in time n^c, we choose $\gamma = \lambda + c + 2$.

We use an argument closely analogous to that of Lemma 4.7 to show that Kt is bounded-error hard on average to approximate within an additive factor of $\lambda \log n$. The idea is simple: every output of the condEP-PRG has Kt complexity at most $n + c\log n + O(1)$, while a random string of length $n + \gamma \log n$ is likely to have Kt complexity close to $n + \gamma \log n$. Hence, for our choice of parameters, an efficient heuristic algorithm that approximates Kt complexity within an additive factor of $\lambda \log n$ can distinguish the outputs of G from random. □

Theorem 7.1 and Theorem 7.2 yield the following corollary.

Corollary 7.3. Kt *is bounded-error hard on average iff there is a polynomial p such that* K^p *is bounded-error hard on average.*

Corollary 7.3 gives a new non-trivial connection between meta-complexity problems that seems hard to argue without using one-way functions as an intermediate notion.

7.2 A Complexity Theoretic Analogue

Theorem 7.2 shows that the weak average-case hardness of Kt is equivalent to the existence of cryptographic pseudo-random generators. We next show that for a slightly different setting of parameters, the average-case hardness of Kt is equivalent to the existence of complexity-theoretic pseudo-random generators against non-uniform adversaries. Thus average-case complexity of a single natural problem, namely Kt, can be used to characterize both cryptographic pseudorandomness and complexity-theoretic pseudorandomness.

Recall that cryptographic PRGs are required to be computable in fixed polynomial time but secure against adversaries that can run within any polynomial time bound. In contrast, complexity-theoretic PRGs are allowed to use more resources than the adversary.

Definition 7.4. Given functions $t : \mathbb{N} \to \mathbb{N}$, $\ell : \mathbb{N} \to \mathbb{N}$ (satisfying $\ell(n) \leq n$ for each n) and $s : \mathbb{N} \to \mathbb{N}$, we say that a family of functions $\{G_n\}$, where $G_n : \{0,1\}^{\ell(n)} \to \{0,1\}^n$ is a time t PRG (resp. i.o.PRG) with seed length ℓ against size s if $G(z)$ is computable in time $t(|z|)$ and for each n (resp. for infinitely many n), $G_n(\boldsymbol{\mathcal{U}}_{\ell(n)})$ is $1/s(n)$-indistinguishable from $\boldsymbol{\mathcal{U}}_n$ by size $s(n)$ circuits. The PRG is said to be seed-extending if z is a prefix of $G(z)$ for each seed z.

Nisan and Wigderson [78, 13] showed how to base seed-extending complexity-theoretic PRGs on the hardness of E (exponential-time). The parameters in the following theorem statement are implicit in their main result.

Theorem 7.5 ([78, 13])**.** *If* $\mathsf{DTIME}[2^n \mathrm{poly}(n)] \not\subseteq \mathsf{P}_{/\mathrm{poly}}$*, then for each* ℓ *such that* $\ell(n) = n^{\Omega(1)}$*, there is a seed-extending time* $2^\ell \mathrm{poly}(\ell)$ *i.o.PRG with seed length* ℓ *against polynomial size.*

We use Theorem 7.5 to derive an equivalence between the worst-case hardness of Kt, the existence of complexity-theoretic PRGs with non-trivial seed length, and very mild average-case hardness of Kt, where the hardness is against non-uniform adversaries. The idea of the proof is similar to that of [4], who showed that computing Kt complexity is hard for exponential-time under polynomial-size reductions.

Theorem 7.6. *The following are equivalent:*

1. $\mathsf{EXP} \not\subseteq \mathsf{P}_{/\mathrm{poly}}$.
2. *For each* $\epsilon > 0$*, there is a time* $2^\ell \mathrm{poly}(\ell)$ *i.o.PRG with seed length* n^ϵ *against polynomial size.*
3. *There are no polynomial size circuits for* Kt.
4. *For each* $\epsilon > 0$*, there is a seed-extending time* $2^\ell \mathrm{poly}(\ell)$ *i.o.PRG with seed length* n^ϵ *against polynomial size.*
5. *For any constant* $\delta > 1/2$*, there are no polynomial size circuits computing* Kt *on at least a* $1 - 1/2^{\delta n}$ *fraction of inputs of length* n *for infinitely many* n*.*

Proof. (1) $\iff$ (3) is shown in [4].

(1) $\iff$ (2) is shown in [78, 13].

(5) $\implies$ (3) is trivial.

(3) $\implies$ (4): We use Theorem 7.5. Kt can be computed in time $O(2^n \text{poly}(n))$, and we can define a decision version of Kt that is equivalent to the search version and computable in time $2^n \text{poly}(n)$ as follows: For $x \in \{0,1\}^n$ and $k \in [n + \log n]$, (x, k) is a Yes instance of the decision version of Kt iff $\text{Kt}(x) \leq k$. By Theorem 7.5, the hardness of the decision version implies that for each $\epsilon > 0$, there is a seed-extending time $2^m \text{poly}(m)$ i.o.PRG with seed length n^ϵ against polynomial size.

(4) $\implies$ (5): Consider a seed-extending $2^m \text{poly}(m)$ time i.o.PRG $G = \{G_n\}$ with seed length γn, where $1/2 > \gamma > 1 - \delta$. Such an i.o.PRG is implied by an i.o.PRG with smaller seed length, simply by ignoring all but the first n^ϵ bits of the seed when computing the output. Since the seed length is γn and the i.o.PRG is computable in time $2^{\gamma n} \text{poly}(n)$, we have that each output of the i.o.PRG has Kt complexity at most $2\gamma n + O(\log n) < n - \log n$. On the other hand, a uniformly chosen input of length n has Kt complexity very close to n, with high probability.

Suppose that there are polynomial size circuits $\{C_n\}$ computing Kt on a $1 - 1/2^{\delta n}$ fraction of inputs for all n. By our choice of δ, this means that they are correct on at least a $2/3$ fraction of strings $G_n(z)$ for seed z of length γn. Now we can define a family of distinguishers $\{D_n\}$ as follows: D_n computes $C_n(x)$ and accepts iff $C_n(x) \leq n - \log(n)$. D_n accepts with probability $2/3$ on $G_n(z)$ for uniformly chosen z, but with probability at most $1/3$ on x for uniformly chosen x of length n, since all but a $o(1)$ fraction of strings have $\text{Kt}(x) > n - \log(n)$ and C_n answers correctly on all but a $o(1)$ fraction of these strings with high Kt complexity. Therefore D_n is a distinguisher of polynomial size for all n, contradicting the assumption that G is an i.o.PRG. □

8 Open Problems

We conclude this paper with a few open questions.

Perebor hypotheses. How plausible are the Perebor hypotheses in Section 5.5? Recently, Mazor and Pass [73] and, independently, Hirahara, Ilango, and Williams [47], refuted the non-uniform version of Hypothesis 5.19, by giving an efficient circuit implementation of the inverter in [30].

It would be exciting to refute the other Strong Perebor Hypotheses. Let $t(n)$ be a polynomial (say $t(n) = 10n$ for simplicity). Is there a (probabilistic) algorithm running in $2^n/n^{\omega(1)}$ time that computes K^t in the worst case? What about the average case? Does such an algorithm imply new circuit lower bounds, as in the case of SAT algorithms [101] and learning algorithms [81]? Is there a circuit family of $2^n/n^{\omega(1)}$ size that computes cK^t (on input length $2n = n + n$)?

The Strong Exponential Time Hypothesis is used extensively in *fine-grained complexity*. Conditioning on SETH, we can prove many polynomial lower bounds for problems in P (e.g. the Orthogonal Vectors problem requires $n^{2-o(1)}$ time [100]). Do the Strong Perebor Hypotheses imply non-trivial conditional lower bounds for natural problems in P?

Random circuits. Due to our limited knowledge about circuit complexity, the relations presented in Section 6 are not tight. We point out a few questions whose resolution would tighten the relationship between MCSP and one-way functions.

First, is there an efficiently samplable distribution over circuits such that for most truth table $tt \in \{0,1\}^N$, the probability that the optimal circuit for tt is sampled is at least $2^{-N}/\mathrm{poly}(N)$? Such a distribution would imply a one-way function from super-polynomial hardness of MCSP. The trivial solution as presented in Section 6.2 is to sample a uniformly random circuit according to some encoding. The probability that the optimal circuit is sampled is $2^{-N}/2^{O\left(N\frac{\log\log N}{\log N}\right)}$.

Second, is there a $\oplus\mathsf{L}$ heuristic algorithm for evaluating a random circuit that succeeds on *most* circuits? Of course, this depends on the exact definition of "random" circuits. Such a heuristic implies a DLOGTIME-computable one-way function from the hardness of MCSP, establishing a tighter converse of Theorem 6.8.

Last, does the existence of DLOGTIME-computable one-way functions imply the hardness of MFSP? The main technical difficulty is that we do not have a formula version of Theorem 6.5.

Other cryptographic primitives? Meta-complexity can characterize the existence of one-way functions [70] and one-way functions in NC^0 (this paper). Is there a similar characterization for other cryptographic primitives, such as public-key encryption [85, 28] or indistinguishability obfuscation [15]? Recently, Ball, Liu, Mazor, and Pass [14] introduced the notion of "interactive Kolmogorov complexity" IK^t, and used the hardness of estimating IK^t to characterize key agreement. This was followed by Liu, Mazor, and Pass [69] who characterized public-key encryption by the hardness of a certain learning problem.

Is there a meta-complexity characterization of exponentially-hard *strong* one-way functions? This would bring new insights to the old question of hardness amplification for one-way functions that preserve exponential security [35].

Errorless vs. error-prone heuristics. The results of Liu and Pass [70, 71], along with our findings on the average-case hardness of meta-complexity problems, suggest a potentially significant conceptual gap between the *errorless* and *error-prone* average-case complexity of such problems. Specifically, if MKtP admits an efficient errorless heuristic, then $\mathsf{EXP} = \mathsf{BPP}$ [4, 71]. In contrast, the existence of efficient error-prone heuristics for MKtP only implies the non-existence of one-way functions, which is a far weaker consequence (compared with $\mathsf{EXP} = \mathsf{BPP}$).

This raises important questions: Which problems admit efficient errorless-to-error-prone average-case reductions? For which problems are such reductions difficult to obtain? Partly inspired by the above characterizations in meta-complexity, Hirahara and Santhanam [49] initiated a broader study of errorless vs. error-prone average-case complexity. However, these questions remain largely open and far from understood.

Acknowledgments

The first author is grateful to Lijie Chen, Mahdi Cheraghchi, and Yanyi Liu for helpful discussions. The second author thanks Yuval Ishai for a useful e-mail discussion. We thank the anonymous CCC and ToC reviewers for helpful comments that improve the presentation of this paper.

References

[1] MIKLÓS AJTAI: Generating hard instances of lattice problems (extended abstract). In *Proc. 28th STOC*, pp. 99–108. ACM Press, 1996. [doi:10.1145/237814.237838] 3

[2] ADI AKAVIA, ODED GOLDREICH, SHAFI GOLDWASSER, AND DANA MOSHKOVITZ: On basing one-way functions on NP-hardness. In *Proc. 38th STOC*, pp. 701–710. ACM Press, 2006. [doi:10.1145/1132516.1132614] 3

[3] ERIC ALLENDER: When worlds collide: Derandomization, lower bounds, and Kolmogorov complexity. In *Proc. 21st Found. Softw. Techn. Theoret. Comp. Sci. Conf. (FSTTCS'01)*, volume 2245 of *LNCS*, pp. 1–15. Springer, 2001. [doi:10.1007/3-540-45294-X_1] 3, 4, 6, 17

[4] ERIC ALLENDER, HARRY BUHRMAN, MICHAL KOUCKÝ, DIETER VAN MELKEBEEK, AND DETLEF RONNEBURGER: Power from random strings. *SIAM J. Comput.*, 35(6):1467–1493, 2006. [doi:10.1137/050628994] 3, 4, 5, 6, 10, 15, 19, 61, 62, 63, 64

[5] ERIC ALLENDER, MAHDI CHERAGHCHI, DIMITRIOS MYRISIOTIS, HARSHA TIRUMALA, AND ILYA VOLKOVICH: One-way functions and a conditional variant of MKTP. In *Proc. 41st Found. Softw. Techn. Theoret. Comp. Sci. Conf. (FSTTCS'21)*, pp. 7:1–19. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2021. [doi:10.4230/LIPIcs.FSTTCS.2021.7] 9

[6] ERIC ALLENDER AND SHUICHI HIRAHARA: New insights on the (non-)hardness of circuit minimization and related problems. *ACM Trans. Comput. Theory*, 11(4):27:1–27, 2019. [doi:10.1145/3349616] 3

[7] NOGA ALON, JEHOSHUA BRUCK, JOSEPH NAOR, MONI NAOR, AND RON M. ROTH: Construction of asymptotically good low-rate error-correcting codes through pseudo-random graphs. *IEEE Trans. Inform. Theory*, 38(2):509–516, 1992. [doi:10.1109/18.119713] 52

[8] BENNY APPLEBAUM: *Cryptography in Constant Parallel Time*. Information Security and Cryptography. Springer, 2014. [doi:10.1007/978-3-642-17367-7] 4, 9, 11

[9] BENNY APPLEBAUM: Cryptographic hardness of random local functions – Survey. *Comput. Complexity*, 25(3):667–722, 2016. [doi:10.1007/s00037-015-0121-8] 9

[10] BENNY APPLEBAUM: Exponentially-hard Gap-CSP and local PRG via local hardcore functions. In *Proc. 58th FOCS*, pp. 836–847. IEEE Comp. Soc., 2017. [doi:10.1109/FOCS.2017.82] 14, 15, 40

[11] BENNY APPLEBAUM, YUVAL ISHAI, AND EYAL KUSHILEVITZ: Cryptography in NC^0. *SIAM J. Comput.*, 36(4):845–888, 2006. [doi:10.1137/S0097539705446950] 4, 5, 7, 9, 11, 12, 22, 23, 27, 32, 40

[12] SANJEEV ARORA AND BOAZ BARAK: *Computational Complexity: A Modern Approach*. Cambridge Univ. Press, 2009. [doi:10.1017/CBO9780511804090] 31, 32

[13] LÁSZLÓ BABAI, LANCE FORTNOW, NOAM NISAN, AND AVI WIGDERSON: BPP has subexponential time simulations unless EXPTIME has publishable proofs. *Comput. Complexity*, 3:307–318, 1993. [doi:10.1007/BF01275486] 62, 63

[14] MARSHALL BALL, YANYI LIU, NOAM MAZOR, AND RAFAEL PASS: Kolmogorov comes to cryptomania: On interactive Kolmogorov complexity and key-agreement. In *Proc. 64th FOCS*, pp. 458–483. IEEE Comp. Soc., 2023. [doi:10.1109/FOCS57990.2023.00034] 64

[15] BOAZ BARAK, ODED GOLDREICH, RUSSELL IMPAGLIAZZO, STEVEN RUDICH, AMIT SAHAI, SALIL P. VADHAN, AND KE YANG: On the (im)possibility of obfuscating programs. *J. ACM*, 59(2):6:1–48, 2012. [doi:10.1145/2160158.2160159] 64

[16] JOSHUA BARON, YUVAL ISHAI, AND RAFAIL OSTROVSKY: On linear-size pseudorandom generators and hardcore functions. *Theoret. Comput. Sci.*, 554:50–63, 2014. [doi:10.1016/j.tcs.2014.06.013] 14, 40, 42

[17] ELI BIHAM, YARON J. GOREN, AND YUVAL ISHAI: Basing weak public-key cryptography on strong one-way functions. In *Proc. Theory of Cryptography Conf. (TCC'08)*, pp. 55–72. Springer, 2008. [doi:10.1007/978-3-540-78524-8_4] 44

[18] AVRIM BLUM, MERRICK L. FURST, MICHAEL J. KEARNS, AND RICHARD J. LIPTON: Cryptographic primitives based on hard learning problems. In *Proc. 13th Ann. Internat. Cryptology Conf. (CRYPTO'93)*, pp. 278–291. Springer, 1993. [doi:10.1007/3-540-48329-2_24] 9

[19] ANDREJ BOGDANOV AND LUCA TREVISAN: On worst-case to average-case reductions for NP problems. *SIAM J. Comput.*, 36(4):1119–1159, 2006. [doi:10.1137/S0097539705446974] 3

[20] J. L. BORDEWIJK: Inter-reciprocity applied to electrical networks. *Applied Scientific Research, Section A*, pp. 1–74, 1957. [doi:10.1007/BF02410413] 53

[21] SAMUEL R. BUSS: The Boolean formula value problem is in ALOGTIME. In *Proc. 19th STOC*, pp. 123–131. ACM Press, 1987. [doi:10.1145/28395.28409] 60

[22] CHRIS CALABRO, RUSSELL IMPAGLIAZZO, AND RAMAMOHAN PATURI: The complexity of satisfiability of small depth circuits. In *Parameterized and Exact Computation, 4th International Workshop, (IWPEC) 2009*, volume 5917 of *Lecture Notes in Computer Science*, pp. 75–85. Springer, 2009. [doi:10.1007/978-3-642-11269-0_6] 8, 43

[23] RAN CANETTI, YEVGENIY DODIS, SHAI HALEVI, EYAL KUSHILEVITZ, AND AMIT SAHAI: Exposure-resilient functions and all-or-nothing transforms. In *Proc. 29th Ann. Internat. Conf. on Theory Appl. of Cryptographic Techniques (EUROCRYPT'00)*, pp. 453–469. Springer, 2000. [doi:10.1007/3-540-45539-6_33] 53

[24] MARCO CARMOSINO, KENNETH HOOVER, RUSSELL IMPAGLIAZZO, VALENTINE KABANETS, AND ANTONINA KOLOKOLOVA: Lifting for constant-depth circuits and applications to MCSP. In *Proc. 48th Internat. Colloq. on Automata, Languages, and Programming (ICALP'21)*, pp. 44:1–20. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2021. [doi:10.4230/LIPICS.ICALP.2021.44] 10

[25] LIJIE CHEN, CE JIN, AND R. RYAN WILLIAMS: Hardness magnification for all sparse NP languages. In *Proc. 60th FOCS*, pp. 1240–1255. IEEE Comp. Soc., 2019. [doi:10.1109/FOCS.2019.00077] 3

[26] LIJIE CHEN AND HANLIN REN: Strong average-case lower bounds from non-trivial derandomization. In *Proc. 52nd STOC*, pp. 1327–1334. ACM Press, 2020. [doi:10.1145/3357713.3384279] 22, 23

[27] BENNY CHOR, ODED GOLDREICH, JOHAN HÅSTAD, JOEL FRIEDMAN, STEVEN RUDICH, AND ROMAN SMOLENSKY: The bit extraction problem or t-resilient functions (preliminary version). In *Proc. 26th FOCS*, pp. 396–407. IEEE Comp. Soc., 1985. [doi:10.1109/SFCS.1985.55] 53

[28] WHITFIELD DIFFIE AND MARTIN E. HELLMAN: New directions in cryptography. *IEEE Trans. Inform. Theory*, 22(6):644–654, 1976. [doi:10.1109/TIT.1976.1055638] 3, 64

[29] BILL FEFFERMAN, RONEN SHALTIEL, CHRISTOPHER UMANS, AND EMANUELE VIOLA: On beating the hybrid argument. *Theory of Computing*, 9:809–843, 2013. [doi:10.4086/toc.2013.v009a026] 12, 22

[30] AMOS FIAT AND MONI NAOR: Rigorous time/space trade-offs for inverting functions. *SIAM J. Comput.*, 29(3):790–803, 1999. [doi:10.1137/S0097539795280512] 44, 63

[31] GUDMUND SKOVBJERG FRANDSEN AND PETER BRO MILTERSEN: Reviewing bounds on the circuit size of the hardest functions. *Inform. Process. Lett.*, 95(2):354–357, 2005. [doi:10.1016/j.ipl.2005.03.009] 15, 54, 55

[32] MARTIN FÜRER: The tight deterministic time hierarchy. In *Proc. 14th STOC*, pp. 8–16. ACM Press, 1982. [doi:10.1145/800070.802172] 18

[33] OFER GABBER AND ZVI GALIL: Explicit constructions of linear-sized superconcentrators. *J. Comput. System Sci.*, 22(3):407–420, 1981. [doi:10.1016/0022-0000(81)90040-4] 49, 50

[34] ODED GOLDREICH: *The Foundations of Cryptography – Volume 1: Basic Techniques*. Cambridge Univ. Press, 2001. [doi:10.1017/CBO9780511546891] 5, 20, 55

[35] ODED GOLDREICH, RUSSELL IMPAGLIAZZO, LEONID A. LEVIN, RAMARATHNAM VENKATESAN, AND DAVID ZUCKERMAN: Security preserving amplification of hardness. In *Proc. 31st FOCS*, pp. 318–326. IEEE Comp. Soc., 1990. [doi:10.1109/FSCS.1990.89550] 35, 64

[36] ODED GOLDREICH AND LEONID A. LEVIN: A hard-core predicate for all one-way functions. In *Proc. 21st STOC*, pp. 25–32. ACM Press, 1989. [doi:10.1145/73007.73010] 14, 36, 38

[37] SHAFI GOLDWASSER, DAN GUTFREUND, ALEXANDER HEALY, TALI KAUFMAN, AND GUY N. ROTHBLUM: Verifying and decoding in constant depth. In *Proc. 39th STOC*, pp. 440–449. ACM Press, 2007. [doi:10.1145/1250790.1250855] 22

[38] ALEXANDER GOLOVNEV, RAHUL ILANGO, RUSSELL IMPAGLIAZZO, VALENTINE KABANETS, ANTONINA KOLOKOLOVA, AND AVISHAY TAL: $\mathsf{AC}^0[p]$ lower bounds against MCSP via the coin problem. In *Proc. 46th Internat. Colloq. on Automata, Languages, and Programming (ICALP'19)*, pp. 66:1–15. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2019. [doi:10.4230/LIPIcs.ICALP.2019.66] 3

[39] VENKATESAN GURUSWAMI AND PIOTR INDYK: Expander-based constructions of efficiently decodable codes. In *Proc. 42nd FOCS*, pp. 658–667. IEEE Comp. Soc., 2001. [doi:10.1109/SFCS.2001.959942] 52

[40] VENKATESAN GURUSWAMI, CHRISTOPHER UMANS, AND SALIL P. VADHAN: Unbalanced expanders and randomness extractors from Parvaresh–Vardy codes. *J. ACM*, 56(4):20:1–34, 2009. [doi:10.1145/1538902.1538904] 13, 35, 38

[41] IFTACH HAITNER, OMER REINGOLD, AND SALIL P. VADHAN: Efficiency improvements in constructing pseudorandom generators from one-way functions. *SIAM J. Comput.*, 42(3):1405–1430, 2013. [doi:10.1137/100814421] 11

[42] JOHAN HÅSTAD, RUSSELL IMPAGLIAZZO, LEONID A. LEVIN, AND MICHAEL LUBY: A pseudorandom generator from any one-way function. *SIAM J. Comput.*, 28(4):1364–1396, 1999. [doi:10.1137/S0097539793244708] 6, 11, 37, 42

[43] MARTIN E. HELLMAN: A cryptanalytic time-memory trade-off. *IEEE Trans. Inform. Theory*, 26(4):401–406, 1980. [doi:10.1109/TIT.1980.1056220] 44

[44] SHUICHI HIRAHARA: Non-black-box worst-case to average-case reductions within NP. In *Proc. 59th FOCS*, pp. 247–258. IEEE Comp. Soc., 2018. [doi:10.1109/FOCS.2018.00032] 3

[45] SHUICHI HIRAHARA: Characterizing average-case complexity of PH by worst-case meta-complexity. In *Proc. 61st FOCS*, pp. 50–60. IEEE Comp. Soc., 2020. [doi:10.1109/FOCS46700.2020.00014] 14, 38, 39

[46] SHUICHI HIRAHARA: Unexpected hardness results for Kolmogorov complexity under uniform reductions. In *Proc. 52nd STOC*, pp. 1038–1051. ACM Press, 2020. [doi:10.1145/3357713.3384251] 14, 38

[47] SHUICHI HIRAHARA, RAHUL ILANGO, AND R. RYAN WILLIAMS: Beating brute force for compression problems. In *Proc. 56th STOC*, pp. 659–670. ACM Press, 2024. [doi:10.1145/3618260.3649778] 10, 63

[48] SHUICHI HIRAHARA AND RAHUL SANTHANAM: On the average-case complexity of MCSP and its variants. In *Proc. 32nd Comput. Complexity Conf. (CCC'17)*, pp. 7:1–20. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2017. [doi:10.4230/LIPIcs.CCC.2017.7] 3, 9, 22, 32

[49] SHUICHI HIRAHARA AND RAHUL SANTHANAM: Errorless versus error-prone average-case complexity. In *Proc. 13th Innovations in Theoret. Comp. Sci. Conf. (ITCS'22)*, pp. 84:1–23. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2022. [doi:10.4230/LIPICS.ITCS.2022.84] 64

[50] SHUICHI HIRAHARA AND NOBUTAKA SHIMIZU: Planted clique conjectures are equivalent. In *Proc. 56th STOC*, pp. 358–366. ACM Press, 2024. [doi:10.1145/3618260.3649751] 4

[51] THOMAS HOLENSTEIN, UELI M. MAURER, AND JOHAN SJÖDIN: Complete classification of bilinear hard-core functions. In *Proc. 24th Ann. Internat. Cryptology Conf. (CRYPTO'04)*, pp. 73–91. Springer, 2004. [doi:10.1007/978-3-540-28628-8_5] 14, 40, 42

[52] SHLOMO HOORY, NATHAN LINIAL, AND AVI WIGDERSON: Expander graphs and their applications. *Bull. AMS*, pp. 439–561, 2006. [doi:10.1090/S0273-0979-06-01126-8] 49

[53] RAHUL ILANGO: Constant depth formula and partial function versions of MCSP are hard. In *Proc. 61st FOCS*, pp. 424–433. IEEE Comp. Soc., 2020. [doi:10.1109/FOCS46700.2020.00047] 3

[54] RAHUL ILANGO: Connecting perebor conjectures: Towards a search to decision reduction for minimizing formulas. In *Proc. 35th Comput. Complexity Conf. (CCC'20)*, pp. 31:1–35. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2020 (virtual conference). [doi:10.4230/LIPIcs.CCC.2020.31] 3, 54

[55] RAHUL ILANGO, BRUNO LOFF, AND IGOR CARBONI OLIVEIRA: NP-hardness of circuit minimization for multi-output functions. In *Proc. 35th Comput. Complexity Conf. (CCC'20)*, pp. 22:1–36. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2020 (virtual conference). [doi:10.4230/LIPIcs.CCC.2020.22] 3

[56] RUSSELL IMPAGLIAZZO: Hard-core distributions for somewhat hard problems. In *Proc. 36th FOCS*, pp. 538–545. IEEE Comp. Soc., 1995. [doi:10.1109/SFCS.1995.492584] 14, 37, 44

[57] RUSSELL IMPAGLIAZZO: A personal view of average-case complexity. In *Proc. 10th IEEE Conf. Structure in Complexity Theory (SCT'95)*, pp. 134–147. IEEE Comp. Soc., 1995. [doi:10.1109/SCT.1995.514853] 3

[58] RUSSELL IMPAGLIAZZO AND LEONID A. LEVIN: No better ways to generate hard NP instances than picking uniformly at random. In *Proc. 31st FOCS*, pp. 812–821. IEEE Comp. Soc., 1990. [doi:10.1109/FSCS.1990.89604] 8

[59] RUSSELL IMPAGLIAZZO AND RAMAMOHAN PATURI: On the complexity of k-SAT. *J. Comput. System Sci.*, 62(2):367–375, 2001. [doi:10.1006/jcss.2000.1727] 8, 43

[60] RUSSELL IMPAGLIAZZO, RAMAMOHAN PATURI, AND FRANCIS ZANE: Which problems have strongly exponential complexity? *J. Comput. System Sci.*, 63(4):512–530, 2001. [doi:10.1006/jcss.2001.1774] 8, 43

[61] YUVAL ISHAI AND EYAL KUSHILEVITZ: Randomizing polynomials: A new representation with applications to round-efficient secure computation. In *Proc. 41st FOCS*, pp. 294–304. IEEE Comp. Soc., 2000. [doi:10.1109/SFCS.2000.892118] 22, 23

[62] YUVAL ISHAI AND EYAL KUSHILEVITZ: Perfect constant-round secure computation via perfect randomizing polynomials. In *Proc. 29th Internat. Colloq. on Automata, Languages, and Programming (ICALP'02)*, pp. 244–256. Springer, 2002. [doi:10.1007/3-540-45465-9_22] 22, 23

[63] YUVAL ISHAI, EYAL KUSHILEVITZ, RAFAIL OSTROVSKY, AND AMIT SAHAI: Cryptography with constant computational overhead. In *Proc. 40th STOC*, pp. 433–442. ACM Press, 2008. [doi:10.1145/1374376.1374438] 40, 42, 49, 52, 53

[64] STASYS JUKNA: *Boolean Function Complexity - Advances and Frontiers*. Volume 27 of *Algorithms and combinatorics*. Springer, 2012. [doi:10.1007/978-3-642-24508-4] 60

[65] VALENTINE KABANETS AND JIN-YI CAI: Circuit minimization problem. In *Proc. 32nd STOC*, pp. 73–79. ACM Press, 2000. [doi:10.1145/335305.335314] 2, 3, 10

[66] KER-I KO: On the complexity of learning minimum time-bounded Turing machines. *SIAM J. Comput.*, 20(5):962–986, 1991. [doi:10.1137/0220059] 17

[67] LEONID A. LEVIN: Randomness conservation inequalities; information and independence in mathematical theories. *Inform. Control*, 61(1):15–37, 1984. [doi:10.1016/S0019-9958(84)80060-1] 4, 17

[68] LEONID A. LEVIN: The tale of one-way functions. *Probl. Info. Transmission*, 39(1):92–103, 2003. [doi:10.1023/A%3A1023634616182] 5

[69] YANYI LIU, NOAM MAZOR, AND RAFAEL PASS: On white-box learning and public-key encryption. In *Proc. 16th Innovations in Theoret. Comp. Sci. Conf. (ITCS'25)*, pp. 73:1–22. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2025. [doi:10.4230/LIPICS.ITCS.2025.73] 64

[70] YANYI LIU AND RAFAEL PASS: On one-way functions and Kolmogorov complexity. In *Proc. 61st FOCS*, pp. 1243–1254. IEEE Comp. Soc., 2020. [doi:10.1109/FOCS46700.2020.00118] 4, 5, 9, 10, 11, 12, 13, 14, 21, 26, 30, 31, 34, 35, 36, 37, 39, 49, 54, 60, 64

[71] YANYI LIU AND RAFAEL PASS: On the possibility of basing cryptography on $\mathsf{EXP} \neq \mathsf{BPP}$. In *Proc. 41st Ann. Internat. Cryptology Conf. (CRYPTO'21)*, pp. 11–40. Springer, 2021. [doi:10.1007/978-3-030-84242-0_2] 9, 64

[72] GREGORY A. MARGULIS: Explicit constructions of expanders. *Problemy Peredachi Informatsii*, pp. 71–80, 1973. Link at Mathnet.ru. Original title: Yavnye konstruktsii rasshiritelei. 49

[73] NOAM MAZOR AND RAFAEL PASS: The non-uniform Perebor conjecture for time-bounded Kolmogorov complexity is false. In *Proc. 15th Innovations in Theoret. Comp. Sci. Conf. (ITCS'24)*, pp. 80:1–20. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2024. [doi:10.4230/LIPICS.ITCS.2024.80] 10, 63

[74] DYLAN M. MCKAY, CODY D. MURRAY, AND R. RYAN WILLIAMS: Weak lower bounds on resource-bounded compression imply strong separations of complexity classes. In *Proc. 51st STOC*, pp. 1215–1225. ACM Press, 2019. [doi:10.1145/3313276.3316396] 3

[75] CODY D. MURRAY AND R. RYAN WILLIAMS: On the (non) NP-hardness of computing circuit complexity. *Theory of Computing*, 13(1):1–22, 2017. [doi:10.4086/toc.2017.v013a004] 3

[76] MIKITO NANASHIMA: On basing auxiliary-input cryptography on NP-hardness via nonadaptive black-box reductions. In *Proc. 12th Innovations in Theoret. Comp. Sci. Conf. (ITCS'21)*, pp. 29:1–15. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2021 (virtual conference). [doi:10.4230/LIPIcs.ITCS.2021.29] 3

[77] NOAM NISAN: Extracting randomness: How and why. A survey. In *Proc. 11th IEEE Conf. on Comput. Complexity (CCC'96)*, pp. 44–58. IEEE Comp. Soc., 1996. [doi:10.1109/CCC.1996.507667] 13

[78] NOAM NISAN AND AVI WIGDERSON: Hardness vs randomness. *J. Comput. System Sci.*, 49(2):149–167, 1994. [doi:10.1016/S0022-0000(05)80043-1] 6, 15, 62, 63

[79] NOAM NISAN AND DAVID ZUCKERMAN: More deterministic simulation in logspace. In *Proc. 25th STOC*, pp. 235–244. ACM Press, 1993. [doi:10.1145/167088.167162] 13

[80] IGOR C. OLIVEIRA, JÁN PICH, AND RAHUL SANTHANAM: Hardness magnification near state-of-the-art lower bounds. *Theory of Computing*, 17:1–38, 2021. [doi:10.4086/toc.2021.v017a011] 3

[81] IGOR CARBONI OLIVEIRA AND RAHUL SANTHANAM: Conspiracies between learning algorithms, circuit lower bounds, and pseudorandomness. In *Proc. 32nd Comput. Complexity Conf. (CCC'17)*, pp. 18:1–49. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2017. [doi:10.4230/LIPIcs.CCC.2017.18] 3, 9, 63

[82] IGOR CARBONI OLIVEIRA AND RAHUL SANTHANAM: Hardness magnification for natural problems. In *Proc. 59th FOCS*, pp. 65–76. IEEE Comp. Soc., 2018. [doi:10.1109/FOCS.2018.00016] 3

[83] ALEXANDER A. RAZBOROV AND STEVEN RUDICH: Natural proofs. *J. Comput. System Sci.*, 55(1):24–35, 1997. [doi:10.1006/jcss.1997.1494] 9, 10

[84] HANLIN REN AND RAHUL SANTHANAM: Hardness of KT characterizes parallel cryptography. In *Proc. 36th Comput. Complexity Conf. (CCC'21)*, pp. 35:1–58. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2021 (virtual conference). [doi:10.4230/LIPICS.CCC.2021.35] 1

[85] RONALD L. RIVEST, ADI SHAMIR, AND LEONARD M. ADLEMAN: A method for obtaining digital signatures and public-key cryptosystems. *Comm. ACM*, 21(2):120–126, 1978. [doi:10.1145/359340.359342] 3, 64

[86] RAHUL SANTHANAM: Pseudorandomness and the minimum circuit size problem. In *Proc. 11th Innovations in Theoret. Comp. Sci. Conf. (ITCS'20)*, pp. 68:1–26. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2020. [doi:10.4230/LIPIcs.ITCS.2020.68] 9

[87] CLAUDE E. SHANNON: The synthesis of two-terminal switching circuits. *Bell Sys. Tech. J.*, 28(1):59–98, 1949. [doi:10.1002/j.1538-7305.1949.tb03624.x] 55

[88] DANIEL A. SPIELMAN: Linear-time encodable and decodable error-correcting codes. *IEEE Trans. Inform. Theory*, 42(6):1723–1731, 1996. [doi:10.1109/18.556668] 49, 50, 51

[89] AMNON TA-SHMA AND DAVID ZUCKERMAN: Extractor codes. *IEEE Trans. Inform. Theory*, 50(12):3015–3025, 2004. [doi:10.1109/TIT.2004.838377] 14

[90] ROEI TELL: Quantified derandomization of linear threshold circuits. In *Proc. 50th STOC*, pp. 855–865. ACM Press, 2018. [doi:10.1145/3188745.3188822] 52

[91] BORIS A. TRAKHTENBROT: A survey of Russian approaches to perebor (brute-force searches) algorithms. *IEEE Annals of the History of Computing*, 6(4):384–400, 1984. [doi:10.1109/MAHC.1984.10036] 2, 5

[92] LUCA TREVISAN: Extractors and pseudorandom generators. *J. ACM*, 48(4):860–879, 2001. [doi:10.1145/502090.502099] 14

[93] DIETMAR UHLIG: On the synthesis of self-correcting schemes from functional elements with a small number of reliable elements. *Math. Notes. Acad. Sci. USSR (English translation)*, 15:558–562, 1974. [doi:10.1007/BF01152835] 15, 56

[94] DIETMAR UHLIG: Zur Parallelberechnung Boolescher Funktionen. *TR Ing. Hochsch. Mittweida*, 1984. 15, 56

[95] SALIL P. VADHAN: Pseudorandomness. *Found. Trends Theor. Comp. Sci.*, 7(1–3):1–336, 2012. [doi:10.1561/0400000010] 37

[96] SALIL P. VADHAN AND COLIN JIA ZHENG: A uniform min-max theorem with applications in cryptography. In *Proc. 33rd Ann. Internat. Cryptology Conf. (CRYPTO'13)*, pp. 93–110. Springer, 2013. [doi:10.1007/978-3-642-40041-4_6] 35

[97] VIRGINIA VASSILEVSKA WILLIAMS: On some fine-grained questions in algorithms and complexity. In BOYAN SIRAKOV, PAULO NEY DE SOUZA, AND MARCELO VIANA, editors, *Proceedings of the International Congress of Mathematicians (ICM 2018), Rio de Janeiro*, volume 3, pp. 3431–3472. World Scientific, 2018. Available at author's website. [doi:10.1142/9789813272880_0188] 43

[98] HOETECK WEE: Finding Pessiland. In *Proc. Theory of Cryptography Conf. (TCC'06)*, pp. 429–442. Springer, 2006. [doi:10.1007/11681878_22] 3

[99] INGO WEGENER: *The complexity of Boolean functions*. Wiley–Teubner, 1987. Link at Univ. Dortmund. 56

[100] RYAN WILLIAMS: A new algorithm for optimal 2-constraint satisfaction and its implications. *Theoret. Comput. Sci.*, 348(2-3):357–365, 2005. [doi:10.1016/j.tcs.2005.09.023] 63

[101] RYAN WILLIAMS: Improving exhaustive search implies superpolynomial lower bounds. *SIAM J. Comput.*, 42(3):1218–1244, 2013. [doi:10.1137/10080703X] 63

[102] ANDREW CHI-CHIH YAO: Theory and applications of trapdoor functions (extended abstract). In *Proc. 23rd FOCS*, pp. 80–91. IEEE Comp. Soc., 1982. [doi:10.1109/SFCS.1982.45] 14, 20

[103] YU YU, XIANGXUE LI, AND JIAN WENG: Pseudorandom generators from regular one-way functions: New constructions with improved parameters. *Theoret. Comput. Sci.*, 569:58–69, 2015. [doi:10.1016/j.tcs.2014.12.013] 35, 36, 37

AUTHORS

Hanlin Ren
Postdoctoral Member
School of Mathematics
Institute for Advanced Study
Princeton, NJ, USA
h4n1in.r3n@gmail.com
https://hanlin-ren.github.io/

Rahul Santhanam
Professor
Department of Computer Science
University of Oxford
Oxford, Oxfordshire, UK
rahul.santhanam@cs.ox.ac.uk
https://www.cs.ox.ac.uk/people/Rahul.Santhanam/

ABOUT THE AUTHORS

HANLIN REN is a postdoctoral member at the Institute for Advanced Study. He received his DPhil from the University of Oxford under the supervision of Rahul Santhanam. He is interested in circuit complexity, but he found himself completely unable to understand circuits. He recently turned to proof complexity and finds himself completely unable to understand proofs as well.

RAHUL SANTHANAM works on computational complexity theory and its connections with cryptography, learning theory, game theory and algorithms. He had the privilege of doing his PhD at the University of Chicago, and later worked at Simon Fraser University, University of Toronto and University of Edinburgh, before arriving at his present position as Professor of Computer Science at University of Oxford and Tutorial Fellow in Computer Science at Magdalen College, Oxford. His main leisure activities are reading (mostly about complexity theory) and walking (while thinking about complexity theory).