


# Hardness of $K^t$ Characterizes Parallel Cryptography

Hanlin Ren   

Institute for Interdisciplinary Information Sciences, Tsinghua University, Beijing, China

Rahul Santhanam 

University of Oxford, UK

---

## 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  $K^t$  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  $K^t$  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  $K^t$  complexity and a variant of  $K^t$  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  $K^t$ . 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.

**2012 ACM Subject Classification** Theory of computation → Cryptographic primitives; Theory of computation → Problems, reductions and completeness; Theory of computation → Circuit complexity

**Keywords and phrases** one-way function, meta-complexity,  $K^t$  complexity, parallel cryptography, randomized encodings

**Digital Object Identifier** 10.4230/LIPIcs.CCC.2021.35

**Related Version** *Full Version*: <https://eccc.weizmann.ac.il/report/2021/057/>

**Acknowledgements** 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 would like to thank anonymous CCC reviewers for helpful comments that improve the presentation of this paper.



© Hanlin Ren and Rahul Santhanam;  
licensed under Creative Commons License CC-BY 4.0  
36th Computational Complexity Conference (CCC 2021).

Editor: Valentine Kabanets; Article No. 35; pp. 35:1–35:58  
Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany



## 1 Introduction

### 1.1 Backgrounds and Motivation

#### 1.1.1 Meta-Complexity

Let  $\mu$  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  $\mu$ : given an input function, what is its  $\mu$  value?

Meta-complexity problems are fundamental to theoretical computer science and have been studied since the very beginning of the discipline [81]. 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  $\mu$  complete for some natural complexity class? For example, is the Minimum Circuit Size Problem (MCSP, [58]) NP-complete?
- Can we show unconditional circuit lower bounds for computing  $\mu$ , at least for weak circuit classes? Can we distinguish truth tables with  $2^{o(n)}$ -size circuits from random truth tables by a small  $AC^0[2]$  circuit?
- Is deciding whether  $\mu$  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  $\mu$ ? 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  $\mu$  and  $\mu'$  can we show that the problem of computing  $\mu$  reduces to the problem of computing  $\mu'$ ? 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 affirmatively for specific measures [4, 48, 3, 6, 72, 43, 34, 40, 46, 47], there are also barriers to understanding these questions better, such as our inability to prove circuit lower bounds [58, 66] and the magnification phenomenon [73, 71, 65, 22]. 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 [75], discrete logarithm [25], 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)  $\text{NP} \not\subseteq \text{BPP}$  [25]. 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 [17]. As the latter is necessary for one-way functions, this barrier result demonstrates limits of such “black-box” reductions on basing one-way function from worst-case assumptions such as  $\text{NP} \not\subseteq \text{BPP}$ . For the task of constructing one-way functions (instead of just a hard-on-average problem in NP), stronger barrier results are known [2, 67].
- Even the seemingly easier task of basing one-way functions from *average-case* hardness of NP remains elusive. Indeed, Impagliazzo [50] 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 [87], 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 [62] 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, [62] 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?<sup>1</sup>

Given these motivations, it is natural to ask if the main result of [62] 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 [4]. Maybe this problem is also closely related to the existence of one-way functions? What about Levin’s  $K_t$  complexity [60]?<sup>2</sup>

<sup>1</sup> 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.

<sup>2</sup> See Definition 14 for the precise definitions of  $K^t$ , KT, and  $K_t$ .

## 1.2 Our Contributions

We give strong positive answers to the above questions. We show somewhat surprisingly that the average-case hardness of  $K^t$  complexity is equivalent to the existence of one-way functions computable in *fast parallel time*.<sup>3</sup> 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  $K^t$  complexity turns out to be equivalent to the existence of one-way functions in one natural setting of parameters (despite the fact that computing  $K^t$  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<sup>4</sup>, 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** (Main Result; Informal). *There is a one-way function computable in uniform  $NC^1$  if and only if  $K^t$  is bounded-error hard on average.*

The class “uniform  $NC^1$ ” in the above theorem is somewhat arbitrary since [11] proved that the existence of one-way functions in  $\oplus L$  implies the existence of one-way functions in  $NC^0$ .<sup>5</sup>

For comparison, Liu and Pass [62] showed an equivalence between (“sequential”) cryptography and the average-case hardness of time-bounded Kolmogorov complexity ( $K^t$ ).

► **Theorem 2** (Main Result of [62]). *There is a one-way function if and only if for some polynomial  $t$ ,  $K^t$  is bounded-error hard on average.*

Theorem 2 shows that the one-way function defined based on hardness of  $K^t$  is a natural *universal* one-way function.<sup>6</sup> Similarly, Theorem 1 shows that the one-way function we define based on the hardness of  $K^t$  is a natural universal one-way function in  $NC^1$ .

<sup>3</sup> 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.

<sup>4</sup> Our terminology is inspired by Trakhtenbrot’s survey [81] 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.

<sup>5</sup>  $\oplus L$  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).

<sup>6</sup> An artificial universal one-way function can be defined by enumerating uniform algorithms and concatenating their outputs [61, 30].

As a corollary, the classical open question of whether polynomial-time computable one-way functions imply one-way functions in  $\text{NC}^0$  is equivalent to the question of whether average-case hardness of  $\text{Kt}$  implies average-case hardness of  $\text{KT}$ .

**Results for MCSP.** The  $\text{KT}$  complexity was defined as a variant of Kolmogorov-complexity that resembles circuit complexity [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, as follows:

► **Theorem 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  $\text{NC}^0$ , 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 76).

**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  $\text{Kt}$ .

► **Theorem 4.** *There is a one-way function if and only if  $\text{Kt}$  is bounded-error hard on average.*

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

The harder direction in Theorem 4 is to construct a one-way function from hardness of  $\text{Kt}$ . 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  $\text{Kt}$  complexity is witnessed by a machine  $d$  and a time bound  $t$  where  $\text{Kt}(x) = |d| + \log t$ , we have  $t \leq \text{poly}(n)$ . We refer the reader to Section 2.1.2 and Section 7 for more details.

Note that Theorem 4 can also be seen as a characterization of cryptographic pseudorandomness, by the known equivalence between one-way functions and cryptographic pseudorandomness [38]. In a different regime of parameters, average-case hardness of  $\text{Kt}$  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 ( $\text{Kt}$ ) can be used to capture both cryptographic pseudorandomness and complexity-theoretic pseudorandomness!

► **Theorem 5 (Informal).** *For each  $\epsilon > 0$ , there is a pseudo-random generator from  $n^\epsilon$  bits to  $n$  bits computable in time  $2^{n^\epsilon} \text{poly}(n)$  secure against  $\text{poly}(n)$  size circuits iff for each  $c > 1/2$  there are no polynomial-size circuits solving  $\text{Kt}$  on more than  $1 - 2^{-cn}$  fraction of inputs of length  $n$ .*

## 1.2.2 Application in Meta-Complexity: Robustness Theorems

We exploit the connection between  $\text{MKtP}$  and parallel cryptography to establish more robustness results for meta-complexity. It is known that parallel cryptography is *extremely* robust:  $\text{L}$ -computable one-way functions exist, if and only if  $\text{NC}^1$ -computable one-way

functions exist, if and only if  $\text{NC}^0$ -computable one-way functions exist [11]. We define  $\text{L}$ - and  $\text{NC}^1$ -variants of  $\text{K}^t$  complexity, and translate the result in [11] to the following robustness theorem:

► **Theorem 6** (Bounded-Error Robustness of Meta-Complexity; Informal). *The following statements are equivalent:*

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

It is natural to ask whether the above theorem can be interpreted as a *reduction*. Somewhat surprisingly, we show the answer is *yes!* We discover an average-case *reduction* from  $\text{L-K}^t$  to  $\text{MKTP}$ , as follows:

► **Theorem 7** (Informal). *Let  $n, t$  be parameters,  $m := \text{poly}(n, t)$ . There is a randomized reduction  $\text{Red}(x)$  that maps a length- $n$  input to a length- $m$  input, and satisfies the following property:*

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

To the best of our knowledge, this is the first reduction from a variant of  $\text{K}^t$  complexity to a variant of  $\text{KT}$  complexity. The only special property of  $\text{L}$  that we use is that  $\text{L}$ -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)  $\text{K}^t$  complexity to the  $\text{KT}$  complexity.

We have focused on the *bounded-error* average-case complexity of meta-complexity problems so far. However, Theorem 7 also implies robustness in the *zero-error* regime. Here, let  $\text{MKTP}[s]$  be the problem of determining whether the input  $x$  satisfies  $\text{KT}(x) \leq s(|x|)$ , and let  $\text{MINK}^t[s]$  be the problem of determining whether the input  $x$  satisfies  $\text{K}^t(x) \leq s(|x|)$ .

► **Theorem 8** (Zero-Error Robustness of Meta-Complexity; Informal). *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  $\text{NC}^1\text{-MINK}^{t_1}[n - c \log n]$  is zero-error easy on average.*
2. *There is a constant  $c > 0$  such that  $\text{L-MINK}^{t_2}[n - c \log n]$  is zero-error easy on average.*
3. *There is a constant  $c > 0$  such that  $\text{MKTP}[n - c \log n]$  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  $2^n/\text{poly}(n)$  time to invert (say, on a constant fraction of inputs)!

The results of [62] opens up the possibility to *characterize* the hardest one-way functions by the meta-complexity of Kolmogorov complexity. In particular, the existence of maximally hard one-way functions may be equivalent to the ‘‘Perebor’’ hypothesis, i.e. some meta-complexity problem requires brute force to solve.

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

► **Theorem 9 (Informal).** *For every constant  $\alpha > 0$ , there exists a weak one-way function with hardness  $2^{(1-o(1))\alpha n}$  if and only if  $K^t$  complexity is hard on average for algorithms of size  $2^{(1-o(1))\alpha n}$ .*

Note that the two  $\alpha$ '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.

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

► **Theorem 10 (Informal).** *If there is a weak one-way function in  $\text{NC}^0$  with hardness  $2^{\Omega(n)}$ , then KT requires  $2^{\Omega(n)}$  size to solve on average.*

Finally, we put forward a few Peregory Hypotheses. These hypotheses assert brute-force search is unavoidable for solving meta-complexity problems such as  $K^t$  and KT, and are closely related to the maximum hardness of (weak) one-way functions. See Section 5.5 for more details.

### 1.3 Related Work

There have been several previous works connecting meta-complexity to cryptography. Impagliazzo and Levin [51] show that the existence of one-way functions is equivalent to the hardness of a certain learning task related to time-bounded Kolmogorov complexity. Oliveira and Santhanam [72] 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 [76] 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.

None of the above results gives an unconditional equivalence between the average-case hardness of a natural decision problem and the existence of one-way functions. This was finally achieved by Liu and Pass [62], who showed that the weak hardness of  $K^{\text{poly}}$  over the uniform distribution is equivalent to the existence of one-way functions. [62] 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  $\text{NC}^1$ , which by [11] is equivalent to the existence of one-way functions in  $\text{NC}^0$ .

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 first studied explicitly in [43]. [43] essentially observe that the identity reduction trivially reduces  $\mu$  to  $\mu'$  over the uniform distribution in a zero-error average-case sense, where  $\mu$  and  $\mu'$  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 [63] and [5].** We now discuss the relationship of our work with the concurrent works of [63] and [5], which overlap in some respects with ours.

Liu and Pass [63] show an equivalence between the bounded-error weak average-case hardness of  $K^t$  over the uniform distribution and the existence of one-way functions - this is essentially the same as our Theorem 4. They also show that the *zero-error* average-case hardness of  $K^t$  over the uniform distribution is equivalent to  $\text{EXP} \neq \text{BPP}$ . In contrast, our Theorem 5 gives an equivalence between the bounded-error average-case hardness of  $K^t$  over the uniform distribution in a different parametric regime and the worst-case hardness of  $\text{EXP}$ , 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  $\text{EXP}$  implies the existence of one-way functions.

[63] 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  $\text{KT}$  or  $\text{MCSP}$ .

Allender, Cheraghchi, Myrasiotis, Tirumala, and Volkovich [5] relate the average-case hardness of the conditional version of  $\text{KT}$  complexity over the uniform distribution to the existence of one-way functions. They show that if the conditional version 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 the conditional version of  $\text{KT}$  is hard on an exponential fraction of instances. In contrast, we *characterize* parallel cryptography by the average-case hardness of  $\text{KT}$ .

## 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  $\text{NC}^0$ -computable one-way functions and the hardness of  $\text{KT}$  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 Pereg Hypotheses in Section 5.5, which are closely related to the existence of maximally-hard one-way functions. The results related to  $\text{MCSP}$  are proved in Section 6, and the results related to  $\text{MKtP}$  are proved in Section 7. Finally, we leave a few open questions in Section 8.

## 2 Intuitions and Techniques

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

### 2.1 Parallel Cryptography and the Hardness of $\text{KT}$

Our proof of Theorem 1 builds on [62]. However, it turns out that we need new ideas for both directions of the equivalence.

#### 2.1.1 Hardness of $\text{KT}$ from One-Way Functions in $\text{NC}^0$

We first review how Liu and Pass [62] proved that one-way functions imply average-case hardness of  $K^t$ .



Any cryptographically-secure PRG  $G$  implies *zero-error* hardness of  $K^t$  [74, 58, 4]. Roughly speaking, the outputs of  $G$  have “non-trivial”  $K^t$  complexity, but random strings are likely to have “trivial”  $K^t$  complexity.<sup>7</sup> 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  $G$ ,<sup>8</sup> and used  $G$  to argue that  $K^t$  is bounded-error average-case hard. Roughly speaking, if the outputs of  $G$  occupy a  $1/\text{poly}(n)$  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.

Let  $t$  be the running time of  $G$  (which is a large polynomial). Let  $\text{out} := G(\text{seed})$  be any output of  $G$ , we can see that  $K^t(\text{out})$  is indeed non-trivial, as we can describe  $\text{seed}$  and the code of  $G$  with  $|\text{seed}| + O(1) < n$  bits. Given this description, we can “decompress”  $\text{out}$  in  $t(n)$  steps by computing  $G$  on  $\text{seed}$ . However,  $\text{KT}(\text{out})$  is the sum of the description length and the running time, which is  $|\text{seed}| + O(\log n) + t(n) \gg n$ . This is even worse than the trivial description for  $\text{out}$  whose complexity is  $n + O(\log n)$ .

One attempt is to pad both the seed and the output by a random string of length  $\text{poly}(t(n))$ , so that  $G$  becomes *sublinear-time* computable. That is,  $G'(\text{seed} \circ r) = \text{out} \circ r$  where  $r$  is a long string. Still, we only have  $\text{KT}(\text{out} \circ r) \leq |\text{seed}| + |r| + t(n)$ , but the trivial upper bound for  $\text{KT}(\text{out} \circ r)$  is only  $|\text{out}| + |r|$ . If  $t(n)$  is larger than the stretch of  $G$  (i.e.,  $|\text{out}| - |\text{seed}|$ ), then we do not have non-trivial KT-complexity upper bounds on outputs of  $G$ .

This problem is inherent as we need  $G$  to be *dense*. Suppose that the number of possible outputs of  $G$  is  $2^n/\text{poly}(n)$ , then there must be an output of  $G$  whose *Kolmogorov* complexity is at least  $n - O(\log n)$ . That is, the seed length of  $G$  has to be  $n - O(\log n)$ , even if we place no restrictions on the complexity of  $G$ ! Now, if we want the outputs of  $G$  to have non-trivial KT complexity, we only have  $O(\log n)$  time to compute each output bit of  $G$ . Therefore,  $G$  is a PRG in constant parallel time.<sup>9</sup>

<sup>7</sup> 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”).

<sup>8</sup> The input distribution of their PRG is not the uniform distribution, which is different from standard PRGs; see Definition 24. We ignore this difference in the informal exposition.

<sup>9</sup> 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.

## 35:10 Hardness of KT Characterizes Parallel Cryptography

We discovered that the (bounded-error) average-case complexity of KT is related to *cryptography in  $\text{NC}^0$* . Now it is easy to see that  $\text{NC}^0$ -computable dense PRGs imply bounded-error hardness of KT complexity. We can construct such a PRG from  $\text{NC}^0$ -computable one-way functions, as follows.<sup>10</sup> We first use [62] to construct a dense PRG  $G$ . This PRG is not necessarily in  $\text{NC}^0$ , as [62] needs some more complex primitives (e.g. extractors). Nevertheless, we can apply the randomized encodings in [11] to compile  $G$  into a PRG in  $\text{NC}^0$ .

### 2.1.2 One-Way Functions in $\text{NC}^0$ from Hardness of KT

It is straightforward to construct a one-way function from hardness of KT, using techniques of [62, 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  $(\ell, x)$ , is required to find a description of  $x$  with complexity at most  $\ell$ , thus it needs to solve MKTP. (All one-way functions in this section are *weak*, meaning they cannot be inverted efficiently on a  $1 - 1/\text{poly}(n)$  fraction of inputs.)

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

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  $\text{KT}(x) = |d| + t$  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  $t \leq c \log n$  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  $\text{NC}^0$  by [11].

## 2.2 Applebaum-Ishai-Kushilevitz as a Reduction

For any “reasonable” circuit class  $\mathcal{C}$ , we can use [62] to show that the existence of one-way functions computable in  $\mathcal{C}$  is equivalent to the hardness of  $\mathcal{C}\text{-K}^t$ . (The precise definition of  $\mathcal{C}\text{-K}^t$  is beyond the scope of this paper, but  $\text{NC}^1\text{-K}^t$  and  $\text{L}\text{-K}^t$  are defined in Definition 15.) Now, let us review the main results of [11]:  $\oplus\text{L}$ -computable one-way functions exist if and only if  $\text{NC}^0$ -computable one-way functions exist. In other words,  $\oplus\text{L}\text{-K}^t$  is hard on average if and only if  $\text{NC}^0\text{-K}^t$  is hard on average!<sup>11</sup>

It is natural to ask whether there is a reduction between  $\oplus\text{L}\text{-K}^t$  and  $\text{NC}^0\text{-K}^t$ . 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  $\oplus\text{L}\text{-K}^t$  complexity to a string with non-trivial  $\text{NC}^0\text{-K}^t$  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  $\oplus\text{L}\text{-K}^t$  and  $\text{NC}^0\text{-K}^t$ .

---

<sup>10</sup>Note that this is different from [38, 37]. The PRG we construct is dense, but its input distribution is not uniform. In contrast, [38, 37] constructs a PRG (on uniformly random inputs) from an arbitrary one-way function, but the PRG is not necessarily dense.

<sup>11</sup> $\text{NC}^0$  may not be reasonable in the above sense, but the reduction we present in this section is correct.

The property that enables our reduction is *resamplability* [26]. For now, think of “easy” as being  $\text{NC}^0$ -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 uniform random input of  $f$  whose answer is the same as  $x$ .

► **Example 11.** The parity function  $\text{PARITY}(x) = x_1 \oplus x_2 \oplus \dots \oplus x_n$  is hard (i.e., not computable in  $\text{NC}^0$ ). Given  $n$  input bits  $x_1, x_2, \dots, x_n$  and  $n-1$  random bits  $r_1, r_2, \dots, r_{n-1}$ , we can produce a uniform random input whose answer is the same as  $x$ , as follows:

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

Note that the resampler is easy (i.e., in  $\text{NC}^0$ ), thus parity is resamplable.

**The reduction.** We will use a  $\oplus\text{L}$ -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  $N = \text{poly}(n)$ , 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  $\oplus\text{L-K}^t(x) = n - \gamma$  is non-trivial, and  $d$  is a  $\oplus\text{L}$  machine of description length  $n - \gamma$  that “computes”  $x$ . Since DCMD is  $\oplus\text{L}$ -complete (under  $\text{NC}^0$ -reductions), for each  $i$ , the computation of  $x_i$  can be reduced to a DCMD-instance  $s_i$  of length  $N$  such that  $\text{DCMD}(s_i) = x_i$ . Moreover, given the description  $d$ , we can produce  $s_1 \circ s_2 \circ \dots \circ s_n$  in  $\text{NC}^0$ .

We use the *resamplability* of DCMD. The resampler for DCMD only uses  $N - 1$  random bits (which is optimal). Consider the following  $\text{NC}^0$  circuit. It receives  $d$  and  $r_1, r_2, \dots, r_n$  as inputs, where each  $r_i$  is a random string of length  $N - 1$ . It computes  $s_1, s_2, \dots, s_n$  from  $d$ , and for each  $i$ , feeds  $s_i$  and  $r_i$  to the resampler to obtain a uniform random DCMD instance whose answer is the same as  $s_i$ . When  $r_i$  are random bits, the output distribution of this  $\text{NC}^0$  circuit is identical to the distribution of  $\text{NC}^0\text{-K}^t$  instances we reduced  $x$  to. Moreover, the  $\text{NC}^0\text{-K}^t$  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.<sup>12</sup>

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

## 2.3 Tighter Connections

To obtain a tight relationship between hardness of  $\text{K}^t$  and hardness of weak one-way functions, we optimize the construction from one-way functions to PRGs in [62]. Suppose that given a one-way function  $f$  with input length  $n$ , we could construct a PRG with output length  $m'$ . Then solving  $\text{K}^t$  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*  $\text{Ext}(\mathcal{X}, r)$  [70, 68], which given a “somewhat random” distribution  $\mathcal{X}$  and a truly random seed  $r$ , outputs a distribution that is statistically close to the uniform random distribution.

We use the near-optimal explicit extractors with  $O(\log^2 n)$  seed length [36].

<sup>12</sup>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*  $\text{HC}(x, r)$  [32]. 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  $\text{HC}(x, r)$  and a uniformly random string. Note that  $\text{HC}(x, r)$  needs to have multiple output bits; in contrast, a *hardcore predicate* (also defined in [32]) only has one output bit.

We use the observation, implicit in [82, 79], that any seed-extending “black-box” pseudorandom generator is a good hardcore function. We use the *direct product* generator [42, 41] 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: [62] 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 [92] 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* [49] 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 [62]; see Footnote 8) 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 has 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 function to the hardness of  $K^t$  at almost no cost.

**Tighter connections between MKTP and one-way functions in  $\text{NC}^0$ .** Here, the problem becomes to construct  $\text{NC}^0$ -computable PRGs from  $\text{NC}^0$ -computable one-way functions. We use a construction of universal hash functions in  $\text{NC}^0$  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 [15, 44]). As the hash functions require linear seed length, from an  $\text{NC}^0$ -computable one-way function with input length  $n$ , we obtain an  $\text{NC}^0$ -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  $2^{-n}/\text{poly}(n)$ , 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 [28], 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  $\text{NC}^0$ ; see Remark 76 for details.

**Hardness of MCSP from one-way functions in  $\text{NC}^0$ .** 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 [83, 84] 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  $\epsilon n$  for some constant  $\epsilon > 0$ . This is why we need the hardness of the one-way function in our assumption to be at least  $\text{poly}(2^{\epsilon n})$ .

## 2.5 Using Hardness of Kt to Capture Cryptographic and Complexity-Theoretic Pseudorandomness

To show Theorem 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  $\text{Kt}(x) = |d| + \log t$ , we have  $t \leq \text{poly}(n)$ . Our one-way function receives two inputs  $d, t$ , where  $d$  is the description of a machine, and  $t \leq \text{poly}(n)$  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. The proof that one-way functions imply the average-case hardness of Kt complexity mimics the proof of the corresponding implication in Theorem 2, since the outputs of a cryptographic PRG with stretch  $\lambda \log n$  have non-trivial Kt complexity when  $\lambda$  is large enough compared to the time required to compute the PRG.

To show Theorem 5, we use the Nisan-Wigderson generator [69] 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 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  $\text{negl} : \mathbb{N} \rightarrow [0, 1]$  is *negligible* if for every constant  $c$ ,  $\text{negl}(n) \leq 1/n^c$  for large enough integers  $n$ .

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

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

Otherwise we say  $X$  and  $Y$  are  $\epsilon$ -indistinguishable by  $D$ .

We often consider ensemble of functions in this paper. For example, a function  $f : \{0, 1\}^* \rightarrow \{0, 1\}^*$  can be interpreted as an ensemble  $f = \{f_n : \{0, 1\}^n \rightarrow \{0, 1\}^*\}$ , and each  $f_n$  is 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, so 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 analogue of  $\text{NC}^0$ .

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 12.** Let  $c > 0$  be a constant,  $p(\cdot)$  be a polynomial, and  $F = \{F_n : \{0, 1\}^n \rightarrow \{0, 1\}^{p(n)}\}$  be an ensemble of functions. We say  $F \in \text{TIME}[c \log n]$  if there is a Turing machine  $M$  with running time  $c \log n$  such that, for every  $x \in \{0, 1\}^n$  and  $1 \leq i \leq p(n)$ ,  $M^x(n, i)$  outputs the  $i$ -th bit of  $F_n(x)$ .

Let  $\text{DLOGTIME} = \bigcup_{c \geq 1} \text{TIME}[c \log n]$ .

► **Definition 13.** Let  $c > 0$  be a constant,  $p(\cdot)$  be a polynomial, and  $F = \{F_n : \{0, 1\}^n \rightarrow \{0, 1\}^{p(n)}\}$  be an ensemble of functions.

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

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

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

For the readers not familiar with parity Turing machines, we provide an alternative definition of  $\oplus \text{L}$  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  $\text{K}$ , the  $\text{KT}$  complexity [4], the time-bounded Kolmogorov complexity  $\text{K}^t$  [59], and Levin's  $\text{Kt}$  complexity [60]. Then we define the  $\text{NC}^1$ - and  $\text{L}$ -versions of  $\text{K}^t$ .

► **Definition 14.** Let  $U$  be a Turing machine,  $x$  be a string. Artificially let  $x_{|x|+1} = \star$ .

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

► **Definition 15.** 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  $t : \mathbb{N} \rightarrow \mathbb{N}$  be a resource bound.  $\text{NC}^1\text{-K}_{U_a}^t(x)$  is the minimum value of  $|d|$  such that for every  $1 \leq i \leq |x| + 1$  and  $b \in \{0, 1, \star\}$ ,  $U_a^d(i, b)$  accepts in alternating time  $\log t(|x|)$  if and only if  $x_i = b$ .
- Let  $t : \mathbb{N} \rightarrow \mathbb{N}$  be a resource bound.  $\text{L-K}_{U_s}^t(x)$  is the minimum value of  $|d|$  such that for every  $1 \leq i \leq |x| + 1$  and  $b \in \{0, 1, \star\}$ ,  $U_s^d(i, b)$  accepts in space  $\log t(|x|)$  if and only if  $x_i = b$ .

Our results hold for every efficient enough universal Turing machine  $U$ . Therefore, in this paper, we drop the subscript  $U$  and simply write  $\text{KT}$ ,  $\text{K}^t$ , etc.

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

► **Definition 16.** Let  $N = 2^n$ ,  $tt \in \{0, 1\}^N$  be a truth table that corresponds to a function  $f : \{0, 1\}^n \rightarrow \{0, 1\}$ . We define  $\text{Size}(tt)$  as the size (number of gates) of the smallest circuit that computes  $f$ .

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

## 35:16 Hardness of KT Characterizes Parallel Cryptography

► **Definition 17.** We define the following problems:

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

There are natural search versions for the problems above. The search version for MKTP is to find an optimal description  $d$  for  $x$ , such that  $x$  can be generated from  $d$  implicitly in time at most  $k - |d|$ . The search version for MINKT is to find an optimal 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 18** ([4, Proposition 13]). *There is an absolute constant  $c' > 0$  such that  $\text{KT}(x) \leq |x| + c' \log |x|$  for every string  $x$ .*

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

► **Fact 19.** *Let  $n$  be an integer,  $s \leq n - 1$ , then*

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

**Proof Sketch.** The number of strings  $x$  such that  $\text{K}(x) \leq s$  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  $\text{H}(X)$ , is defined as

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

The *min-entropy* of a random variable  $X$ , denoted as  $\text{H}_\infty(X)$ , 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  $\text{SD}(X, Y)$ , is defined as

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

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

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



### 3.4 Bounded-Error Average-Case Hardness

We define the (bounded-error) average-case hardness of a function  $f$ . (Think of  $f = \text{KT}$  or  $\text{K}^t$ .) 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 20.** Let  $f : \{0, 1\}^* \rightarrow \mathbb{N}$  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<sup>13</sup> machine  $\mathcal{A}$  and every large enough input length  $n$ ,

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

- 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  $n$ ,

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

### 3.5 One-Way Functions

We recall the standard definition of one-way functions and weak one-way functions.

► **Definition 21 (One-Way Functions).** Let  $f : \{0, 1\}^* \rightarrow \{0, 1\}^*$  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  $n \in \mathbb{N}$ ,

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

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  $1 - \text{negl}(n)$  fraction of inputs.

► **Definition 22 (Weak One-Way Functions).** Let  $f : \{0, 1\}^* \rightarrow \{0, 1\}^*$  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  $n \in \mathbb{N}$ ,

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

By a standard padding trick (see e.g., [30]), we can assume that (weak or strong) one-way functions are *length-preserving*, i.e. for every input  $x \in \{0, 1\}^*$ ,  $|f(x)| = |x|$ . In this paper, we will implicitly assume that *every one-way function is length-preserving*.

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

► **Theorem 23 ([92, 30]).** 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.

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

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

<sup>13</sup>PPT stands for probabilistic polynomial-time.

### 3.6 Conditionally Secure Entropy-Preserving PRGs

Here we define conditionally secure entropy-preserving PRGs (condEP-PRGs), introduced in [62].

A *pseudorandom generator*, according to the standard definition, is a polynomial-time computable function  $G : \{0, 1\}^n \rightarrow \{0, 1\}^m$  (where  $m > n$ ), such that  $G(\mathcal{U}_n)$  and  $\mathcal{U}_m$  are computationally indistinguishable. Compared with standard PRGs, a *condEP-PRG*  $G : \{0, 1\}^n \rightarrow \{0, 1\}^m$  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  $\log |\mathcal{E}_n| \leq n \leq m$ . 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 24** (Conditionally Secure Entropy-Preserving PRG, abbr. condEP-PRG, [62]). *Let  $\gamma > 0$  be a constant, and  $p(\cdot)$  be a polynomial. Consider a polynomial-time computable ensemble of functions  $G = \{G_n : \{0, 1\}^n \rightarrow \{0, 1\}^{n+\gamma \log n}\}$ . 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.*

1. (*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  $n$ ,

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

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

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

► **Theorem 25** ([62]). *There is a function EP-PRG computable in ALOGTIME, such that the following holds. For any one-way function  $f : \{0, 1\}^* \rightarrow \{0, 1\}^*$  and any constant  $\gamma > 0$ , let  $G(x, z) = \text{EP-PRG}(\gamma, x, f(x), z)$ , then  $G$  is a condEP-PRG with stretch  $\gamma \log n$  and security  $1/n^\gamma$ .*

► **Remark 26.** It is important that the machine EP-PRG is fixed and does not depend on the constant  $\gamma$ . Suppose there is an absolute constant  $c > 0$  such that for every  $\gamma > 0$ , there is a PRG  $G_\gamma$  that runs in  $\text{TIME}[c \log n]$  and stretches  $n$  bits into  $n + \gamma \log n$  bits. The outputs of  $G_\gamma$  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  $G_\gamma$  from truly random strings. It follows that we can use such  $G_\gamma$  to argue about the hardness of MKTP. On the other hand, if the time complexity of  $G_\gamma$  depends on  $\gamma$ , it does not necessarily imply any hardness of MKTP.

### 3.7 Complete Problems for $\oplus\text{L}$

We introduce the  $\oplus\text{L}$ -complete problems, called Connected Matrix Determinant (CMD) and Decomposed Connected Matrix Determinant (DCMD), that will be crucial to us. Originally motivated by secure multi-party computation [54, 55], these problems have found surprisingly many applications in cryptography and complexity theory [11, 33, 26, 43, 23].

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

► **Definition 27** (See e.g., [23]). *An instance of CMD is an  $n \times n$  matrix over  $\text{GF}(2)$  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  $\text{GF}(2)$ ):*

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

The instance is an  $(n(n+1)/2)$ -bit string specifying elements on and above the main diagonal. We define  $x \in \text{CMD}$  if and only if the determinant (over  $\text{GF}(2)$ ) 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  $x$ ,  $\text{DCMD}(x)$  is computed as follows: we partition  $x$  into blocks of length  $n^2$ , let  $y_i$  ( $1 \leq i \leq n(n+1)/2$ ) be the parity of the  $i$ -th block, and define  $\text{DCMD}(x) := \text{CMD}(y_1 \circ y_2 \circ \cdots \circ y_{n(n+1)/2})$ .

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

► **Theorem 28** ([11]). *Let  $n$  be an integer. There is a function  $P_{\text{CMD}} : \{0, 1\}^{\ell_{\text{CMD}}(n)} \times \{0, 1\}^{\ell_{\text{DCMD}}(n)-1} \rightarrow \{0, 1\}^{\ell_{\text{DCMD}}(n)}$ , computable in DLOGTIME, such that the following hold. For any input  $x \in \{0, 1\}^{\ell_{\text{CMD}}(n)}$ , the distribution of  $P_{\text{CMD}}(x, \mathcal{U}_{\ell_{\text{DCMD}}(n)-1})$  is equal to the uniform distribution over  $\{y : y \in \{0, 1\}^{\ell_{\text{DCMD}}(n)} : \text{DCMD}(y) = \text{CMD}(x)\}$ .*

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

► **Corollary 29.** *DCMD is balanced. In other words, for every integer  $n$ , the number of Yes instances and No instances of DCMD on input length  $\ell_{\text{DCMD}}(n)$  are the same.*

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

► **Theorem 30** ([54, 55]). *CMD is  $\oplus\text{L}$ -complete under projections.<sup>14</sup>*

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

► **Remark 31.** A proof of Theorem 30 can be found in [23, Section B.1]. However, the proof in [23] 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  $(\sigma_1, \sigma_2, \dots, \sigma_m$  in [23, Section B.1]), which may not be computable in DLOGTIME.

<sup>14</sup> A *projection* is a (multi-output) function where each output bit either is a constant, or only depends on one input bit.

## 35:20 Hardness of KT Characterizes Parallel Cryptography

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  $G^{\text{new}}$  has a vertex  $(i, v)$  for every  $0 \leq i \leq |V|$  and  $v \in V$ , and has edges from  $(i, u)$  to  $(i+1, v)$  for every edge  $(u, v) \in G$  and every  $0 \leq i < |V|$ . Let  $V = \{v_1, \dots, v_n\}$ , then

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

is a valid topological ordering of  $G^{\text{new}}$ . Now we can use [23, Section B.1] to reduce the computation of  $G^{\text{new}}$  to CMD by a DLOGTIME-uniform projection.

We would like to thank Yanyi Liu for pointing out this issue.

Theorem 28 and 30 implies the following beautiful result in [11].

► **Theorem 32** ([11]). *Suppose there is a one-way function computable in  $\oplus\text{L}$ . Then there is a one-way function computable in DLOGTIME.*

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

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

It turns out that the function  $g(x, y) = g(x, y, 1) \circ \dots \circ g(x, y, n)$  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  $\text{NC}^1$ ,  $\text{L}$ , or  $\oplus\text{L}$ .

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

### 4.1 One-Way Functions in $\text{NC}^0$ from Hardness of MKTP

► **Theorem 33.** *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 23, there is a one-way function in logarithmic space, and by Theorem 32, there is a one-way function in DLOGTIME.

Suppose KT is bounded-error hard on average. By Definition 20, 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  $\text{KT}(x)$  (Definition 14). Formally,  $t(x)$  is the smallest integer  $t$  such that there is a description  $d$  of length  $\text{KT}(x) - t$ , such that for every  $1 \leq i \leq |x|$  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 18, such that for every  $x \in \{0, 1\}^n$ ,  $\text{KT}(x) \leq |x| + c_1 \log |x|$ .

▷ **Claim 34.** For all but an  $1/n^{c+1}$  fraction of strings  $x \in \{0,1\}^n$ , we have  $t(x) \leq (c + c_1 + 2) \log n$ .

*Proof of Claim 34.* By Fact 18, for every  $x \in \{0,1\}^n$ ,  $\text{KT}(x) \leq n + c_1 \log n$ . By Fact 19, all but a  $1/n^{c+1}$  fraction of strings  $x \in \{0,1\}^n$  satisfies that  $\text{K}(x) > n - (c + 1) \log n - 1$ . For such strings  $x$ , we have  $t(x) \leq \text{KT}(x) - \text{K}(x) \leq (c + c_1 + 2) \log n$ . ◁

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

We define a weak one-way function  $f$  as follows.

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

---

```

1: function  $f(\ell, t, M)$ 
2:   The input consists of integers  $\ell \in [n + c_1 \log n]$ ,  $t \in [(c + c_1 + 2) \log n]$ , and a string
    $M \in \{0, 1\}^{n+c_1 \log n}$ .
3:    $M' \leftarrow$  the first  $\ell$  bits of  $M$ 
4:    $out \leftarrow \text{Output}(M', t)$ 
5:   if  $|out| = n$  then
6:     return the concatenation of  $\ell$ ,  $t$ , and  $out$ 
7:   else
8:     return  $\perp$ 

```

---

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

Let  $\mathcal{D}_{\text{owf}}$  be the output distribution of  $f$  on uniform inputs. In other words, to sample from  $\mathcal{D}_{\text{owf}}$ , 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  $\mathcal{D}_{\text{owf}}$  *almost dominates* the uniform distribution over  $\{0, 1\}^n$ , in the following sense.

▷ **Claim 35.** 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  $\mathcal{D}_{\text{owf}}$  is equal to  $(\text{KT}(x) - t(x), t(x), x)$  is at least  $\frac{1}{2^n n^{2+c_1}}$ .

*Proof of Claim 35.* For a large enough  $n$ , with probability at least  $\frac{1}{n^2}$ , the sampler for  $\mathcal{D}_{\text{owf}}$  samples  $t = t(x)$  and  $\ell = \text{KT}(x) - t(x)$ . Then, with probability  $\frac{1}{2^\ell} \geq \frac{1}{2^n n^{c_1}}$ , the sampler samples a description  $M'$  such that  $\text{Output}(M', t) = x$ . It follows that w.p. at least  $\frac{1}{2^n n^{2+c_1}}$  the sampler outputs  $(\text{KT}(x) - t(x), t(x), x)$ . ◁

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

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

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

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

## 35:22 Hardness of KT Characterizes Parallel Cryptography

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

---

```

1: function  $\mathcal{A}_{\text{KT}}(x)$ 
2:    $n \leftarrow |x|$ ;  $\text{OPT} \leftarrow +\infty$ ;  $\text{WITNESS} \leftarrow \perp$ 
3:   for  $\ell \in [n + c_1 \log n]$  and  $t \in [(c + c_1) \log n]$  do
4:      $(\ell', t', M) \leftarrow \mathcal{A}_{\text{owf}}(\ell, t, x)$ 
5:      $M' \leftarrow$  the first  $\ell'$  bits of  $M$ 
6:     if  $\text{Output}(M', t') = x$  and  $\text{OPT} > |M'| + t'$  then
7:        $\text{OPT} \leftarrow |M'| + t'$ 
8:        $\text{WITNESS} \leftarrow (M', t')$ 
9:   return  $(\text{OPT}, \text{WITNESS})$ 

```

---

$$\begin{aligned}
p &\geq \sum_{\substack{x \in \{0,1\}^n \\ t(x) \leq (c+c_1+2) \log n}} \Pr_{\mathbf{y} \leftarrow \mathcal{D}_{\text{owf}}} [\mathbf{y} = (\text{KT}(x) - t(x), t(x), x)] \cdot p_{\text{fail}}(x) \\
&\geq \sum_{\substack{x \in \{0,1\}^n \\ t(x) \leq (c+c_1+2) \log n}} \frac{1}{2^n n^{2+c_1}} \cdot p_{\text{fail}}(x) && \text{(Claim 35)} \\
&\geq \frac{1}{2^n n^{2+c_1}} \left( \sum_{x \in \{0,1\}^n} p_{\text{fail}}(x) - \frac{2^n}{n^{c+1}} \right) && \text{(Claim 34)} \\
&\geq \frac{1}{n^{2+c_1}} \left( \mathbb{E}_{\mathbf{x} \leftarrow \mathcal{U}_n} p_{\text{fail}}(\mathbf{x}) \right) - \frac{1}{n^{c+c_1+3}} \\
&\geq \frac{1}{n^{2+c_1+c}} - \frac{1}{n^{c+c_1+3}}. && \text{By (1)}
\end{aligned}$$

Let  $c' = c + c_1 + 4$ , then every PPT adversary  $\mathcal{A}_{\text{owf}}$  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 $\oplus\text{L}$

In this section, we prove the following theorem.

► **Theorem 36.** *Suppose there is a one-way function computable in  $\oplus\text{L}$ . 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  $\oplus\text{L}$ . The proof consists of three steps:

- First, we use  $f$  to build a condEP-PRG  $G$ . If  $f$  is computable in  $\oplus\text{L}$ , then  $G$  is also computable in  $\oplus\text{L}$ . This step is the same as in [62], and follows directly from Theorem 25.
- 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 37.
- 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 39.

### 4.2.1 CondEP-PRG in DLOGTIME

In this section, we prove the following lemma that constructs a DLOGTIME-computable condEP-PRG from a  $\oplus\text{L}$ -computable condEP-PRG.

► **Lemma 37.** *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  $\oplus\text{SPACE}[c \log n]$ .*

*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  $\text{TIME}[c' \log n]$ .*

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

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

$$G_1^{\text{proj}}, G_2^{\text{proj}}, \dots, G_\ell^{\text{proj}} : \{0, 1\}^n \rightarrow \{0, 1\}^{\ell_{\text{DCMD}}(N)},$$

such that the  $i$ -th output bit of  $G(x)$  is equal to  $\text{CMD}(G_i^{\text{proj}}(x))$ . Let  $r_1, r_2, \dots, r_\ell$  be random strings of length  $\ell_{\text{DCMD}}(N) - 1$ . Let  $P_{\text{CMD}}$  be the  $\text{DLOGTIME}$ -computable function defined in Theorem 28, then the  $i$ -th output bit of  $G(x)$  is equal to  $\text{DCMD}(P_{\text{CMD}}(G_i^{\text{proj}}(x), r_i))$ . We define

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

( $\tilde{G}$  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  $\tilde{G} \in \text{TIME}[c_g \log n]$ . Note that the input length of  $\tilde{G}$  is  $n_{\text{in}} := n + \ell \cdot (\ell_{\text{DCMD}}(N) - 1)$ , the output length of  $\tilde{G}$  is  $n_{\text{out}} := \ell \cdot \ell_{\text{DCMD}}(N)$ , and  $n_{\text{out}} = n_{\text{in}} + (\ell - n) = n_{\text{in}} + \lambda' \log n$ . Here, we fix  $\lambda'$  large enough such that  $\lambda' \geq \lambda \frac{\log n_{\text{in}}}{\log n}$ .

► **Claim 38.**  $\tilde{G}$  is a condEP-PRG with stretch  $\lambda \log n_{\text{in}}$  and security  $1/(n_{\text{in}})^\lambda$ .

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

Suppose  $\mathcal{E} = \{\mathcal{E}_n \subseteq \{0, 1\}^n\}$  is a sequence of events such that  $G$  and  $\mathcal{E}$  satisfy Definition 24. Let  $\tilde{\mathcal{E}} := \{\tilde{\mathcal{E}}_{n_{\text{in}}}\}$  where  $\tilde{\mathcal{E}}_{n_{\text{in}}} := \mathcal{E}_n \times \{0, 1\}^{\ell \cdot (\ell_{\text{DCMD}}(N) - 1)}$ . We verify that  $\tilde{G}$  and  $\tilde{\mathcal{E}}$  satisfy Definition 24.

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

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

Consider an adversary  $\mathcal{A}$  that distinguishes  $G(\mathcal{E}_n)$  from  $\mathcal{U}_\ell$  as follows. On input  $y$ , for every  $1 \leq i \leq \ell$ , let  $\mathbf{r}_i$  be a uniformly random length- $\ell_{\text{DCMD}}(N)$  input of  $\text{DCMD}$  such that  $\text{DCMD}(\mathbf{r}_i) = y_i$ . We concatenate them as  $\mathbf{r} = \mathbf{r}_1 \circ \mathbf{r}_2 \circ \dots \circ \mathbf{r}_\ell$ , 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  $\tilde{G}(\tilde{\mathcal{E}}_{n_{\text{in}}})$ . On the other hand, suppose  $\mathbf{y} \sim \mathcal{U}_\ell$ , then the distribution of  $\mathbf{r}$  is exactly  $\mathcal{U}_{n_{\text{out}}}$ . As  $\mathcal{A}'$  distinguishes  $\tilde{G}(\tilde{\mathcal{E}}_{n_{\text{in}}})$  from  $\mathcal{U}_{n_{\text{out}}}$  with advantage  $\geq 1/(n_{\text{in}})^\lambda$ , we can see that  $\mathcal{A}$  also distinguishes  $G(\mathcal{E}_n)$  from  $\mathcal{U}_\ell$  with advantage  $\geq 1/(n_{\text{in}})^\lambda \geq 1/n^{\lambda'}$ , 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  $\ell_{\text{DCMD}}(N)$  such that  $\text{DCMD}(\mathbf{r}_i) = \mathbf{y}_i$  for every  $1 \leq i \leq \ell$ , and finally concatenate them as  $\mathbf{r} = \mathbf{r}_1 \circ \mathbf{r}_2 \circ \dots \circ \mathbf{r}_\ell$ . The distribution of  $\mathbf{r}$  is exactly  $\tilde{G}(\tilde{\mathcal{E}}_{n_{\text{in}}})$ . Therefore,

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

◁

We have only defined  $\tilde{G}$  and  $\tilde{\mathcal{E}}$  on input lengths of the form

$$n_{\text{in}}(n) = n + (n + \lambda' \log n)(\ell_{\text{DCMD}}(n^c) + 1).$$

However, it is straightforward to define  $\tilde{G}$  and  $\tilde{\mathcal{E}}$  on every input length. Let  $m$  be an input length,  $m' = n_{\text{in}}(n)$  be the largest number of the form  $n_{\text{in}}(n)$  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 39.** *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  $\text{TIME}[c \log n]$ .*

*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  $\text{TIME}[c \log n]$ . 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  $G \in \text{TIME}[c \log n]$ , 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  $x \in \{0, 1\}^n$ ,

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

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  $c_{\text{kt}}$  that we fix later, there is a PPT machine  $\mathcal{A}$  such that

$$\Pr_{\mathbf{y} \leftarrow \mathcal{U}_\ell} [\text{KT}(\mathbf{y}) \leq \mathcal{A}(\mathbf{y}) \leq \text{KT}(\mathbf{y}) + \lambda \log n] \geq 1 - \frac{1}{n^{c_{\text{kt}}}}. \quad (2)$$

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

*Proof.* By Fact 19, all but a  $\frac{1}{n^2}$  fraction of strings  $y \in \{0, 1\}^\ell$  satisfies that  $\text{K}(y) \geq \ell - 2 \log n$ . Therefore, for all but a  $(\frac{1}{n^2} + \frac{1}{n^{c_{\text{kt}}}})$  fraction of strings  $y \in \{0, 1\}^\ell$ , we have  $\mathcal{A}(y) \geq \text{KT}(y) \geq \text{K}(y) \geq \ell - 2 \log n$ . On these strings  $y$  we have  $\mathcal{A}'(y) = 1$ . ◀

► **Lemma 41.**  $\Pr_{\mathbf{y} \leftarrow G(\mathcal{E}_n)} [\mathcal{A}'(\mathbf{y}) = 1] \leq 1 - \frac{1}{n}$ .

*Proof.* Let  $H := \text{H}(G(\mathcal{E}_n))$ . Let  $d$  be the constant such that  $\ell - H \leq d \log n$ . The constant  $d$  does not depend on  $c_{\text{kt}}$ , which means we can set  $c_{\text{kt}} := d + 15$ .

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

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



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

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

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

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

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

By Equation (2),  $|\text{Err}| \leq 2^\ell / n^{c_{\text{kt}} - 4}$ . Therefore,

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

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  $y \in \text{Good} \setminus \text{Err}$ , we have  $\mathcal{A}'(y) = 0$  w.p. at least  $1 - 1/n^4$  over the internal randomness of  $\mathcal{A}'$ . It follows that

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

From the pseudorandomness of the condEP-PRG  $G$ , we conclude that  $\text{KT}$  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  $\text{KT}(x_1) - d \log m \leq \text{KT}(x) \leq \text{KT}(x_1) + d \log m$ . It follows that if we can approximate MKTP on input length  $m'$ , then we can also approximate MKTP on input length  $m$ .  $\blacktriangleleft$

### 4.2.3 Proof of Theorem 36

► **Theorem 36.** *Suppose there is a one-way function computable in  $\oplus\text{L}$ . Then for every constant  $\lambda > 0$ ,  $\text{KT}$  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  $\oplus\text{SPACE}[c \log n]$ . Let EP-PRG be the Turing machine guaranteed in Theorem 25. For every constant  $\lambda > 0$ , let  $G(x, z) = \text{EP-PRG}(\lambda, x, f(x), z)$ . Then there is a constant  $c_1$  only depending on  $c$  (not on  $\lambda$ ) such that  $G$  is computable in  $\oplus\text{SPACE}[c_1 \log n]$ . Moreover,  $G$  is a condEP-PRG with stretch  $\lambda \log n$  and security  $1/n^\lambda$ .

By Lemma 37, 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  $\text{TIME}[c_2 \log n]$ . By Lemma 39, for every constant  $\lambda > 0$ ,  $\text{KT}$  is bounded-error hard on average to approximate within an additive error of  $\lambda \log n$ .  $\blacktriangleleft$

### 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  $\text{KT}$ ,  $\text{NC}^1\text{-K}^t$ , and  $\text{L-K}^t$ , either all of them admit bounded-error polynomial-time heuristics, or none of them do. (See Section 3.2 for their definition.)

► **Theorem 42.** *The following are equivalent:*

1. *There is a one-way function computable in  $\oplus\text{L}$ .*
2. *There is a one-way function computable in  $\text{DLOGTIME}$ .*
3. *The search version of  $\text{KT}$  is hard on average.*
4. *For every constant  $\lambda > 0$ ,  $\text{KT}$  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  $\text{NC}^1\text{-K}^t$  is hard on average.*
6. *For every constant  $\lambda > 0$  and polynomial  $t(\cdot)$  such that  $t(n) > 2n$ ,  $\text{NC}^1\text{-K}^t$  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  $\text{L-K}^t$  is hard on average.*
8. *For every constant  $\lambda > 0$  and polynomial  $t(\cdot)$  such that  $t(n) > 2n$ ,  $\text{L-K}^t$  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 33.

(5)  $\implies$  (2) and (7)  $\implies$  (2): The construction from [62, Section 4] gives a one-way function computable in  $\text{ALOGTIME}$  (i.e., uniform  $\text{NC}^1$ ), based on the hardness of  $\text{NC}^1\text{-K}^t$ . By Theorem 32, there is a one-way function computable in  $\text{DLOGTIME}$ . The same argument works for  $\text{L-K}^t$ .

(1)  $\implies$  (4): Directly from Theorem 36.

(1)  $\implies$  (6): Consider the condEP-PRG  $G$  computable in  $\text{TIME}[c \log n]$  that we constructed in the proof of Theorem 36, where  $c$  is some constant. Let  $t'(n) := n^{O(c)}$ , for every  $x \in \{0, 1\}^n$  that in the range of  $G$ ,  $\text{NC}^1\text{-K}^{t'}(x) \leq n - \Theta(\log n)$ . It follows that there is a polynomial  $t'$  such that  $\text{NC}^1\text{-K}^{t'}$  is hard on average to approximate.

To prove that  $\text{NC}^1\text{-K}^t$  is hard on average to approximate for every polynomial  $t$ , we use a padding trick. (See also [62, Theorem 5.6].) Let  $\epsilon > 0$  be a small enough constant, and  $n_1 = n^\epsilon$ . 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  $\epsilon$  to be a small enough constant, we have  $\text{NC}^1\text{-K}^t(x) \leq n - \Theta(\log n)$ . Since  $G'$  is pseudorandom,  $\text{NC}^1\text{-K}^t$  is hard on average to approximate.

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

### 4.4 Zero-Error Average-Case Reductions

Our techniques actually imply reductions among  $\text{MKTP}$ ,  $\text{NC}^1\text{-MINKT}$ , and  $\text{L-MINKT}$ . 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  $\text{MINKT}$  and  $\text{MKTP}$ .

The standard definition of an average-case complexity class, such as  $\text{AvgZPP}$ , 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  $\text{AvgZPP}$  as a class of languages rather than (language, distribution) pairs.

► **Definition 43.** Let  $L$  be a language and  $\delta > 0$  be a constant. We say  $L \in \text{Avg}_\delta \text{ZPP}$  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  $x \in \{0, 1\}^*$  and  $r \in \{0, 1\}^{\text{poly}(|x|)}$ ,  $\mathcal{H}(x; r) \in \{L(x), \perp\}$ .
- For every integer  $n$ ,  $\Pr_{\mathbf{x} \leftarrow \mathcal{U}_n, \mathbf{r} \leftarrow \mathcal{U}_{\text{poly}(n)}}[\mathcal{H}(\mathbf{x}; \mathbf{r}) \neq \perp] \geq \delta$ .

Let  $\text{Avg}_{\Omega(1)} \text{ZPP} := \bigcup_{\delta > 0} \text{Avg}_\delta \text{ZPP}$ .

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

A language  $L$  is *sparse* if for every integer  $n$ ,  $\Pr_{\mathbf{x} \leftarrow \mathcal{U}_n}[\mathbf{x} \in L] \leq o(1)$ . From Fact 19, for every unbounded function  $f(n) = \omega(1)$ ,  $\text{MKTP}[n - f(n)]$  and  $\text{MINK}^{\text{poly}(n)}[n - f(n)]$  are sparse. In general, to solve a 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 44.** 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  $\text{Red} : \{0, 1\}^n \times \{0, 1\}^{p(n)} \rightarrow \{0, 1\}^{m(n)}$ , such that the following holds.

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

► **Remark 45.** 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  $\text{Red}(\mathcal{U}_n; \mathcal{U}_{p(n)})$ . (See [12, Definition 18.6].) Nevertheless, thanks to the perfect randomized encodings [11], we are able to design reductions as strong as Definition 44.

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  $\text{Avg}_{\Omega(1)} \text{ZPP}$ .

► **Fact 46.** Let  $L_1, L_2$  be two sparse problems. Suppose that there is a one-sided mapping reduction  $\text{Red}$  from  $L_1$  to  $L_2$ . If there is a constant  $\delta_2 > 0$  such that  $L_2 \in \text{Avg}_{\delta_2} \text{ZPP}$ , then there is a constant  $\delta_1 > 0$  such that  $L_1 \in \text{Avg}_{\delta_1} \text{ZPP}$ .

For every  $s_1(n) \leq s_2(n)$ , there is a one-sided mapping reduction from  $\text{MKTP}[s_1(n)]$  to  $\text{MKTP}[s_2(n)]$ . (The identity mapping is a valid reduction [43].) 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  $\text{NC}^1\text{-MINK}^{t_1}[s_1]$  to  $\text{L-MINK}^{t_2}[s_2]$ . (Again, the identity mapping is a valid reduction.)

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

► **Theorem 47.** For every polynomial  $t(\cdot)$  and integer  $c > 0$ , there is a constant  $c' > 0$  such that there is a one-sided mapping reduction  $\text{Red}$  from  $\text{L-MINK}^t[n - c' \log n]$  to  $\text{MINK}^{t'}[n - c \log n]$ .

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

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

Since DCMD is balanced (Corollary 29), 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  $\text{Red}(x; r) := s_1 \circ s_2 \circ \dots \circ s_n$ , where  $r$  is the random coins that our reduction uses. For  $t'(n) = \lambda \log n$ , we want to prove that  $\text{K}^{t'}(\text{Red}(x; r)) \leq \tilde{N} - c \log \tilde{N}$ .

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  $1 \leq i \leq n + 1$  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 30), for  $N = \text{poly}(t(n))$ , there is a DLOGTIME-computable projection

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

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

The description of  $\text{Red}(x; r)$  contains the string  $d$ , and  $n$  strings  $s'_1, s'_2, \dots, s'_n$  of length  $\ell_{\text{DCMD}}(N) - 1$  each. Let  $P_{\text{CMD}}$  be the DLOGTIME-computable projection in Theorem 28. 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  $1 \leq i \leq |\text{Red}(x; r)|$ . To compute the  $i$ -th bit of  $\text{Red}(x; r)$ , 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  $\text{Red}(x; r)$  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  $\text{L-K}^t(x) \leq n - c' \log n$ , regardless of the random bits  $r$  we choose, there is a description that allows us to quickly retrieve each bit of  $\text{Red}(x; r)$ . 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  $\text{Red}(x; r)$  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 48.) Therefore, Theorem 47 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 47 demonstrates the *robustness* of meta-complexity w.r.t. the zero-error average-case complexity. In particular:

► **Theorem 48.** *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  $\text{NC}^1\text{-MINK}^t[n - c \log n] \in \text{Avg}_{\Omega(1)}\text{ZPP}$ .*
2. *There is a constant  $c > 0$  such that  $\text{L-MINK}^t[n - c \log n] \in \text{Avg}_{\Omega(1)}\text{ZPP}$ .*
3. *There is a constant  $c > 0$  such that  $\text{MINK}^{t'}[n - c \log n] \in \text{Avg}_{\Omega(1)}\text{ZPP}$ , 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  $\text{MKTP}[n - c \log n] \in \text{Avg}_{\Omega(1)}\text{ZPP}$ .*

**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  $\text{K}^{t'}(x) \leq n - c' \log n$ . Since  $(d, t'(n))$  is also a witness that  $\text{KT}(x) \leq n - c \log n$ . we have  $x \in \text{NC}^1\text{-MKTP}[n - c \log n]$ .

(3)  $\implies$  (2): By Theorem 47 and Fact 46.

(3)  $\implies$  (1): Note that the only property of  $\text{L}$  used in the proof of Theorem 47 is that  $\text{CMD}$  is hard for  $\text{L}$ . (In other words,  $\text{L} \subseteq \oplus \text{L}$ .) As  $\text{CMD}$  is also hard for  $\text{NC}^1$ , the proof of Theorem 47 is also true for  $\text{L-MINKT}$  replaced by  $\text{NC}^1\text{-MINKT}$ .

(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  $\text{L-MINK}^{t_1}[n - c \log n]$ , where  $t_1(n) = 2^{O(t'(n))}$ . We can use a padding trick [62, Theorem 5.6] to reduce  $\text{L-MINK}^{t_1}$  to  $\text{L-MINK}^t$ .

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

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

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

► **Definition 49.** Let  $f : \{0, 1\}^n \rightarrow \{0, 1\}^n$  be a 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 50.** 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:

- There is a weak one-way function with security  $S(n) \cdot n^{\Theta(\log n)}$ .
- There are polynomials  $p, t$  such that the search version of  $\text{K}^t$  requires  $S(n) \cdot n^{\Theta(\log n)}$  size to compute on a  $1 - 1/p(n)$  fraction of inputs.
- For every constant  $\lambda > 0$ , there are polynomials  $p, t$ , such that  $\text{K}^t$  requires  $S(n) \cdot n^{\Theta(\log n)}$  size to  $(\lambda \log n)$ -approximate on a  $1 - 1/p(n)$  fraction of inputs.

► **Theorem 51.** Suppose there is a weak one-way function  $f$  with security  $2^{\Omega(n)}$  computable in  $\text{DLOGTIME}$ . Then there is a polynomial  $p$  such that  $\text{KT}$  requires  $2^{\Omega(n)}$  size to compute on a  $1 - 1/p(n)$  fraction of inputs.

► **Remark 52.** A few remarks are in order.

- In this section, we only consider non-uniform adversaries. The reason is that we will use Impagliazzo's hardcore lemma (Lemma 84) in the proof of Theorem 57, 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 [86] achieves  $t' = \text{poly}(t)$ , but we need  $t' = t \cdot \text{polylog}(t)$ . 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 [31]. Yao’s hardness amplification theorem (Theorem 23) 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 51) 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  $\beta$  that is much smaller than  $\alpha$ .
- The best seed length of known explicit extractors that extract all min-entropy is  $O(\log^2 n)$  [36]. This is why we see an  $n^{\Theta(\log n)}$  factor in Theorem 50.

We rely on the construction of condEP-PRGs from weak one-way functions in [93, 62], 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 [93, 62]. (The correctness of this construction is proved in Appendix A.) The proofs of Theorem 50 and 51 appear in Section 5.3 and 5.4 respectively.

## 5.1 Technical Building Blocks

### 5.1.1 Extractors

► **Definition 53.** A function  $\text{Ext} : \{0, 1\}^n \times \{0, 1\}^d \rightarrow \{0, 1\}^m$  is a  $(k, \epsilon)$ -extractor if for every random variable  $X$  over  $\{0, 1\}^n$  such that  $H_\infty(X) \geq k$ , the statistical distance between  $\text{Ext}(X, \mathcal{U}_d)$  and  $\mathcal{U}_m$  is at most  $\epsilon$ .

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

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

### 5.1.2 Hardcore Functions

► **Definition 54.** Let  $\epsilon = \epsilon(n) > 0$ ,  $L = L(n) \leq \text{poly}(n)$ ,  $\text{HC} : \{0, 1\}^n \times \{0, 1\}^d \rightarrow \{0, 1\}^m$  be a function, and  $R$  be a probabilistic oracle algorithm. We say  $\text{HC}$  is a hardcore function with reconstruction algorithm  $R$ , distinguishing probability  $\epsilon$ , and list size  $L$ , if the following holds.

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

Our definition of hardcore functions indeed implies the standard definition in [32]:

► **Fact 55.** Let  $\text{HC} : \{0, 1\}^n \times \{0, 1\}^d \rightarrow \{0, 1\}^m$  be a hardcore function with a  $\text{poly}(n)$ -time reconstruction algorithm, distinguishing probability  $\epsilon = 1/\text{poly}(n)$ , and list size  $L \leq \text{poly}(n)$ .

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  $2\epsilon$ -distinguish the distribution  $f(\mathbf{x}) \circ \mathbf{r} \circ \text{HC}(\mathbf{x}, \mathbf{r})$  from the distribution  $f(\mathbf{x}) \circ \mathbf{r} \circ \mathcal{U}_m$ .

**Proof.** Let  $\mathcal{A}$  be an adversary of size  $\text{poly}(n)$  that  $2\epsilon$ -distinguishes the distribution  $f(\mathbf{x}) \circ \mathbf{r} \circ \text{HC}(\mathbf{x}, \mathbf{r})$  from  $f(\mathbf{x}) \circ \mathbf{r} \circ \mathcal{U}_m$ . Say  $x \in \{0, 1\}^n$  is *good* if  $\mathcal{A}$  can  $\epsilon$ -distinguish  $f(x) \circ \mathbf{r} \circ \text{HC}(x, \mathbf{r})$  from  $f(x) \circ \mathbf{r} \circ \mathcal{U}_m$ . Then by a Markov bound, at least an  $\epsilon$  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  $(\epsilon/2) > 1/\text{poly}(n)$ , this contradicts the one-wayness of  $f$ .

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

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

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

## 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 56** ([93, 62]). Let  $0 < \epsilon < \frac{1}{10n^2}$  be the desired security parameter of the condEP-PRG (i.e., it should be  $O(\epsilon)$ -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  $(k, \epsilon)$ -extractor  $\text{Ext} : \{0, 1\}^n \times \{0, 1\}^d \rightarrow \{0, 1\}^m$  with optimal output length, where  $d := d_{\text{Ext}}(n, \epsilon)$  and  $m := k - 2 \log(1/\epsilon) - O(1)$ . We write the extractor as  $\text{Ext}^{(k)}$  if we need to emphasize the min-entropy parameter  $k$ .
- For  $k_{\text{HC}} := \alpha + \log(n/\delta) + 4 \log(1/\epsilon) + O(1)$ , a hardcore function  $\text{HC} : \{0, 1\}^n \times \{0, 1\}^{d'} \rightarrow \{0, 1\}^{k_{\text{HC}}}$  with  $\text{poly}(n/\epsilon)$ -time reconstruction algorithm  $R$ , distinguishing probability  $\epsilon$ , and list size  $L \leq \text{poly}(n/\epsilon)$ , where  $d' := d_{\text{HC}}(n, k_{\text{HC}}, \epsilon)$ .

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

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

► **Theorem 57.** Let  $\epsilon, \delta, \alpha, f$  be defined as in Construction 56. If  $\epsilon \geq 1/\text{poly}(n)$  and  $L \leq \text{poly}(n)$ , then there is a function  $r : \mathbb{N} \rightarrow \mathbb{N}$  such that  $G = \{G_{n,r(n)}\}_{n \in \mathbb{N}}$  is a condEP-PRG with stretch  $\alpha$  and security  $4\epsilon$ .

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

The proof basically follows from [62], and we present a self-contained proof in Appendix A. However, there are two major differences between our proof and the proof in [62]:

- We replace the extractors and hardcore functions with better constructions. In particular, our extractors and hardcore functions in Section 5.3 requires only  $O(\log^2 n)$  random bits.
- More importantly, in the very beginning, we need to transform the weak one-way function into a strong one. [62] uses hardness amplification (Theorem 23) to implement this step. However, Theorem 23 does not preserve *exponential* security, therefore we use *Impagliazzo's hardcore lemma* [49] 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 25

Theorem 57 immediately implies Theorem 25.

► **Theorem 25** ([62]). *There is a function EP-PRG computable in ALOGTIME, such that the following holds. For any one-way function  $f : \{0, 1\}^* \rightarrow \{0, 1\}^*$  and any constant  $\gamma > 0$ , let  $G(x, z) = \text{EP-PRG}(\gamma, x, f(x), z)$ , 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 [93, 62].

- The extractors are derived from the *leftover hash lemma* [38]. (See also [85, Theorem 6.18].) Let  $h : \{0, 1\}^n \times \{0, 1\}^d \rightarrow \{0, 1\}^m$  be a pairwise independent family of hash functions, where  $d = O(n + m)$ , then for every  $k, \epsilon$  such that  $m = k - 2 \log(1/\epsilon)$ ,  $h$  is also a strong  $(k, \epsilon)$ -extractor.

We instantiate the pairwise independent hash family by Toeplitz matrices.<sup>15</sup> 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  $1 \leq i \leq m$  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  $\text{GF}(2)$ . In other words,  $\text{Ext}(x, key)$  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  $\text{GL} : \{0, 1\}^n \times \{0, 1\}^d \rightarrow \{0, 1\}^k$  be the Goldreich-Levin hardcore function. In [32],  $\text{GL}$  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  $1 \leq i \leq k$ , the  $i$ -th output bit of  $\text{GL}(x, r)$  is the inner product of  $x$  and  $r_{i \sim (i+n-1)}$  in  $\text{GF}(2)$ . Also, it is shown in [32] that for every  $\epsilon > 0$ ,  $\text{GL}$  is a hardcore function with distinguishing probability  $\epsilon$  and list size  $\text{poly}(n \cdot 2^k/\epsilon)$ .

**Proof Sketch of Theorem 25.** We can plug the parameters  $\epsilon := \frac{1}{4n^\gamma}$ ,  $\alpha := \gamma \log n$ ,  $\delta := 1/2$  into Theorem 57. The list size of  $\text{GL}$  is  $L \leq \text{poly}(n)$ . Theorem 57 gives us a function  $r : \mathbb{N} \rightarrow \mathbb{N}$  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  $r \leq n$ , 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  $\gamma$ . 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 50

To prove Theorem 50, we replace the leftover hash lemma and  $\text{GL}$  by extractors and hardcore functions with very short seed length:

► **Theorem 58** ([36, Theorem 5.14]). *Let  $d_{\text{Ext}}(n, \epsilon) := O(\log n \cdot \log(n/\epsilon))$ , then for every  $1 \leq k \leq n$  and  $\epsilon > 0$ , there is a strong  $(k, \epsilon)$ -extractor  $\text{Ext} : \{0, 1\}^n \times \{0, 1\}^{d_{\text{Ext}}(n, \epsilon)} \rightarrow \{0, 1\}^m$ , where  $m = k - 2 \log(1/\epsilon) - O(1)$  is optimal.*

<sup>15</sup>An  $n \times m$  matrix  $M$  is *Toeplitz* if  $M_{i,j} = M_{i+1,j+1}$  holds for every  $1 \leq i < n$ ,  $1 \leq j < m$ . We can represent a Toeplitz matrix by  $n + m - 1$  elements, namely the elements in the first row and the first column.



We observe that the “ $k$ -wise direct product generator” used in [42, 41] is a good hardcore function:

► **Theorem 59.** *Let  $d_{\text{HC}}(n, k, \epsilon) := O(k \log(n/\epsilon))$ , then there is a hardcore function  $\text{HC} : \{0, 1\}^n \times \{0, 1\}^{d_{\text{HC}}(n, k, \epsilon)} \rightarrow \{0, 1\}^k$  with a  $\text{poly}(n2^k/\epsilon)$ -time reconstruction algorithm  $R$ , distinguishing probability  $\epsilon$ , and list size  $L \leq 2^k \cdot \text{poly}(k/\epsilon)$ .*

**Proof Sketch.** Consider the function  $\text{DP} : \{0, 1\}^n \times \{0, 1\}^d \rightarrow \{0, 1\}^{d+k}$  defined in [41, Theorem 7.1]. The first  $d$  bits of  $\text{DP}(x, z)$  is always equal to  $z$ , and we let  $\text{HC}(x, z)$  be the remaining  $k$  bits of  $\text{DP}(x, z)$ .

In [41], the reconstruction algorithm is stated as  $R^{\mathcal{O}} : \{0, 1\}^a \times \{0, 1\}^r \rightarrow \{0, 1\}^n$ . Here,  $a \leq k + O(\log(k/\epsilon))$  is the “advice complexity” of  $\text{DP}$ , the first  $a$  input bits correspond to the advice, and the remaining  $r = \text{poly}(n/\epsilon)$  input bits are random coins used by  $R$ . For every  $x \in \{0, 1\}^n$  and every oracle  $\mathcal{O}$  that  $\epsilon$ -distinguishes  $\text{DP}(x, \mathcal{U}_d)$  from  $\mathcal{U}_{d+k}$ , we have

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

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

Now we use Construction 56 to prove Theorem 50.

► **Theorem 50.** *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  $\text{K}^t$  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  $\text{K}^t$  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  $\text{K}^t$  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 [62, 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  $\text{K}^t$  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  $\epsilon := 1/n^2$ . Let  $\alpha := (\lambda + C) \log n$  be the stretch of the condEP-PRG we construct, where  $C$  is a large absolute constant. Let  $r : \mathbb{N} \rightarrow \mathbb{N}$  be any function. Consider the function  $G_{n, r(n)}$  in Construction 56, where the input length of  $G_{n, r(n)}$  is

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

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

Consider any sequence of subsets  $\mathcal{E} = \{\mathcal{E}_n \subseteq \{0, 1\}^{\tilde{n}}\}$  such that  $\text{H}(G_{n, r(n)}(\mathcal{E}_n)) \geq \tilde{n} - \Omega(\log n)$ . The same argument as in Lemma 39 shows that there is an adversary of size

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

that  $4\epsilon$ -distinguishes  $G_{n, r(n)}(\mathcal{E}_n)$  from the uniform distribution. It follows 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 51

To prove Theorem 51, we need a family of universal hash functions that admit very efficient randomized encodings, constructed in [56, 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 [44, 15]).

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 51, 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 [56].

### 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, with multiple output gates. Let  $C$  be such a circuit,  $X$  be a subset of input variables. (For example, let  $C : \{0, 1\}^n \times \{0, 1\}^d \rightarrow \{0, 1\}^m$ , 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  $C : \{0, 1\}^n \times \{0, 1\}^d \rightarrow \{0, 1\}^{d+m}$  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  $C$ .<sup>16</sup> Let  $s$  be the number of internal (i.e. non-input, non-output) gates of  $C$ . The randomized encoding of  $C$ , denoted as  $\tilde{C}$ , is a function  $\tilde{C} : \{0, 1\}^n \times \{0, 1\}^{d+s} \rightarrow \{0, 1\}^{d+m+s}$  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  $g_i = g_j \nabla g_k$  (where  $\nabla \in \{\text{AND}, \text{XOR}\}$ ), then the  $i$ -th output is  $r(g_i)$  XOR  $(r(g_j) \nabla r(g_k))$ .

### 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 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 DLOGTIME.

<sup>16</sup>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.

► **Definition 60** (Uniform Skew Circuits). Let  $C = \{C_n : \{0, 1\}^n \times \{0, 1\}^{d(n)} \rightarrow \{0, 1\}^{s(n)}\}$  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 61.** It may seem strange that we need to output not only predecessors but also successors of each gate. The reason is that in [56], 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 Appendix B.4 for details.

We need a family of universal hash functions  $\mathcal{H} = \{h_{n,m} : \{0, 1\}^n \times \{0, 1\}^k \rightarrow \{0, 1\}^m\}$  in [56], 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 62.** For every integer  $n, m$  where  $m = O(n)$ , there exists an integer  $k = O(n)$ , and a family of universal hash functions  $\{h_{n,m} : \{0, 1\}^n \times \{0, 1\}^k \rightarrow \{0, 1\}^m\}$ , 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 [56], 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 62 in Appendix B, with an emphasis on the uniformity of these circuits.

By the leftover hash lemma of [38],  $\{h_{n,m}\}$  is a strong  $(k, \epsilon)$ -extractor whenever  $m = k - 2\log(1/\epsilon)$ . It was proved by [15] (based on [44]) that  $\{h_{n,m}\}$  are good hardcore functions:

► **Lemma 63.** For every  $\epsilon > 0$ ,  $h_{n,m}$  is a hardcore function with distinguishing probability  $\epsilon$  and a reconstruction algorithm of  $\text{poly}(2^m \cdot n/\epsilon)$  time. (As a result, the list size is also  $\text{poly}(2^m \cdot n/\epsilon)$ .)

### 5.4.3 Proof of Theorem 51

► **Theorem 51.** 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  $\delta = 1/\text{poly}(n)$  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 56, to build a condEP-PRG  $G : \{0, 1\}^{n_1} \rightarrow \{0, 1\}^{n_1 + \alpha}$  with stretch  $\alpha := O(\log n)$  and security  $4\epsilon \leq 1/n^{10}$ . Here, since the seed length of  $h$  is  $O(n)$ , we have  $n_1 = O(n)$ . Moreover, by Theorem 57, the condEP-PRG is  $4\epsilon$ -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. We replace  $C$  by its randomized encoding to obtain another condEP-PRG  $\tilde{G} : \{0, 1\}^{n_2} \rightarrow \{0, 1\}^{n_2 + \alpha}$ , which is computable in DLOGTIME. Here, since the size of  $C$  is  $O(n)$ , we have  $n_2 = O(n)$ . As every output of  $\tilde{G}$  has non-trivial KT complexity, and  $\tilde{G}$  is  $4\epsilon$ -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 [52, 53, 20]. The *exponential time hypothesis* (ETH) asserts that 3-SAT requires  $2^{\epsilon n}$  time to solve, where  $\epsilon > 0$  is some absolute constant and  $n$  is the number of variables. The *strong exponential time hypothesis* (SETH) asserts that for *any* constant  $\epsilon > 0$ , CNF-SAT requires  $2^{(1-\epsilon)n}$  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 [91].

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 64** (Weak Perebor Hypothesis for  $K^t$ ). *There is 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 65** (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 50 shows that the non-uniform version of Hypothesis 64 is equivalent to the existence of exponentially-hard weak one-way functions (against non-uniform adversaries). Theorem 51 shows that the non-uniform version of Hypothesis 65 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 66** (Strong Perebor Hypothesis for  $K^t$ ). *There are polynomials  $t(n) \geq 2n$  and  $p(n)$ , such that for every constant  $\epsilon > 0$ , every probabilistic algorithm that runs in  $2^{(1-\epsilon)n}$  time and attempts to solve  $K^t$  fails w.p. at least  $1/p(n)$  over a uniformly random input.*

By Theorem 50, the non-uniform version of Hypothesis 66 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 [39], Fiat and Naor [27] showed that no such one-way function exists in the *non-uniform RAM* model. In particular, for *any* function  $f : \{0, 1\}^n \rightarrow \{0, 1\}^n$ , there is an algorithm that runs in  $2^{3n/4}$  time, with random access to an advice tape of length  $2^{3n/4}$ , 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 66 is false. To the best of our knowledge, (the uniform version of) Hypothesis 66 seems secure.

Following [16, 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.

Fix a universal Turing machine  $U$ , let  $x, y$  be two strings and  $t$  be a time bound. Define  $\text{cK}^t(x \mid y)$  as the length of the smallest description  $d$ , such that for every  $1 \leq i \leq |x| + 1$  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  $\text{cK}^t$  consists of a random string  $x$  and a random string  $y$ , both of input length  $n$ . Hence in the hypothesis below, we actually state that no non-uniform algorithm of  $2^{(1-\epsilon)n}$  size can solve  $\text{cK}^t$  on input length  $2n$ .

► **Hypothesis 67** (Strong Peregbor Hypothesis for  $\text{cK}^t$ ; Non-uniform Version). *There are polynomials  $t(n) \geq 2n$  and  $p(n)$ , such that for every constant  $\epsilon > 0$ , the following holds. Every non-uniform algorithm of  $2^{(1-\epsilon)n}$  size that attempts to solve  $\text{cK}^t$  fails on a  $1/p(n)$  fraction of inputs.*

## 6 MCSP-Related Results

In this section, we generalize Theorem 1 to the case of MCSP. Throughout this section, we maintain the convention that our input is the truth table of a function  $f : \{0, 1\}^n \rightarrow \{0, 1\}$ , and  $N = 2^n$  is the input length. We use  $tt$  to denote an input truth table. Recall that  $\text{Size}(tt)$  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.

► **Theorem 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  $\text{NC}^0$ , then MCSP is (exponentially) hard on average.*

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  $\text{DLOGTIME}$ . 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  $\text{KT}(x) \in [N - O(\log N), N + O(\log N)]$ . 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 [62] is  $2^{O(\eta)}$ , which is polynomial in  $N$ .

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

► **Theorem 68** ([28]). *There is a constant  $c$  such that the following is true. Let  $C(n)$  be the maximum circuit complexity of any function  $f : \{0, 1\}^n \rightarrow \{0, 1\}$ , then  $C_{\text{lb}}(n) \leq C(n) \leq C_{\text{ub}}(n)$ , where*

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

Therefore, given a random truth table  $tt$  of length  $N = 2^n$ , we could use any value between  $C_{\text{lb}}(n)$  and  $C_{\text{ub}}(n)$  as an estimate of  $\text{Size}(tt)$ . However, we could only prove that our additive error is  $\eta := (C_{\text{ub}}(n) - C_{\text{lb}}(n)) \cdot O(n) = O(2^n \log n/n)$ .<sup>17</sup> The overhead in [62] would be

<sup>17</sup>The extra  $O(n)$  factor is because we measure  $\eta$  by *bit*-complexity instead of *gate*-complexity, and every gate in the (maximum) circuit needs  $O(n)$  bits to describe.

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

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

► **Remark 69.** Ilango encountered a similar issue in his search-to-decision reduction for MFSP (Minimum Formula Size Problem) [46]. 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 [46] and our work, the  $2^{O(\eta)}$  factor comes from different reasons. Ilango’s algorithm runs in time  $\text{poly}(t)$  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 paper, we need to sample a *uniformly random circuit* (w.r.t. some encoding), and let  $p$  be the probability that the truth table of a sampled circuit is equal to a given one; 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  $f_{\text{MCSP}}$  based on the assumption that MCSP is *exponentially* hard on average. However, we are only able to prove that  $f_{\text{MCSP}}$  is hard to invert on an inverse-sub-exponential fraction ( $2^{-o(N)}$ ) of inputs. We will need the following variant of Theorem 23, 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”.

► **Theorem 70.** *Let  $p(n) = 2^{o(n)}$ ,  $f$  be a length-preserving function that is exponentially hard to invert on a  $1/p(n)$  fraction of inputs. In other words, there is a constant  $\epsilon > 0$  such that for every integer  $n$  and every randomized algorithm  $\mathcal{A}$  that runs in  $2^{\epsilon n}$  time,*

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

*Then there exists a one-way function.*

**Proof Sketch.** We verify that the standard proof for Theorem 23 also works in our setting. We use notations in [30, 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 [30, 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  $a(n) := 2n^2 p(n) q(m(n))$  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 scheme naturally occurs in the lower bound proofs for the maximum circuit complexity, which usually use a counting argument [77, 28]: If every circuit of size  $\text{LB}(n)$  can be encoded as a string of length  $2^n - 1$ , then there must exist an  $n$ -bit Boolean function without size- $\text{LB}(n)$  circuits.

In particular, in the lower bound proof of [28], the authors represented a circuit as a *stack program*. For a detailed description of stack programs, the reader is referred to [28]. We only need the following property of them:

► **Theorem 71.** *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

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

By Theorem 71, every Boolean function over  $n$  inputs has a stack program of bit-length  $C'_{\text{ub}}(n)$ .

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 72** ([83, 84]; see also [88, p. 304]). *Let  $f : \{0, 1\}^n \rightarrow \{0, 1\}$  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  $x_1, x_2, \dots, x_r \in \{0, 1\}^n$ ,  $C(x_1, x_2, \dots, x_r) = f(x_1) \circ f(x_2) \circ \dots \circ f(x_r)$ .*

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

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

*Then there exists a one-way function.*

**Proof.** By Theorem 70, 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  $2^{\epsilon \tilde{N}/10}$  time,

$$\Pr_{\mathbf{x} \leftarrow \mathcal{U}_{\tilde{N}}} [\mathcal{A}(f(\mathbf{x})) \in f^{-1}(f(\mathbf{x}))] \leq 1 - 1/p(\tilde{N}). \quad (3)$$

Let  $\tilde{N}$  be the input length of  $f$ ,  $n$  be the largest integer such that  $n + C'_{\text{ub}}(n) \leq \tilde{N}$ . (Recall that every Boolean function over  $n$  inputs can be represented by a circuit, or stack program, of bit-length  $C'_{\text{ub}}(n)$ .) The first  $n$  bits of the input denote an integer  $s \leq C_{\text{ub}}(n)$ ,

## 35:40 Hardness of KT Characterizes Parallel Cryptography

and the next  $C'_{\text{ub}}(n)$  bits denote a circuit  $C$  of size at most  $s$ . If the input is invalid (e.g., if  $s > C_{\text{ub}}(n)$  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  $C$ . In other words, our weak one-way function is defined as follows:

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

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

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

---

```

1: function  $\mathcal{A}_{\text{MCSP}}(tt)$ 
2:    $opt \leftarrow +\infty$ 
3:   for  $s \in [C'_{\text{ub}}(n)]$  do
4:      $(s', C) \leftarrow \mathcal{A}_{\text{owf}}(s, tt)$ 
5:     if  $tt(C) = tt$  then
6:        $opt \leftarrow \min\{opt, |C|\}$ 
7:   return  $opt$ 

```

---

Let  $\text{Err}$  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.  $\geq 1/2q(N)$ . By the hardness of MCSP and a Markov bound, we have

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

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

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

It follows that

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

Let  $p(\tilde{N}) := (2q(N) - 1)2q(N)2^{O(\frac{N \log \log N}{\log N})}$ . It is indeed the case that  $p(\tilde{N}) = 2^{o(\tilde{N})}$ , since  $N = 2^n \geq \Omega(\tilde{N})$ , and  $q(N) = 2^{o(N)}$ . We can see that every adversary  $\mathcal{A}_{\text{owf}}$  that runs in  $2^{\epsilon\tilde{N}/10}$  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 73. We show that if there is an exponentially hard weak one-way function in DLOGTIME, then MCSP is (also exponentially) hard on average.



► **Theorem 74.** *Suppose that there is a weak one-way function  $f$  computable in DLOGTIME with security  $2^{\Omega(N)}$ . (See Definition 49.) Then, no nonuniform algorithm of size  $2^{o(N)}$  can solve MCSP on a  $1 - 2^{-o(N)}$  fraction of inputs.*

**Proof.** Fix an input length  $M$ . Let  $\delta := 1/\text{poly}(M)$ , so there is a constant  $\kappa_1$  such that every adversary of size  $2^{\kappa_1 M}$  fails to invert  $f$  on a  $1 - \delta$  fraction of inputs. We construct a condEP-PRG  $G$  according to Construction 56. The stretch of  $G$  is  $\alpha := \kappa_2 M$  for some small enough constant  $\kappa_2 > 0$ . Its outputs are  $4\epsilon$ -indistinguishable from true random strings, where  $\epsilon := 1/M^{10}$ . We use the hash functions in Section 5.4 as the extractors and hardcore functions. Note that the list size of the hardcore function is  $L := 2^{O(\alpha)}$ . Still, for some positive constant  $\kappa_3 = \kappa_1 - O(\kappa_2) > 0$ , no adversary of size  $2^{\kappa_3 M}$  could  $4\epsilon$ -distinguish the outputs of  $G$  from random strings.

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  $(K + \kappa_2)M$  input bits, where  $K$  is some absolute constant. W.l.o.g. we may assume that  $KM$  is a power of 2 (by padding a random string to both the input and output of  $\tilde{G}$ ). Again, no adversary of size  $2^{\kappa_3 M}$  could  $4\epsilon$ -distinguish the outputs of  $\tilde{G}$  from random strings. It suffices to prove that the outputs of  $\tilde{G}$ , when viewed as truth tables (and padded to length  $2KM$ ), have non-trivial circuit complexity. (As a result, if MCSP can be solved by a size- $2^{o(M)}$  circuit on average, then  $\tilde{G}$  is not exponentially secure.)

Now, let  $N := KM$ ,  $n := \log N$ , and  $\kappa_4 := \frac{\kappa_2}{K}$ , then  $\tilde{G}$  is a condEP-PRG that maps  $N$  input bits to  $(1 + \kappa_4)N$  input bits. Let  $tt^{\text{in}} \in \{0, 1\}^N$  be an input,  $tt^{\text{out}} \in \{0, 1\}^{2N}$  be the string whose first  $(1 + \kappa_4)N$  bits are  $\tilde{G}(tt^{\text{in}})$ , and other bits are zero.

▷ **Claim 75.**  $\text{Size}(tt^{\text{out}}) \leq (1 + o(1))2^n/n$ .

**Proof.** 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  $i$ ,  $\tilde{G}(tt^{\text{in}})$  computes the indices  $q(i, 1), q(i, 2), \dots, q(i, r)$ , queries  $(tt^{\text{in}})_{q(i, j)}$  for each  $1 \leq j \leq r$ , and computes  $(tt^{\text{out}})_i$  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 72, there is a circuit  $C$  of size  $(1 + o(1))2^n/n$  that on input  $(x_1, x_2, \dots, x_r)$ , outputs the concatenation of  $(tt^{\text{in}})_{x_1}, (tt^{\text{in}})_{x_2}, \dots, (tt^{\text{in}})_{x_r}$ . We design a circuit for  $tt^{\text{out}}$  as follows.

- On input  $i$ , if  $i > (1 + \kappa_4)N$ , then output 0.
- Otherwise we simulate  $\tilde{G}(i)$  to obtain the indices  $q(i, j)$  for every  $1 \leq j \leq r$ . This step takes  $O(n)$  time, and thus can be implemented in size  $\text{poly}(n)$ .
- Use the circuit  $C$  of size  $(1 + o(1))2^n/n$  to obtain  $(tt^{\text{in}})_{q(i, j)}$  for every  $1 \leq j \leq r$ .
- Finally, we can simulate  $\tilde{G}(i)$  to obtain  $(tt^{\text{out}})_i$ . Again, this step can be implemented in size  $\text{poly}(n)$ .

It follows that the circuit complexity of  $tt^{\text{out}}$  is at most  $(1 + o(1))2^n/n$ . ◁

On the other hand, let  $r \in \{0, 1\}^{(1+\kappa_4)N}$  be a truly random string. We also append zeros in the end of  $r$  to make it a truth table of length  $2N$ . Denote  $\text{Size}(r)$  the circuit complexity of this length- $2N$  truth table. Let  $\kappa_5 := \kappa_4/10$ , and  $s := (1 + \kappa_5)2^n/n$ . By Theorem 71, the number of strings  $r$  such that  $\text{Size}(r) \leq s$  is at most

$$2^{(s+1)(O(1)+\log(n+s))} \leq 2^{(1+2\kappa_5)N} \lll 2^{(1+\kappa_4)N}.$$

It follows that with overwhelming probability, for a random string  $r \in \{0, 1\}^{(1+\kappa_4)N}$ ,  $\text{Size}(r) \geq (1 + \kappa_5)2^n/n$ . If we can solve MCSP by a nonuniform algorithm of size  $2^{o(N)}$ , then  $\hat{G}$  would not be a secure condEP-PRG. ◀

► **Remark 76.** Theorem 73 and 74 are not exactly converses of each other, as there are two gaps. First, there is a loss of  $2^{O(\frac{N \log \log N}{\log N})}$ . Second, Theorem 73 only produces a (polynomial-time computable) one-way function, but Theorem 74 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  $\oplus\text{L}$  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.<sup>18</sup> For example, if the circuit that corresponds to a uniformly random description has depth at most  $O(\log n)$  with high probability, then a  $\oplus\text{L}$  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 77 (Results for MFSP).** It is possible to extend Theorem 73 to the case of MFSP (Minimum Formula Size Problem). In particular, suppose that 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 [19], hence this one-way function is in ALOGTIME, and we obtain DLOGTIME-computable one-way functions from Theorem 32. We omit the proof here, as it is essentially the same as Theorem 73 except that it uses the best bounds for maximum formula complexity (see [57] and references therein).

However, we are not aware of any “mass production theorem” (Theorem 72) for formulas. Therefore we are not able to prove an MFSP-version of Theorem 74.

## 7 The Average-Case Complexity of MKtP

### 7.1 Characterizing One-Way Functions Using MKtP

We recall the main result of [62] showing an equivalence between the average-case hardness of  $K^p$  for some polynomial  $p$  and the existence of one-way functions.

► **Theorem 78** ([62]). *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  $\lambda \log n$ .*

Somewhat counter-intuitively, we show that 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 very 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.

<sup>18</sup> If this heuristic is always true (i.e. it is a worst-case algorithm), then  $\oplus\text{L} = \text{P}$ .

► **Theorem 79.** *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 33.

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  $d + \log t = \text{Kt}(x)$  have the property that  $t \leq O(n^{2c+1})$ . Actually, for all but a  $1/n^{2c}$  fraction of inputs,  $\text{K}(x) \geq n - 2c \log n - 1$ , while for all inputs  $x$  we have  $\text{Kt}(x) \leq n + \log n + O(1)$ . Hence for all strings  $x$  with  $\text{K}(x) \geq n - 2c \log n - 1$ ,  $x$  can be generated from its optimal description in time  $O(n^{2c+1})$ .

We define a weak one-way function  $f$  as follows. It takes as input a triple  $(\ell, k, M)$ , where  $\ell \in [n + \log n]$ ,  $k \in [(2c + 1) \log n]$ , and  $M \in \{0, 1\}^{n + \log n}$ , and outputs  $(\ell, k, \text{out})$ . Here  $\text{out}$  is the result of running  $U^{M'}$  for at most  $2^k$  steps, where  $M'$  is the  $\ell$ -bit prefix of  $M$ . Just as in Claim 35, 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  $\text{Kt}(x)$  can cycle over all possible  $\ell$  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 25 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 39 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 78 and Theorem 79 yield the following corollary.

► **Corollary 80.** *Kt is bounded-error hard on average iff there is a polynomial  $p$  such that  $\text{K}^p$  is bounded-error hard on average.*

Corollary 80 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 79 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 to be 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 81.** Given functions  $t : \mathbb{N} \rightarrow \mathbb{N}$ ,  $\ell : \mathbb{N} \rightarrow \mathbb{N}$  (satisfying  $\ell(n) \leq n$  for each  $n$ ) and  $s : \mathbb{N} \rightarrow \mathbb{N}$ , we say that a family of functions  $\{G_n\}$ , where  $G_n : \{0, 1\}^{\ell(n)} \rightarrow \{0, 1\}^n$  is a time  $t$  pseudo-random generator (PRG) with seed length  $\ell$  against size  $s$  if  $G(z)$  is computable in time  $t(|z|)$  and for each  $n$ ,  $G_n(\mathcal{U}_{\ell(n)})$  is  $1/s(n)$ -indistinguishable from  $\mathcal{U}_n$  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 [69, 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 82** ([69, 13]). If  $\text{DTIME}(2^n \text{poly}(n)) \not\subseteq \text{P}_{/\text{poly}}$ , then for each  $\ell$  such that  $\ell(n) = n^{\Omega(1)}$ , there is a seed-extending time  $2^\ell \text{poly}(\ell)$  PRG with seed length  $\ell$  against polynomial size.

We use Theorem 82 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 83.** The following are equivalent:

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

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

(1)  $\iff$  (2) is shown in [69, 13].

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

(3)  $\implies$  (4): We use Theorem 82. 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 82, the hardness of the decision version implies that for each  $\epsilon > 0$ , there is a seed-extending time  $2^m \text{poly}(m)$  PRG with seed length  $n^\epsilon$  against polynomial size.

(4)  $\implies$  (5): Consider a seed-extending  $2^m \text{poly}(m)$  time PRG  $G = \{G_n\}$  with seed length  $\gamma n$ , where  $1/2 > \gamma > 1 - \delta$ . Such a PRG is implied by a PRG with smaller seed length, simply by truncating the output. Since the seed length is  $\gamma n$  and the PRG is computable in time  $2^{\gamma n} \text{poly}(n)$ , we have that each output of the 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. By our choice of  $\delta$ , this means that they are correct on at least a  $2/3$  fraction of strings  $G_n(z)$  for seed  $z$  of length  $\gamma n$ . Now we can define a distinguisher  $D$  as follows:  $D$  computes  $C_n(x)$  and accepts iff  $C_n(x) \leq n - \log(n)$ .  $D$  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  $D$  is a distinguisher of polynomial size, contradicting the assumption that  $G$  is a 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? We believe it is within reach to refute the non-uniform version of Hypothesis 66, by e.g. implementing the inverter in [27] as circuits.

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 algorithm imply new circuit lower bounds, as in the case of SAT algorithms [90] and learning algorithms [72]? 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 [89]). 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  $2^{-N}/\text{poly}(N)$ ? 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  $2^{-N}/2^{O(N \frac{\log \log N}{\log N})}$ .

Second, is there a  $\oplus L$  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 hardness of MCSP, establishing a tighter converse of Theorem 74.

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 72.

**Other cryptographic primitives?** Meta-complexity can characterize the existence of one-way functions [62] and one-way functions in  $NC^0$  (this paper). Is there a similar characterization for other cryptographic primitives, such as public-key encryption [75, 25], or indistinguishability obfuscation [14]?

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 [31].

---

### References

- 1 Miklós Ajtai. Generating hard instances of lattice problems (extended abstract). In *Proc. 28th Annual ACM Symposium on Theory of Computing (STOC)*, pages 99–108, 1996. doi:10.1145/237814.237838.
- 2 Adi Akavia, Oded Goldreich, Shafi Goldwasser, and Dana Moshkovitz. On basing one-way functions on NP-hardness. In *Proc. 38th Annual ACM Symposium on Theory of Computing (STOC)*, pages 701–710, 2006. doi:10.1145/1132516.1132614.

- 3 Eric Allender. When worlds collide: Derandomization, lower bounds, and Kolmogorov complexity. In *Proc. 21st Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*, volume 2245 of *Lecture Notes in Computer Science*, pages 1–15, 2001. doi:10.1007/3-540-45294-X\_1.
- 4 Eric Allender, Harry Buhrman, Michal Koucký, Dieter van Melkebeek, and Detlef Ronneburger. Power from random strings. *SIAM Journal of Computing*, 35(6):1467–1493, 2006. doi:10.1137/050628994.
- 5 Eric Allender, Mahdi Cheraghchi, Dimitrios Myrisiotis, Harsha Tirumala, and Ilya Volkovich. One-way functions and a conditional variant of MKTP. *Electronic Colloquium on Computational Complexity (ECCC)*, 2021. URL: <https://eccc.weizmann.ac.il/report/2021/009/>.
- 6 Eric Allender and Shuichi Hirahara. New insights on the (non-)hardness of circuit minimization and related problems. *ACM Transactions on Computation Theory*, 11(4):27:1–27:27, 2019. doi:10.1145/3349616.
- 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 Transactions on Information Theory*, 38(2):509–516, 1992. doi:10.1109/18.119713.
- 8 Benny Applebaum. *Cryptography in Constant Parallel Time*. Information Security and Cryptography. Springer, 2014. doi:10.1007/978-3-642-17367-7.
- 9 Benny Applebaum. Cryptographic hardness of random local functions - survey. *Computational Complexity*, 25(3):667–722, 2016.
- 10 Benny Applebaum. Exponentially-hard Gap-CSP and local PRG via local hardcore functions. In *Proc. 58th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 836–847, 2017. doi:10.1109/FOCS.2017.82.
- 11 Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. Cryptography in  $NC^0$ . *SIAM Journal of Computing*, 36(4):845–888, 2006. doi:10.1137/S0097539705446950.
- 12 Sanjeev Arora and Boaz Barak. *Computational Complexity: A Modern Approach*. Cambridge University Press, 2009.
- 13 László Babai, Lance Fortnow, Noam Nisan, and Avi Wigderson. BPP has subexponential time simulations unless EXPTIME has publishable proofs. *Computational Complexity*, 3:307–318, 1993. doi:10.1007/BF01275486.
- 14 Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. *Journal of the ACM*, 59(2):6:1–6:48, 2012. doi:10.1145/2160158.2160159.
- 15 Joshua Baron, Yuval Ishai, and Rafail Ostrovsky. On linear-size pseudorandom generators and hardcore functions. *Theoretical Computer Science*, 554:50–63, 2014. doi:10.1016/j.tcs.2014.06.013.
- 16 Eli Biham, Yaron J. Goren, and Yuval Ishai. Basing weak public-key cryptography on strong one-way functions. In *Proc. 5th Theory of Cryptography Conference (TCC)*, volume 4948 of *Lecture Notes in Computer Science*, pages 55–72, 2008. doi:10.1007/978-3-540-78524-8\_4.
- 17 Andrej Bogdanov and Luca Trevisan. On worst-case to average-case reductions for NP problems. *SIAM Journal of Computing*, 36(4):1119–1159, 2006. doi:10.1137/S0097539705446974.
- 18 J. L. Bordewijk. Inter-reciprocity applied to electrical networks. *Applied Scientific Research, Section A*, pages 1–74, 1957. doi:10.1007/BF02410413.
- 19 Samuel R. Buss. The Boolean formula value problem is in ALOGTIME. In *Proc. 19th Annual ACM Symposium on Theory of Computing (STOC)*, pages 123–131, 1987. doi:10.1145/28395.28409.
- 20 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*, pages 75–85. Springer, 2009. doi:10.1007/978-3-642-11269-0\_6.

- 21 Ran Canetti, Yevgeniy Dodis, Shai Halevi, Eyal Kushilevitz, and Amit Sahai. Exposure-resilient functions and all-or-nothing transforms. In *Advances in Cryptology - EUROCRYPT 2000, International Conference on the Theory and Application of Cryptographic Techniques*, volume 1807 of *Lecture Notes in Computer Science*, pages 453–469, 2000. doi:10.1007/3-540-45539-6\_33.
- 22 Lijie Chen, Ce Jin, and R. Ryan Williams. Hardness magnification for all sparse NP languages. In *Proc. 60th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 1240–1255, 2019. doi:10.1109/FOCS.2019.00077.
- 23 Lijie Chen and Hanlin Ren. Strong average-case lower bounds from non-trivial derandomization. In *Proc. 52nd Annual ACM Symposium on Theory of Computing (STOC)*, pages 1327–1334, 2020. doi:10.1145/3357713.3384279.
- 24 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 Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 396–407, 1985. doi:10.1109/SFCS.1985.55.
- 25 Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976. doi:10.1109/TIT.1976.1055638.
- 26 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.
- 27 Amos Fiat and Moni Naor. Rigorous time/space trade-offs for inverting functions. *SIAM Journal of Computing*, 29(3):790–803, 1999. doi:10.1137/S0097539795280512.
- 28 Gudmund Skovbjerg Frandsen and Peter Bro Miltersen. Reviewing bounds on the circuit size of the hardest functions. *Information Processing Letters*, 95(2):354–357, 2005. doi:10.1016/j.ipl.2005.03.009.
- 29 Ofer Gabber and Zvi Galil. Explicit constructions of linear-sized superconcentrators. *Journal of Computer and System Sciences*, 22(3):407–420, 1981. doi:10.1016/0022-0000(81)90040-4.
- 30 Oded Goldreich. *The Foundations of Cryptography - Volume 1: Basic Techniques*. Cambridge University Press, 2001. doi:10.1017/CB09780511546891.
- 31 Oded Goldreich, Russell Impagliazzo, Leonid A. Levin, Ramarathnam Venkatesan, and David Zuckerman. Security preserving amplification of hardness. In *Proc. 31st Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 318–326, 1990. doi:10.1109/FSCS.1990.89550.
- 32 Oded Goldreich and Leonid A. Levin. A hard-core predicate for all one-way functions. In *Proc. 21st Annual ACM Symposium on Theory of Computing (STOC)*, pages 25–32, 1989. doi:10.1145/73007.73010.
- 33 Shafi Goldwasser, Dan Gutfreund, Alexander Healy, Tali Kaufman, and Guy N. Rothblum. Verifying and decoding in constant depth. In *Proc. 39th Annual ACM Symposium on Theory of Computing (STOC)*, pages 440–449, 2007. doi:10.1145/1250790.1250855.
- 34 Alexander Golovnev, Rahul Ilango, Russell Impagliazzo, Valentine Kabanets, Antonina Kolokolova, and Avishay Tal.  $AC^0[p]$  lower bounds against MCSP via the coin problem. In *Proc. 46th International Colloquium on Automata, Languages and Programming (ICALP)*, volume 132 of *LIPICs*, pages 66:1–66:15, 2019. doi:10.4230/LIPICs.ICALP.2019.66.
- 35 Venkatesan Guruswami and Piotr Indyk. Expander-based constructions of efficiently decodable codes. In *Proc. 42nd Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 658–667, 2001. doi:10.1109/SFCS.2001.959942.
- 36 Venkatesan Guruswami, Christopher Umans, and Salil P. Vadhan. Unbalanced expanders and randomness extractors from Parvaresh-Vardy codes. *Journal of the ACM*, 56(4):20:1–20:34, 2009. doi:10.1145/1538902.1538904.
- 37 Iftach Haitner, Omer Reingold, and Salil P. Vadhan. Efficiency improvements in constructing pseudorandom generators from one-way functions. *SIAM Journal of Computing*, 42(3):1405–1430, 2013. doi:10.1137/100814421.

- 38 Johan Håstad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. A pseudorandom generator from any one-way function. *SIAM Journal of Computing*, 28(4):1364–1396, 1999. doi:10.1137/S0097539793244708.
- 39 Martin E. Hellman. A cryptanalytic time-memory trade-off. *IEEE Transactions on Information Theory*, 26(4):401–406, 1980. doi:10.1109/TIT.1980.1056220.
- 40 Shuichi Hirahara. Non-black-box worst-case to average-case reductions within NP. In *Proc. 59th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 247–258, 2018. doi:10.1109/FOCS.2018.00032.
- 41 Shuichi Hirahara. Characterizing average-case complexity of PH by worst-case meta-complexity. In *Proc. 61st Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 50–60, 2020. doi:10.1109/FOCS46700.2020.00014.
- 42 Shuichi Hirahara. Unexpected hardness results for Kolmogorov complexity under uniform reductions. In *Proc. 52nd Annual ACM Symposium on Theory of Computing (STOC)*, pages 1038–1051, 2020. doi:10.1145/3357713.3384251.
- 43 Shuichi Hirahara and Rahul Santhanam. On the average-case complexity of MCSP and its variants. In *Proc. 32nd Computational Complexity Conference (CCC)*, volume 79 of *LIPICs*, pages 7:1–7:20, 2017. doi:10.4230/LIPICs.CCC.2017.7.
- 44 Thomas Holenstein, Ueli M. Maurer, and Johan Sjödin. Complete classification of bilinear hard-core functions. In *Proc. 24th Annual International Cryptology Conference (CRYPTO)*, volume 3152 of *Lecture Notes in Computer Science*, pages 73–91. Springer, 2004. doi:10.1007/978-3-540-28628-8\_5.
- 45 Shlomo Hoory, Nathan Linial, and Avi Wigderson. Expander graphs and their applications. *Bulletin of the American Mathematical Society*, pages 439–561, 2006. doi:10.1090/S0273-0979-06-01126-8.
- 46 Rahul Ilango. Connecting peregbor conjectures: Towards a search to decision reduction for minimizing formulas. In *Proc. 35th Computational Complexity Conference (CCC)*, volume 169 of *LIPICs*, pages 31:1–31:35, 2020. doi:10.4230/LIPICs.CCC.2020.31.
- 47 Rahul Ilango. Constant depth formula and partial function versions of MCSP are hard. In *Proc. 61st Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 424–433, 2020. doi:10.1109/FOCS46700.2020.00047.
- 48 Rahul Ilango, Bruno Loff, and Igor Carboni Oliveira. NP-hardness of circuit minimization for multi-output functions. In *Proc. 35th Computational Complexity Conference (CCC)*, volume 169 of *LIPICs*, pages 22:1–22:36, 2020. doi:10.4230/LIPICs.CCC.2020.22.
- 49 Russell Impagliazzo. Hard-core distributions for somewhat hard problems. In *Proc. 36th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 538–545, 1995. doi:10.1109/SFCS.1995.492584.
- 50 Russell Impagliazzo. A personal view of average-case complexity. In *Proc. 10th Annual Structure in Complexity Theory Conference*, pages 134–147, 1995. doi:10.1109/SCT.1995.514853.
- 51 Russell Impagliazzo and Leonid A. Levin. No better ways to generate hard NP instances than picking uniformly at random. In *Proc. 31st Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 812–821, 1990. doi:10.1109/SFCS.1990.89604.
- 52 Russell Impagliazzo and Ramamohan Paturi. On the complexity of  $k$ -SAT. *Journal of Computer and System Sciences*, 62(2):367–375, 2001. doi:10.1006/jcss.2000.1727.
- 53 Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity? *Journal of Computer and System Sciences*, 63(4):512–530, 2001. doi:10.1006/jcss.2001.1774.
- 54 Yuval Ishai and Eyal Kushilevitz. Randomizing polynomials: A new representation with applications to round-efficient secure computation. In *Proc. 41st Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 294–304, 2000. doi:10.1109/SFCS.2000.892118.



- 55 Yuval Ishai and Eyal Kushilevitz. Perfect constant-round secure computation via perfect randomizing polynomials. In *Proc. 29th International Colloquium on Automata, Languages and Programming (ICALP)*, pages 244–256, 2002. doi:10.1007/3-540-45465-9\_22.
- 56 Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Cryptography with constant computational overhead. In *Proc. 40th Annual ACM Symposium on Theory of Computing (STOC)*, pages 433–442, 2008. doi:10.1145/1374376.1374438.
- 57 Stasys Jukna. *Boolean Function Complexity - Advances and Frontiers*, volume 27 of *Algorithms and combinatorics*. Springer, 2012. doi:10.1007/978-3-642-24508-4.
- 58 Valentine Kabanets and Jin-Yi Cai. Circuit minimization problem. In *Proc. 32nd Annual ACM Symposium on Theory of Computing (STOC)*, pages 73–79, 2000. doi:10.1145/335305.335314.
- 59 Ker-I Ko. On the complexity of learning minimum time-bounded Turing machines. *SIAM Journal of Computing*, 20(5):962–986, 1991. doi:10.1137/0220059.
- 60 Leonid A. Levin. Randomness conservation inequalities; information and independence in mathematical theories. *Information and Control*, 61(1):15–37, 1984. doi:10.1016/S0019-9958(84)80060-1.
- 61 Leonid A. Levin. The tale of one-way functions. *Problems of Information Transmission*, 39(1):92–103, 2003.
- 62 Yanyi Liu and Rafael Pass. On one-way functions and Kolmogorov complexity. In *Proc. 61st Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 1243–1254, 2020. doi:10.1109/FOCS46700.2020.00118.
- 63 Yanyi Liu and Rafael Pass. On the possibility of basing cryptography on  $\text{EXP} \neq \text{BPP}$ . *Electronic Colloquium on Computational Complexity (ECCC)*, 28:56, 2021. URL: <https://eccc.weizmann.ac.il/report/2021/056>.
- 64 G. A. Margulis. Explicit constructions of concentrators. *Probl. Peredachi Inf.*, pages 71–80, 1973.
- 65 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 Annual ACM Symposium on Theory of Computing (STOC)*, pages 1215–1225, 2019. doi:10.1145/3313276.3316396.
- 66 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.
- 67 Mikito Nanashima. On basing auxiliary-input cryptography on NP-hardness via nonadaptive black-box reductions. In *Proc. 12th Conference on Innovations in Theoretical Computer Science (ITCS)*, volume 185 of *LIPICs*, pages 29:1–29:15, 2021. doi:10.4230/LIPICs.ITCS.2021.29.
- 68 Noam Nisan. Extracting randomness: How and why. A survey. In *Proc. 11th Annual IEEE Conference on Computational Complexity (CCC)*, pages 44–58, 1996. doi:10.1109/CCC.1996.507667.
- 69 Noam Nisan and Avi Wigderson. Hardness vs randomness. *Journal of Computer and System Sciences*, 49(2):149–167, 1994. doi:10.1016/S0022-0000(05)80043-1.
- 70 Noam Nisan and David Zuckerman. More deterministic simulation in logspace. In *Proc. 25th Annual ACM Symposium on Theory of Computing (STOC)*, pages 235–244, 1993. doi:10.1145/167088.167162.
- 71 Igor Carboni Oliveira, Ján Pich, and Rahul Santhanam. Hardness magnification near state-of-the-art lower bounds. In *Proc. 34th Computational Complexity Conference (CCC)*, volume 137 of *LIPICs*, pages 27:1–27:29, 2019. doi:10.4230/LIPICs.CCC.2019.27.
- 72 Igor Carboni Oliveira and Rahul Santhanam. Conspiracies between learning algorithms, circuit lower bounds, and pseudorandomness. In *Proc. 32nd Computational Complexity Conference (CCC)*, volume 79 of *LIPICs*, pages 18:1–18:49, 2017. doi:10.4230/LIPICs.CCC.2017.18.
- 73 Igor Carboni Oliveira and Rahul Santhanam. Hardness magnification for natural problems. In *Proc. 59th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 65–76, 2018. doi:10.1109/FOCS.2018.00016.

- 74 Alexander A. Razborov and Steven Rudich. Natural proofs. *Journal of Computer and System Sciences*, 55(1):24–35, 1997. doi:10.1006/jcss.1997.1494.
- 75 Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978. doi:10.1145/359340.359342.
- 76 Rahul Santhanam. Pseudorandomness and the minimum circuit size problem. In *Proc. 11th Conference on Innovations in Theoretical Computer Science (ITCS)*, volume 151 of *LIPICs*, pages 68:1–68:26, 2020. doi:10.4230/LIPICs.ITCS.2020.68.
- 77 Claude E. Shannon. The synthesis of two-terminal switching circuits. *Bell System technical journal*, 28(1):59–98, 1949. doi:10.1002/j.1538-7305.1949.tb03624.x.
- 78 Daniel A. Spielman. Linear-time encodable and decodable error-correcting codes. *IEEE Transactions on Information Theory*, 42(6):1723–1731, 1996. doi:10.1109/18.556668.
- 79 Amnon Ta-Shma and David Zuckerman. Extractor codes. *IEEE Transactions on Information Theory*, 50(12):3015–3025, 2004. doi:10.1109/TIT.2004.838377.
- 80 Roei Tell. Quantified derandomization of linear threshold circuits. In *Proc. 50th Annual ACM Symposium on Theory of Computing (STOC)*, pages 855–865, 2018. doi:10.1145/3188745.3188822.
- 81 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.
- 82 Luca Trevisan. Extractors and pseudorandom generators. *Journal of the ACM*, 48(4):860–879, 2001. doi:10.1145/502090.502099.
- 83 D. Uhlig. On the synthesis of self-correcting schemes from functional elements with a small number of reliable elements. *Mathematical notes of the Academy of Sciences of the USSR*, 15:558–562, 1974. doi:10.1007/BF01152835.
- 84 D. Uhlig. Zur parallelberechnung boolescher funktionen. *TR Ing.hochsch. Mittweida*, 1984.
- 85 Salil P. Vadhan. Pseudorandomness. *Foundations and Trends in Theoretical Computer Science*, 7(1-3):1–336, 2012. doi:10.1561/0400000010.
- 86 Salil P. Vadhan and Colin Jia Zheng. A uniform min-max theorem with applications in cryptography. In *Proc. 33rd Annual International Cryptology Conference (CRYPTO)*, volume 8042 of *Lecture Notes in Computer Science*, pages 93–110, 2013. doi:10.1007/978-3-642-40041-4\_6.
- 87 Hoeteck Wee. Finding Pessiland. In *Proc. 3rd Theory of Cryptography Conference (TCC)*, volume 3876 of *Lecture Notes in Computer Science*, pages 429–442, 2006. doi:10.1007/11681878\_22.
- 88 Ingo Wegener. *The complexity of Boolean functions*. Wiley-Teubner, 1987. URL: <http://ls2-www.cs.uni-dortmund.de/monographs/bluebook/>.
- 89 Ryan Williams. A new algorithm for optimal 2-constraint satisfaction and its implications. *Theoretical Computer Science*, 348(2-3):357–365, 2005. doi:10.1016/j.tcs.2005.09.023.
- 90 Ryan Williams. Improving exhaustive search implies superpolynomial lower bounds. *SIAM Journal of Computing*, 42(3):1218–1244, 2013. doi:10.1137/10080703X.
- 91 Virginia Vassilevska Williams. On some fine-grained questions in algorithms and complexity. In *Proc. of the ICM*, volume 3, pages 3431–3472, 2018.
- 92 Andrew Chi-Chih Yao. Theory and applications of trapdoor functions (extended abstract). In *Proc. 23rd Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 80–91, 1982. doi:10.1109/SFCS.1982.45.
- 93 Yu Yu, Xiangxue Li, and Jian Weng. Pseudorandom generators from regular one-way functions: New constructions with improved parameters. *Theoretical Computer Science*, 569:58–69, 2015. doi:10.1016/j.tcs.2014.12.013.

## A Proof of Theorem 57

► **Theorem 57.** *Let  $\epsilon, \delta, \alpha, f$  be defined as in Construction 56. If  $\epsilon \geq 1/\text{poly}(n)$  and  $L \leq \text{poly}(n)$ , then there is a function  $r : \mathbb{N} \rightarrow \mathbb{N}$  such that  $G = \{G_{n,r(n)}\}_{n \in \mathbb{N}}$  is a condEP-PRG with stretch  $\alpha$  and security  $4\epsilon$ .*

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

For convenience, we only consider non-uniform adversaries in this section. (See also Remark 52.) 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.

### A.1 Impagliazzo's Hardcore Lemma

► **Lemma 84** ([49]; see e.g., [12]). *Let  $f$  be a candidate (weak) one-way function,  $\epsilon, \delta > 0$ . Suppose for every  $\mathcal{E} \subseteq \{0, 1\}^n$  with  $|\mathcal{E}| \geq \frac{\delta}{2} \cdot 2^n$ , there is a circuit  $C$  of size  $s(n)$  such that*

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

*Then there is a circuit of size  $O(s(n) \cdot n\epsilon^{-2})$  that inverts  $f$  on a  $1 - \delta$  fraction of inputs.*

### A.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  $r : \mathbb{N} \rightarrow \mathbb{N}$ , 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 84, we have:

▷ **Claim 85.** There is a sequence of subsets  $\mathcal{X}' = \{\mathcal{X}'_n \subseteq \{0, 1\}^n\}$  with  $|\mathcal{X}'_n| \geq 2^n/\text{poly}(n)$ , such that for every polynomial-size adversary  $\mathcal{A}$ ,

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

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  $\epsilon$ . Then there is an adversary of size  $O(s \cdot n\epsilon^{-2})$  that inverts  $f$  on a  $1 - \delta$  fraction of inputs. ◀

## 35:52 Hardness of KT Characterizes Parallel Cryptography

Now, for every string  $y \in \{0, 1\}^n$ , let  $|f_{\mathcal{X}'_n}^{-1}(y)|$  denote the number of inputs  $x \in \mathcal{X}'_n$  such that  $f(x) = y$ . Let  $r \in [1, n]$ ,  $W_r$  be the number of strings  $x \in \mathcal{X}'_n$  such that  $|f_{\mathcal{X}'_n}^{-1}(f(x))| \in [2^{r-1}, 2^r]$ . 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 denote  $r(n)$  to be this integer  $r$ , and

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

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  $\mathcal{X}_n$  on an  $\epsilon$  fraction of inputs also inverts  $\mathcal{X}'_n$  on an  $\epsilon/n$  fraction of inputs. To summarize:

▷ **Claim 86.** There is a function  $r(n) \leq n$  and a sequence of subsets  $\mathcal{X} = \{\mathcal{X}_n\}$  with  $|\mathcal{X}_n| \geq 2^n/\text{poly}(n)$ , such that  $f$  is  $r$ -regular on  $\mathcal{X}$ , and for every polynomial-size adversary  $\mathcal{A}$ ,

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

More precisely, suppose that for every function  $r : \mathbb{N} \rightarrow \mathbb{N}$  and sequence of subsets  $\mathcal{X} = \{\mathcal{X}_n\}$  such that  $|\mathcal{X}_n| \geq \frac{\delta}{2^n} \cdot 2^n$  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  $\epsilon$ . Then there is an adversary of size  $s \cdot \text{poly}(n/\epsilon)$  that inverts  $f$  on a  $1 - \delta$  fraction of inputs. ◀

### A.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  $d = d_{\text{Ext}}(n, \epsilon)$ . We need the following two extractors:

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

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

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

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

► **Lemma 87.** For every integer  $n$ , the function  $\tilde{f}_n$  satisfies the following properties:

1. (Uniformity) For every integer  $n$ ,  $\text{SD}(\tilde{f}_n(\mathcal{X}_n, \mathcal{U}_{2d}), \mathcal{U}_{\ell(n)}) \leq 2\epsilon$ .
2. (Hiding) For every polynomial-size adversary  $\mathcal{A}$  and every integer  $n$ ,

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

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  $\gamma$ , then there is an adversary  $\mathcal{A}'$  of size  $O(s)$  that inverts  $f_n(\mathcal{X}_n)$  w.p. at least  $O(\gamma/\epsilon^2)$ .

**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  $p(x_0 | y_0) := \Pr_{\mathbf{x} \leftarrow \mathcal{X}_n} [\mathbf{x} = x_0 | f_n(\mathbf{x}) = y_0]$ .

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

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

$$\text{SD}(\mathbf{z}_1 \circ \text{Ext}_1(\mathbf{x}, \mathbf{z}_1), \mathcal{U}_{d+m_1} \mid f(\mathbf{x})) \leq \epsilon.$$

Now, for every  $y_0 \in f_n(\mathcal{X}_n)$ , since  $|f_{\mathcal{X}}^{-1}(y_0)| \leq 2^r$ , 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  $\log(|\mathcal{X}_n|/2^r) \geq n - r + \log(\delta/2n) \geq k_2$ . Since  $\text{Ext}_2$  is a strong  $(k_2, \epsilon)$ -extractor, we have

$$\text{SD}(\mathbf{z}_2 \circ \text{Ext}_2(f(\mathbf{x}), \mathbf{z}_2), \mathcal{U}_{d+m_2}) \leq \epsilon.$$

It follows that

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

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

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

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

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  $\text{Ext}_1(\mathbf{x}, \mathbf{z}_1) = \mathbf{z}$ . Then we output  $\mathcal{A}'(\mathbf{y}) := \mathcal{A}(\mathbf{z}_1 \circ \mathbf{z} \circ \mathbf{z}_2 \circ \text{Ext}_2(\mathbf{y}, \mathbf{z}_2))$ .

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

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

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

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

#### A.4 Step III: Appending a Hardcore Function

Note that the output length of  $\tilde{f}_n$  is  $\tau := \log(n/\delta) + 4 \log \frac{1}{\epsilon} + O(1)$  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  $\text{HC} : \{0, 1\}^n \times \{0, 1\}^{d'} \rightarrow \{0, 1\}^k$  with distinguishing probability  $\epsilon$ , where  $k := \tau + \alpha$ , and  $d' := d_{\text{HC}}(n, k, \epsilon)$ . Let  $R$  be the reconstruction algorithm of this hardcore function, and  $L := L(n, k, \epsilon)$  be the list size.

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

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

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 88 shows that  $G_{n,r}(\mathcal{E}_{\tilde{n}})$  is pseudorandom, and Lemma 89 shows that  $G_{n,r}(\mathcal{E}_{\tilde{n}})$  is entropy-preserving.

► **Lemma 88.** *Every polynomial-size adversary  $\mathcal{A}$  fails to  $4\epsilon$ -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  $4\epsilon$ -distinguishes  $G_{n,r}(\mathcal{E}_{\tilde{n}})$  from  $\mathcal{U}_{\ell(n)+d'+k}$ , then there is an adversary  $\mathcal{A}'$  of size  $s \cdot \text{poly}(n/\epsilon)$  that on input  $\tilde{f}_n(\mathcal{X}_n, \mathcal{U}_{2d})$ , guesses  $\mathcal{X}_n$  with success probability  $\epsilon/2L$ .*

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

Since  $\text{SD}(\tilde{f}_n(\mathcal{X}_n, \mathcal{U}_{2d}), \mathcal{U}_{\ell(n)}) \leq 2\epsilon$ , it must be the case that  $\mathcal{A}$  could  $2\epsilon$ -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  $2\epsilon$ -distinguish  $\mathcal{U}_{d'} \circ \text{HC}(\mathbf{x}, \mathcal{U}_{d'})$  from  $\mathcal{U}_{d'+k}$ . We say a string  $w := (x, z_1, z_2)$  is *good* if  $\mathcal{A}$  could  $\epsilon$ -distinguish  $\tilde{f}_n(w) \circ \mathcal{U}_{d'} \circ \text{HC}(x, \mathcal{U}_{d'})$  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  $\epsilon$ .

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:

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

runs the algorithm  $R^{\mathcal{O}}$  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  $\mathcal{O}$  indeed  $\epsilon$ -distinguishes  $\mathcal{U}_{d'} \circ \text{HC}(x, \mathcal{U}_{d'})$  from  $\mathcal{U}_{d'+k}$ . Therefore, w.p.  $\geq 1/2$ ,  $x$  is in the list outputted by  $R^{\mathcal{O}}$ . 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.  $\geq \epsilon/2L$ . Finally, as  $R$  is a polynomial-size oracle circuit (actually a PPT oracle machine), the size of  $\mathcal{A}'$  is  $s(n) \cdot \text{poly}(n/\epsilon)$ . ◀

► **Lemma 89.** *Suppose that  $\epsilon < \frac{1}{10n^2}$ . Then  $\text{H}(G_{n,r}(\mathcal{E}_{\tilde{n}})) \geq \tilde{n} - \tau - 2$ .*

**Proof.** Since  $\text{SD}(\tilde{f}_n(\mathcal{X}_n, \mathcal{U}_{2d}), \mathcal{U}_{\ell(n)}) \leq 2\epsilon < \frac{1}{\ell(n)^2}$ , by [62, Lemma 2.2], we have  $\text{H}(\tilde{f}_n(\mathcal{X}_n, \mathcal{U}_{2d})) \geq \ell(n) - 2$ . It follows that  $\text{H}(G_{n,r}(\mathcal{E}_{\tilde{n}})) \geq (\ell(n) - 2) + d' \geq \tilde{n} - \tau - 2$ . ◀

## A.5 Putting It Together

**Proof of Theorem 57.** Suppose that for every  $\mathcal{X} = \{\mathcal{X}_n\}$  that satisfies the premise of Claim 86, and  $\mathcal{E}_{\tilde{n}}$  defined above, there is an adversary of size  $s(n)$  that  $4\epsilon$ -distinguishes  $G_{n,r}(\mathcal{E}_{\tilde{n}})$  from the uniform distribution. Then:

- By Lemma 88, there is an adversary of size  $s(n) \cdot \text{poly}(n/\epsilon)$  that on input  $\tilde{f}(\mathcal{X}_n, \mathcal{U}_{d_1+d_2})$ , guesses  $\mathcal{X}_n$  w.p.  $\geq \epsilon/2L$ .
- By Lemma 87, there is an adversary of size  $s(n) \cdot \text{poly}(n/\epsilon)$  that inverts  $f_n(\mathcal{X}_n)$  w.p.  $\geq \frac{1}{2\epsilon L}$ .

It follows from Claim 86 that there is an adversary of size  $s \cdot \text{poly}(nL/\epsilon)$  that inverts  $f$  on a  $1 - \delta$  fraction of inputs. ◀

## B Proof of Theorem 62

In this section, we briefly review the universal hash functions in [56] 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 60. An XOR-circuit is a (multi-output) circuit that only uses XOR gates of fan-in 2. To match Definition 60, we also require that every gate in an XOR-circuit has fan-out at most 2.

For convenience, we denote  $[n] = \{0, 1, \dots, n-1\}$ , and  $(n) = \{0, 1\}^n$ .

**Outline.** Our start point is the strongly explicit family of expanders by [64, 29]. Spielman [78] 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  $\epsilon > 0$ , one could construct error-correcting codes with relative distance  $1 - \epsilon$ , constant rate, and constant alphabet size. By the construction of [56], such codes imply universal hash functions.

## B.1 Strongly Explicit Expanders

We use the following construction due to [64, 29]. (See also [45, 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  $\gamma_1, \dots, \gamma_4$  are bijections, and the other four neighbors of  $v$  are simply  $\gamma_1^{-1}(v), \dots, \gamma_4^{-1}(v)$ . The graph might contain self-loops or parallel edges.

► **Theorem 90** ([29]). *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 91.** 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  $\sigma_1\sigma_2 \dots \sigma_k$  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  $(\sigma_1 \circ \sigma_2 \circ \dots \circ \sigma_k)(u) = v$ . Each edge has two representations:  $(u, \sigma_1\sigma_2 \dots \sigma_k)$  or  $(v, \sigma_k^{-1}\sigma_{k-1}^{-1} \dots \sigma_1^{-1})$ . We arbitrarily choose a size- $(d/2)$  subset  $S$  of  $\Sigma^k$  such that for each  $\sigma_1, \sigma_2, \dots, \sigma_k \in \Sigma$ , exactly one of  $\sigma_1\sigma_2 \dots \sigma_k$  and  $\sigma_k^{-1}\sigma_{k-1}^{-1} \dots \sigma_1^{-1}$  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  $\sigma_1\sigma_2 \dots \sigma_k \in S$ , and the edge corresponding to  $i$  is represented as  $(v, \sigma_1\sigma_2 \dots \sigma_k)$ . This bijection and its inverse are computable in time  $O(\log n)$ .

## B.2 Error-Reduction Codes

An intermediate step in [78] is to construct *error-reduction codes*, which are weaker primitives compared to error-correcting codes.

Let  $r, \delta, \epsilon > 0$  be constants. Recall that we defined  $(n) = \{0, 1\}^n$  for convenience. An error-reduction code of rate  $r$ , error reduction  $\epsilon$  and reducible distance  $\delta$  is a function  $\mathcal{C} : (rn) \rightarrow ((1-r)n)$  mapping  $rn$  “message” bits into  $(1-r)n$  “check” bits, such that the following holds. The codeword of a message  $x$  is  $x \circ \mathcal{C}(x)$ . For any message  $x$ , if we are given a corrupted codeword that differs from  $x \circ \mathcal{C}(x)$  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  $x \circ \mathcal{C}(x)$  in at most  $\epsilon t$  bits. (We will not be particularly interested in the complexity of recovery or decoding algorithms.)

► **Lemma 92.** *For some absolute constants  $\epsilon < 1$  and  $\delta > 0$ , there is a family of error-reduction codes  $\mathcal{R} = \{\mathcal{R}_n : (n) \rightarrow (\lfloor n/2 \rfloor)\}$  with error-reduction  $\epsilon$  and reducible distance  $\delta$ . Moreover, the sequence of functions  $\{\mathcal{R}_n\}$  can be computed by a uniform family of linear-size XOR-circuits.*

**Proof Sketch.** First, let  $m$  be the smallest integer such that  $dm^2/2 \geq n$ . Note that  $m$  can be computed in  $O(\log n)$  time. Let  $r = 9/10$ , then for large enough  $n$ ,  $dm^2(1-r)/r \leq n/2$ . It suffices to construct an error-reduction code with  $dm^2/2$  message bits and  $dm^2(1-r)/r$  check bits.

We use [78, Definition 16], where  $B$  is the edge-vertex incidence graph of  $\mathcal{G}_m^k$ , and  $\mathcal{S}$  is 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.)

An equivalent formulation is as follows. 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. By [78, Lemma 18], for some absolute constants  $\epsilon < 1$  and  $\delta > 0$ , this error-reduction code has error-reduction  $\epsilon$  and reducible distance  $\delta$ .

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 91, this is computable in  $O(\log n)$  time. ◀

We actually need error-reduction codes with error-reduction  $\epsilon = 1/2$ . We can simply iterate the code in Lemma 92 for  $O(1)$  times. The encoding circuit is still uniform. Therefore, we have:

► **Corollary 93.** *Lemma 92 holds for  $\epsilon = 1/2$ .*

### B.3 Error-Correcting Codes

The construction in [78, 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 [78].

► **Lemma 94.** *There is a constant  $n_0 > 1$  and a family  $\mathcal{C} = \{\mathcal{C}_k : (n_0 2^{k-2}) \rightarrow (n_0 2^k)\}$  of error-correcting codes with constant relative distance. Moreover,  $\mathcal{C}$  can be encoded by a uniform family of linear-size XOR circuits.*

**Proof Sketch.** We recursively define  $\mathcal{C}_k$  as follows. First,  $\mathcal{C}_0 : (n_0/4) \rightarrow (n_0)$  is any good enough error-correcting code. Since  $n_0$  is a constant, our algorithm can hardcode  $\mathcal{C}_0$ .

Now, let  $k \geq 1$ , we define  $\mathcal{C}_k$  as follows. Let  $x \in (n_0 2^{k-2})$  be the inputs of  $\mathcal{C}_k$ .

- The first  $n_0 2^{k-2}$  outputs of  $\mathcal{C}_k$  will always be  $x$  itself.<sup>19</sup> 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  $\mathcal{R}_{k-2} : (n_0 2^{k-2}) \rightarrow (n_0 2^{k-3})$ , and output  $A_k := \mathcal{R}_{k-2}(x)$ .
- Let  $\mathcal{C}_{k-1} : (n_0 2^{k-3}) \rightarrow (n_0 2^{k-1})$  be the error-correcting code we recursively defined. Let  $A_k \circ B_k := \mathcal{C}_{k-1}(A_k)$ , and we output  $B_k$ . (Recall that the first  $n_0 2^{k-3}$  outputs of  $\mathcal{C}_{k-1}$  is equal to its inputs, i.e.,  $A_k$ .)
- We pick an error-reduction code  $\mathcal{R}_{k-1} : (n_0 2^{k-1}) \rightarrow (n_0 2^{k-2})$ , and output  $C_k := \mathcal{R}_{k-1}(A_k \circ B_k)$ .

<sup>19</sup>We can assume this is also true for  $\mathcal{C}_0$ .



The required error-reduction codes are constructed in Corollary 93. The total number of output bits of  $\mathcal{C}_k$  is  $|x| + |A_k| + |B_k| + |C_k|$  which is indeed  $n_0 2^k$ .

The  $i$ -th gate of the encoding circuit of  $\mathcal{C}_{k-1}$  can be computed as follows. Let  $c 2^k$  be the circuit complexity of the first, second, and fourth bullet. (That is, circuit complexity of  $\mathcal{C}_k$  not counting the recursive part for  $\mathcal{C}_{k-1}$ .) We may assume  $c$  is a power of 2. The encoding circuit for  $\mathcal{C}_k$  has  $|C_0| + \sum_{i=1}^k c 2^i = |C_0| + c(2^{k+1} - 1)$  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  $\epsilon > 0$ , the construction of [56] needs a code with relative distance  $1 - \epsilon$  and constant alphabet size. As in [7, 35], we can “amplify” the code in Lemma 94 by an expander:

► **Lemma 95.** *For every constant  $\epsilon > 0$ , there is a constant  $D > 0$  and a family of error-correcting codes  $\{\mathcal{C}'_k : (n_0 2^{k-2}) \rightarrow [2^D]^{O(2^k)}\}$  that has relative distance  $1 - \epsilon$ . Moreover, if we interpret  $[2^D]$  as length- $D$  strings, then  $\mathcal{C}'_k$  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 90, and  $\{\mathcal{C}_k : (n_0 2^{k-2}) \rightarrow (n_0 2^k)\}$  is the family of error-correcting codes constructed in Lemma 94. Let  $m$  be the smallest integer such that  $m^2 \geq n_0 2^k$ . We pad zeros to the outputs of  $\mathcal{C}_k$ , thus  $\mathcal{C}_k$  can be regarded as a code that outputs  $m^2$  bits. We assign an output bit of  $\mathcal{C}_k$  to each vertex in  $\mathcal{G}_m$ . The relative distance of  $\mathcal{C}_k$  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  $(1 - \epsilon)m^2$  neighbors. (See e.g. [7, Corollary 1].) Let  $D := 8^p$ , so every vertex in  $\mathcal{G}_m^p$  has degree  $D$ . On input  $x \in (n_0 2^{k-2})$ , recall that we assigned each vertex in  $V(\mathcal{G}_m^p)$  a bit of the codeword  $\mathcal{C}_k(x)$ . 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  $\mathcal{C}'_k(x)$  simply concatenates the outputs of each vertex  $v \in V(\mathcal{G}_m^p)$  together.

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

► **Remark 96.** Lijie Chen (personal communication) suggested a similar approach based on expander random walks [80, 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.

## B.4 Universal Hash Functions

Finally, we are ready to verify that the universal hash functions in [56] are uniform.

► **Theorem 62.** *For every integer  $n, m$  where  $m = O(n)$ , there exists an integer  $k = O(n)$ , and a family of universal hash functions  $\{h_{n,m} : \{0, 1\}^n \times \{0, 1\}^k \rightarrow \{0, 1\}^m\}$ , 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.*

## 35:58 Hardness of KT Characterizes Parallel Cryptography

**Proof Sketch.** Let  $n_1 := cn$  for a large enough constant  $c$ ,  $\epsilon$  be a small enough constant, and  $D$  be the constant in Lemma 95 depending on  $\epsilon$ . We need three ingredients:

- An  $\ell$ -exposure resilient function (ERF)  $\text{ERF} : \{0, 1\}^{n_1 D} \rightarrow \{0, 1\}^m$  [21]. It is shown in [24] that for any (linear) error-correcting code  $\mathcal{C} : \{0, 1\}^m \rightarrow \{0, 1\}^n$  with generator matrix  $G$  and minimum distance  $d$ , the transpose matrix  $G^T$  mapping  $n$  input bits to  $m$  output bits is a perfect  $\ell$ -ERF where  $\ell := n - d + 1$ .

For an XOR-circuit  $C$  that computes the linear transform  $G$  over  $\text{GF}(2)$ , 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 [18, 56]. In particular, every gate  $g \in C$  whose output feeds to the gates  $g_1, g_2, \dots, g_k$  becomes, in the new circuit, an XOR gate  $g$  whose *inputs* are  $g_1, g_2, \dots, g_k$ .

Therefore, Lemma 94 shows that an  $\ell$ -ERF  $\text{ERF} : \{0, 1\}^{n_1 D} \rightarrow \{0, 1\}^m$  is computable by a uniform family of linear-size XOR circuits.

- An error-correcting code  $\mathcal{C}'_k : \{0, 1\}^n \rightarrow [2^D]^{n_1}$  with relative distance  $1 - \epsilon$ , as in Lemma 95.
- A hash family  $H : \{0, 1\}^D \times \{0, 1\}^{2^{D-1}} \rightarrow \{0, 1\}^D$ , 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 [56] goes as follows. On input  $x \in \{0, 1\}^n$ , we first compute  $\mathcal{C}'(x) \in [2^{h+1}]^{n_1}$ . Next, we receive  $n_1$  keys  $k_1, k_2, \dots, k_{n_1} \in \{0, 1\}^{2^{h+1}}$  which are the keys for our hash function. Let  $t \in [2^{h+1}]^{n_1}$  be the following message:  $t_i := H(\mathcal{C}'(x)_i, k_i)$ . We treat  $t$  as a string of length  $m(h+1)$ , and the output of our hash function is  $\text{ERF}(t)$ .

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