Search Problems: A Cryptographic Perspective

Eylon Yogev

Under the Supervision of Professor Moni Noar
Department of Computer Science and Applied Mathematics
Weizmann Institute of Science

Submitted for the degree of Doctor of Philosophy
to the Scientific Council of the Weizmann Institute of Science

April 2018

Dedicated to Lior

Abstract

Complexity theory and cryptography live in symbiosis. Over the years, we have seen how hard computational problems are used for showing the security of cryptographic schemes, and conversely, analyzing the security of a scheme suggests new areas for research in complexity. While complexity is concentrated traditionally on decision problems, an important sub field involves the study of search problems. This leads us to study the relationship between complexity theory and cryptography in the context of search problems.

The class TFNP is the search analog of NP with the additional guarantee that any instance has a solution. This class of search problems has attracted extensive attention due to its natural syntactic subclasses that capture the computational complexity of important search problems from algorithmic game theory, combinatorial optimization and computational topology.

One of the main objectives in this area is to study the hardness and the complexity of different search problems in TFNP. There are two common types of hardness results: white-box and black-box. While hardness results in the black-box model can be proven unconditionally, in the white-box setting, where the problem at hand is given explicitly and in a succinct manner (e.g., as a polynomially sized circuit), cryptographic tools are necessary for showing hardness.

This cryptographic view of the TFNP class gives rise to a variety of new research directions exploring the relationship between total search problems and cryptography and complexity hardness. In this thesis, we exploit this connection and show several hardness results of TFNP problems both in the black-box and the white-model.

Acknowledgements

In October 2011, I stepped into a class on Theoretical Cryptography taught by Moni Naor. Never would I imagine the significance of that day, which turned out to be my first step down the cryptographic rabbit hole. To my great fortune and gratitude, Moni agreed to be my advisor and guide me through this meaningful and fascinating quest. Moni is an incredible advisor and nowadays, as I see it, the philosophical father of cryptography. During the past 7 years, he showed me how to spot a hard problem, turn it up-side-down, attack the adversary with a sharp pseudorandom object and finally construct a secure and elegant protocol. But Moni's influence on me reaches far beyond brilliant theorems, ingenious constructions and unimaginable lower bounds. It is a part of my day-to-day thinking, habits and perspective on life. Thank you Moni for the best years of my life!

Research is often a team effort. I had the pleasure of working with many talented researchers, each with their own perspective on computer science; Prabhanjan Ananth, Shai Halevi, Pavel Hubáček, Yuval Ishai, Aayush Jain, Abhishek Jain, Ilan Komargodski, Tal Moran, Merav Parter, Rafael Pass, Alon Rosen, Amit Sahai and Gil Segev. Thank you all for the wonderful collaboration, I am looking forward to continue working with all of you in the future. Moreover, special thanks to Karthik C. S., Inbal Livni Navon and Roei Tell for creating a nurturing academic office space, where we share ideas verify proofs and help each other in any way needed.

I would like to thank Zvika Brakerski and Robi Krauthgamer for serving as my Ph.D. committee, and making sure that I am focused on the right goals.

To my dear family, I am grateful for your infinite support over the years. My Parents, Nomi and David, your confidence in me has encouraged me to believe in myself and shoot for the stars. My beloved wife, Romi, I am thankful for your patience and your utter devotion to me and my passions. We met during my bachelor's and you have been encouraging me to push my limits ever since. And last but not least, thank you for bringing Lior to our lives. You two shine even the rainiest of my days.

Contents

1 Introduction **1**
- 1.1 Our Results . . . 3
  - 1.1.1 The Journey from NP to TFNP Hardness . . . 3
  - 1.1.2 Hardness of Continuous Local Search: Query Complexity and Cryptographic Lower Bounds . . . 3
  - 1.1.3 White-Box vs. Black-Box Complexity of Search Problems: Ramsey and Graph Property Testing . . . 4
  - 1.1.4 Collision Resistant Hashing for Paranoids: Dealing with Multiple Collisions . . . 5
  - 1.1.5 Summary of Results . . . 5
- 1.2 Cryptography and Data Structures . . . 5
- 1.3 Organization . . . 7

2 Preliminaries **9**
- 2.1 Information Theoretic Tools . . . 9
  - 2.1.1 Limited independence . . . 9
  - 2.1.2 Randomness extractors . . . 10
  - 2.1.3 List-recoverable codes . . . 11
  - 2.1.4 Nisan-Wigderson type pseudorandom generators . . . 12
- 2.2 Cryptographic Tools . . . 13
  - 2.2.1 Universal one-wayness . . . 14
  - 2.2.2 Lossy functions and collision resistant hash functions . . . 15
  - 2.2.3 Commitment schemes . . . 16
- 2.3 Fully black-box constructions . . . 17
  - 2.3.1 Witness indistinguishable proof systems . . . 18
- 2.4 Average-case complexity . . . 19
- 2.5 Search Problems . . . 19
  - 2.5.1 Black-box and white-box models . . . 21
- 2.6 Ramsey theory . . . 23

3 The Journey from NP to TFNP Hardness **27**
- 3.1 Introduction . . . 27
- 3.2 Our results . . . 30

3.3 Related work (search vs. decision) . . . 30
3.4 Technical Overview . . . 31
3.4.1 TFNP hardness based on average-case NP hardness . . . 31
3.4.2 Using zero-knowledge for TFNP hardness . . . 33
3.5 TFNP Hardness from Average-Case NP Hardness . . . 34
3.6 Hardness based on public-coin distributions . . . 35
3.7 From private coins to public coins . . . 37
3.8 Using Zero-Knowledge for TFNP Hardness . . . 38
3.9 Chapter Appendix . . . 40
3.9.1 Proof of Lemma 1 . . . 40
3.9.2 Proof of Theorem 7 . . . 41

4 Hardness of Continuous Local Search: Query Complexity and Cryptographic Lower Bounds 45
4.1 Introduction . . . 45
4.2 Our Results . . . 48
4.3 Our Techniques . . . 51
4.3.1 Reduction to Continuous-Local-Optimum . . . 51
4.3.2 Query Complexity Lower Bound for End-of-Metered-Line . . . 54
4.3.3 Cryptographic Hardness for End-of-Metered-Line . . . 55
4.3.4 Organization of the Paper . . . 56
4.4 End-of-Metered-Line . . . 57
4.5 End-of-Metered-Line is in CLS . . . 59
4.5.1 Proof of Theorem 12 . . . 64
4.6 On the Hardness of End-of-Metered-Line . . . 69
4.6.1 Query Complexity Lower Bound . . . 69
4.6.2 Cryptographic Hardness of End-of-Metered-Line . . . 72
4.7 Chapter Appendix . . . 76
4.7.1 Proof of f and p being Lipschitz . . . 76
4.7.2 Analysis of Templates from Lemma 3 . . . 78
4.7.3 Our Algorithms' Pseudocode . . . 86
4.7.4 Pseudocode for S' and P' . . . 88
4.7.5 Obfuscation Definitions . . . 91

5 White-Box vs. Black-Box Complexity of Search Problems: Ramsey and Graph Property Testing 95
5.1 Introduction . . . 95
5.1.1 Graph-hash product . . . 99
5.1.2 Cryptographic assumptions and white-box lower bounds . . . 99
5.2 Hardness of The Ramsey Problem . . . 100
5.2.1 Hardness of the colorful Ramsey problem . . . 103
5.2.2 The relation of Ramsey and the WeakPigeon classes . . . 105
5.2.3 The Ramsey problem and Multi-CRH . . . 106

5.3 Hardness of Testing an Extractor . . . 108
5.4 Impossibility of a General Transformation . . . 112
5.5 The Succinct Black-Box Model . . . 114
5.6 Further Research . . . 116
5.7 Chapter Appendix . . . 117
5.7.1 Hardness of Finding a Sunflower . . . 117
5.7.2 An upper bound on colorful Ramsey . . . 119
5.7.3 A lower bound for Ramsey graphs . . . 120

6 Collision Resistant Hashing for Paranoids: Dealing with Multiple Collisions 123
6.1 Introduction . . . 123
6.2 Our Techniques . . . 128
6.2.1 The main commitment scheme . . . 128
6.2.2 Separating Multi-CRH from standard CRH . . . 132
6.3 Multi-Collision-Resistant Function Families . . . 133
6.4 Tree Commitments from Multi-CRH . . . 134
6.4.1 A computationally-binding scheme . . . 136
6.4.2 Getting statistical-hiding generically . . . 142
6.5 Four-Round Short Commitments from Multi-CRH . . . 145
6.6 Separating Multi-CRH from CRH . . . 150
6.6.1 Proof of Lemma 9 . . . 152
6.6.2 Proof of Claim 11 . . . 155
6.6.3 Proof of Claim 12 . . . 157
6.7 UOWHFs, MCRHs and Short Commitments . . . 160
6.8 Chapter Appendix . . . 163
6.8.1 Multi-Pair Collision Resistance . . . 163
6.8.2 On TFNP and Multi-CRH . . . 164

Chapter 1

Introduction

Complexity theory and cryptography live in symbiosis. Over the years, we have seen how hard computational problems are used for showing the security of cryptographic schemes, and conversely, analyzing the security of a scheme suggests new areas for research in complexity. While complexity is concentrated traditionally on decision problems, an important sub field involves the study of search problems. Search problems are central to computer science and contain fundamental problems ranging from algorithmic game theory and combinatorial optimization to computational topology. This leads us to study the relationship between complexity theory and cryptography in the context of search problems.

The class NP captures all *decision* problems for which the "yes" instances have efficiently verifiable proofs. The study of this class and its computational complexity are at the heart of theoretical computer science. Part of the effort has been to study the *search* analog of NP which is defined by the class FNP (for Function NP) and captures all search problems for which verifying a solution can be done efficiently. Megiddo and Papadimitriou [MP91] introduced the complexity class TFNP (for Total Function NP) which is a subclass of FNP with the additional property of the problem being *total*, i.e., for any instance a solution is guaranteed to exist. For this reason, the class TFNP is usually considered as the search variant containing $\mathsf{NP} \cap \mathsf{coNP}$: in particular, for any language $L \in \mathsf{NP} \cap \mathsf{coNP}$, the corresponding search problem is given by an instance x and a solution is either a witness verifying that $x \in L$ or a witness verifying that $x \notin L$. Since $L \in \mathsf{NP} \cap \mathsf{coNP}$, such a solution must always exist.

Beyond its natural theoretical appeal, TFNP attracted extensive attention due to its important syntactic subclasses. The common way of defining such subclasses is via various non-constructive arguments used for proving totality of search problems. For example, the *parity argument for directed graphs*: "If a directed graph has an unbalanced node (a vertex with unequal in-degree and out-degree), then it must have another unbalanced node," gives rise to the class PPAD (for Polynomial Parity Argument on Directed graphs [Pap94]). This might be the most famous subclass of TFNP due to the fact that one of its complete problems is finding Nash equilibria in strategic games [DGP09, CDT09]. Other known subclasses are PPP (for Polynomial

Pigeonhole Principle [Pap94]), PLS (for Polynomial Local Search [JPY88]), and CLS (for Continuous Local Search [DP11]).

One of the main objectives in this area is to study the hardness and the complexity of different search problems in TFNP. There are two common types of hardness results: white-box and black-box. In the white-box setting, the problem at hand is given explicitly and in a succinct manner (e.g., as a polynomially sized circuit). Then, hardness is shown via a reduction from a different problem for which hardness is already established (or assumed). In the black-box setting (also known as the query model, or the decision tree model) the problem is represented via oracle access, and the complexity is measured by the number of performed oracle queries. In this model, it is often possible to prove unconditional lower bounds on the number of queries required to find a solution.

In terms of black-box hardness there has been some progress throughout the years. Exponential query complexity lower bounds for finding Brouwer fixed points (a complete problem for PPAD) [HPV89, Bab14] have been established. Moreover, a series of exponential query complexity lower bounds for Local-Search [Ald83, LTT89, Aar06, Zha09] give black-box hardness for PLS for deterministic, randomized and even quantum algorithms. On the other hand, white-box hardness results are much more scarce.

It is easy to see that if $\mathsf{P} = \mathsf{NP}$ then all search problems in TFNP can be solved efficiently. Therefore, the study of the hardness of TFNP classes must rely on some hardness assumptions (until the $\mathsf{P} \stackrel{?}{=} \mathsf{NP}$ question is resolved), and in particular on cryptographic hardness. For instance, the (average-case) hardness of the subclass PPP has been based on the existence of either one-way permutations[1][Pap94] or collision-resistant hash[2] [Jer16]. For other subclasses even less standard assumptions have been used. The hardness of PPAD has been based on strong cryptographic assumptions, e.g., indistinguishability obfuscation and functional encryption [BPR15, GPS16].

To understand the hierarchy of cryptographic assumptions it is best to turn to Impagliazzo's worlds [Imp95]. He described five possible worlds: `Algorithmica` (where $\mathsf{P} = \mathsf{NP}$), `Heuristica` (where NP is hard in the worst case but easy on average, i.e., one simply does not encounter hard problems in NP), `Pessiland` (where hard-on-average problems in NP exist, but one-way functions do not exist), `Minicrypt` (where one-way functions exist[3]), and `Cryptomania` (where Oblivious Transfer exists[4]). Nowadays, it is possible to add a sixth world, `Obfustopia` where indistinguishability obfuscation for all of P is possible [BGI+01, GGH+13a, SW14a].

[1] A permutation is one-way if it is easy to compute on every input, but hard to invert on a random image. Such permutations exist only if $\mathsf{P} \neq \mathsf{NP} \cap \mathsf{coNP}$ [Bra83].

[2] A collision-resistant hash is a hash function such that it is hard to find two inputs that hash to the same output. Simon [Sim98] showed a *black-box separation* between one-way functions and collision-resistant hashing.

[3] For many primitives such as shared-key encryption, signatures, and zero-knowledge proofs for NP it is known that their existence is equivalent to the existence of one-way functions.

[4] Roughly speaking, this world is where public-key cryptography resides.

1.1 Our Results

This cryptographic view of the TFNP class gives rise to a variety of new research directions exploring the relationship between total search problems and cryptography and complexity hardness. In this thesis, we exploit this direction and show several hardness results of TFNP problems both in the black-box and the white-model. We show the following results:

1.1.1 The Journey from NP to TFNP Hardness

Currently, no problem in TFNP is known to be hard under assumptions such as NP hardness, the existence of one-way functions, or even public-key cryptography. The only known hardness results are based on less general assumptions such as the existence of collision-resistant hash functions, one-way permutations less established cryptographic primitives (e.g., program obfuscation or functional encryption).

Several works explained this status by showing various barriers to proving hardness of TFNP. In particular, it has been shown that hardness of TFNP hardness cannot be based on worst-case NP hardness, unless NP $=$ coNP. Therefore, we ask the following question: What is the weakest assumption sufficient for showing hardness in TFNP?

In this thesis, we answer this question and show that hard-on-average TFNP problems can be based on the weak assumption that there exists a *hard-on-average* language in NP. In particular, this includes the assumption of the existence of one-way functions. In terms of techniques, we show an interesting interplay between problems in TFNP, derandomization techniques, and zero-knowledge proofs. The full details are given in Chapter 3 and are based on [HNY17].

1.1.2 Hardness of Continuous Local Search: Query Complexity and Cryptographic Lower Bounds

Local search proved to be an extremely useful tool when facing hard optimization problems (e.g., via the simplex algorithm, simulated annealing, or genetic algorithms). Although powerful, it has its limitations: there are functions for which exponentially many queries are needed to find a local optimum. In many contexts the optimization problem is defined by a continuous function, which might offer an advantage when performing the local search. This leads us to study the following natural question: How hard is *continuous* local search? The computational complexity of such search problems is captured by the complexity class CLS which is contained in the intersection of PLS and PPAD.

In this work, we show the first hardness results for CLS (the smallest non-trivial class among the currently defined subclasses of TFNP). Our hardness results are in terms of black-box (where only oracle access to the function is given) and white-box (where the function is represented succinctly by a circuit). In the black-box case, we

show instances for which any (computationally unbounded) randomized algorithm must perform exponentially many queries in order to find a local optimum. In the white-box case, we show hardness for computationally bounded algorithms under cryptographic assumptions. Our results demonstrate a strong conceptual barrier precluding design of efficient algorithms for solving local search problems even over continuous domains.

As our main technical contribution we introduce a new total search problem which we call End-of-Metered-Line. The special structure of End-of-Metered-Line enables us to: (1) show that it is contained in CLS, (2) prove hardness for it both in the black-box and the white-box setting, and (3) extend to CLS a variety of results previously known only for PPAD. The full details are given in Chapter 4 and are based on [HY17].

1.1.3 White-Box vs. Black-Box Complexity of Search Problems: Ramsey and Graph Property Testing

Ramsey theory assures us that in any graph there is a clique or independent set of a certain size, roughly logarithmic in the graph size. But how difficult is it to find the clique or independent set? If the graph is given explicitly, then it is possible to do so while examining a linear number of edges. If the graph is given by a black-box, where to figure out whether a certain edge exists the box should be queried, then a large number of queries must be issued. But what if one is given a program or circuit for computing the existence of an edge? This problem was raised by Buss and Goldberg and Papadimitriou in the context of TFNP search problems.

We examine the relationship between black-box complexity and white-box complexity for search problems with guaranteed solution such as the above Ramsey problem. We show that under the assumption that collision resistant hash function exist (which follows from the hardness of problems such as factoring, discrete-log and learning with errors) the white-box Ramsey problem is hard and this is true even if one is looking for a much smaller clique or independent set than the theorem guarantees. This is also true for the colorful Ramsey problem where one is looking, say, for a monochromatic triangle.

In general, one cannot hope to translate all black-box hardness for TFNP into white-box hardness: we show this by adapting results concerning the random oracle methodology and the impossibility of instantiating it.

Another model we consider is that of succinct black-box, where there is a limitation on the size of the black-box (but no limitation on the computation time). In this case we show that for all TFNP problems there is an upper bound proportional to the description size of the box times the solution size. On the other hand, for promise problems this is not the case.

Finally, we consider the complexity of graph property testing in the white-box model. We show a property which is hard to test even when one is given the program for computing the graph (under the appropriate assumptions such as hardness of

Decisional Diffie-Hellman). The hard property is whether the graph is a two-source extractor. The full details are given in Chapter 5 are are based on [KNY17].

1.1.4 Collision Resistant Hashing for Paranoids: Dealing with Multiple Collisions

A collision resistant hash (CRH) function is one that compresses its input, yet it is hard to find a collision, i.e. a $x_1 \neq x_2$ s.t. $h(x_1) = h(x_2)$. Collision resistant hash functions are one of the more useful cryptographic primitives both in theory and in practice and two prominent applications are in signature schemes and succinct zero-knowledge arguments.

In this work we consider a relaxation of the above requirement that we call Multi-CRH: a function where it is hard to find $x_1, x_2, \ldots, x_k$ which are all distinct, yet $h(x_1) = h(x_2) = \cdots = h(x_k)$. We show that for some of the major applications of CRH functions it is possible to replace them by the weaker notion of a Multi-CRH, albeit at the price of adding interaction: we show a constant-round statistically-hiding commitment scheme with succinct interaction (committing to $\mathsf{poly}(n)$ bits requires exchanging $\tilde{O}(n)$ bits) that can be opened locally (without revealing the full string). This in turn can be used to provide succinct arguments for any NP statement.

We formulate four possible worlds of hashing-related assumptions (in the spirit of Impagliazzo's worlds). They are (1) *Nocrypt*, where no one-way functions exist, (2) *Unihash*, where one-way functions exist, and hence also UOWHFs and signature schemes, but no Multi-CRH functions exist, (3) *Minihash*, where Multi-CRH functions exist but no CRH functions exist, and (4) *Hashomaia*, where CRH functions exist. We show that these four worlds are distinct in a black-box model: we show a separation of CRH from Multi-CRH and a separation of Multi-CRH from one-way functions. The full details are given in Chapter 6 and are based on [KNY18].

1.1.5 Summary of Results

A short summary of our results is given in Table 5.1. For each problem (or class of problems) in TFNP we describe its hardness under the different computation models.

1.2 Cryptography and Data Structures

Our results have demonstrated a strong connection between cryptography and search problems. But cryptography lives in symbiosis with other fields as well. One particular example is data structures, which has been studied in [NY15a] (other examples include [BHKN13, NSW08]). However, in order not to divert the focus from search problems we have decided not to include the detailed results in this thesis. Below we give a short exposition of the results for the curious reader.

Problem	Black-Box	White-Box	Succinct Black-Box
PPP	Hard [Folklore]	CRH/OWP [Pap94]	Easy [Section 5.5]
PPAD	Hard [HPV89]	Obfuscation [BPR15]	Easy [Section 5.5]
CLS	Hard [Chapter 4]	Obfuscation [Chapter 4]	Easy [Section 5.5]
PLS	Hard [Chapter 4]	Obfuscation [Chapter 4]	Easy [Section 5.5]
Problem $\mathcal{R}$	Hard [Section 5.4]	Easy [Section 5.4]	Easy [Section 5.5]
Problem $\mathcal{D}$	Hard [Chapter 3]	Average NP [Chapter 3]	Easy [Section 5.5]
Ramsey-like	Hard [IN88]	MCRH [Section 5.2]	Easy [Section 5.5]
TFNP	Hard	Hard	Easy [Section 5.5]

Table 1.1: A summary of our results on the complexity of search problems. For each problem, in each model we mention its hardness and under which assumption (if any). The problem denoted by $\mathcal{R}$ is the problem defined in Section 5.4 and is designed to be hard in the black-box model but easy given any white-box implementation. The problem denoted by $\mathcal{D}$ is the problem defined in Section 3.5 which we show that is hard in the white-box model assuming average-case NP hardness. The term "Ramsey-like" problems refers to the problems we consider in Section 5.2, including Ramsey's problem, the colorful Ramsey problem, and the bipartite Ramsey problem and problem $\mathcal{D}$ refers to the problem defined in Section 3.5.

Data structures are one of the most basic objects in Computer Science. They provide means to organize a large amount of data such that it can be queried efficiently. In general, constructing efficient data structures is key to designing efficient algorithms. Many efficient data structures use randomness, a resource that allows them to bypass lower bounds on deterministic ones. In these cases, their efficiency and/or correctness are analyzed in expectation or with high probability.

To analyze randomized data structures one must first define the underlying model of the analysis. Usually, the model assumes that the inputs (equivalently, the queries) are *independent* of the internal randomness of the data structure. That is, the analysis is of the form: For any sequence of inputs, with high probability (or expectation) over its internal randomness, the data structure will yield a correct answer. This model is reasonable in a situation where the adversary picking the inputs gets no information about the internal state of the data structure or about the random bits actually used (in particular, the adversary does not get the responses on previous inputs).

We consider data structures in a more robust model, which we call the *adversarial model*. Roughly speaking, this model allows an adversary to choose inputs and queries *adaptively* according to previous responses. That is, the analysis is of the form: With high probability over the internal randomness of the data structure, for any adversary adaptively choosing a sequence of inputs, the response to a single query will be correct. Specifically, we consider a data structure known as "Bloom filter" [Blo70, NY15b] and prove a tight connection between Bloom filters in this model and cryptography.

A Bloom filter is a succinct representations of a set S of elements from a large universe U, where the price for succinctness is allowing some errors. It is required to answer queries in the following manner: for any $x \in S$ it should always answer 'Yes', and for any $x \notin S$ it should answer 'Yes' only with small probability. False responses for $x \notin S$ are called *false positive* errors.

Under the adversarial model, we construct Bloom filters that are resilient to adaptive attacks. We consider both efficient adversaries (that run in polynomial time) and computationally unbounded adversaries that are only bounded in the amount of queries they can make. We define a Bloom filter that maintains its error probability in this setting and say it is *adversarial resilient*.

Our first result is that adversarial-resilient Bloom filters against computationally bounded adversaries that are non-trivial (i.e., they require less space than the amount of space it takes to store the elements explicitly) must use one-way functions. That is, we show that if one-way functions do not exist then any Bloom filter can be 'attacked' with high probability.

In the other direction, we show that using one-way functions one can construct a strongly resilient Bloom filter. Actually, we show that one can transform any Bloom filter to be strongly resilient with almost exactly the same memory requirements and at a cost of a single evaluation of a pseudorandom permutation[5] (which can be constructed using one-way functions).

In practice Bloom filters are used when performance is crucial, and extremely fast implementations are required. This raises implementation difficulties since cryptographically secure functions rely on relatively heavy computation. Nevertheless, we provide an implementation of an adversarial resilient Bloom filter that is provably secure under the hardness of AES, and is essentially fast as any other implementation of insecure Bloom filters. Our implementation exploits the AES-NI[6] instruction set that is embedded in most modern CPUs and provides hardware acceleration of the AES enryption and decryption algorithms [Gue09].

In the context of unbounded adversaries, we show a positive result. For a set of size n and an error probability of ε most constructions use about $O(n \log \frac{1}{\varepsilon})$ bits of memory. We give a construction of a resilient Bloom filter that does not use one-way functions, is resilient against t queries, uses $O(n \log \frac{1}{\varepsilon} + t)$ bits of memory, and has query time $O(\log \frac{1}{\varepsilon})$. See [NY15a] for the full details.

1.3 Organization

Chapter 2 includes definitions and preliminaries that are used throughout the rest of the thesis. Then, each chapter corresponds to a published paper, and may be

[5] A pseudorandom permutation is family of functions that a random function from the family cannot be distinguished from a truly random permutation by any polynomially bounded adversary making queries to the function. It models a block cipher.

[6] Advanced Encryption Standard Instruction Set.

read at an arbitrary order. In Chapter 3 we show hardness of TFNP under the weak assumption that NP is hard on average (based on [HNY17]). In Chapter 5 we consider the hardness of the Ramsey problem, in several computational models (based on [KNY17]), In Chapter 4 we show the hardness of continuous local search (based on [HY17]). In Chapter 6 we study the notion of multi-collision resistance hash (based on [KNY18]).

Chapter 2

Preliminaries

Unless stated otherwise, the logarithms in this paper are base 2. For an integer $n \in N$ we denote by $[n]$ the set $\{1, \ldots, n\}$. We denote by U_n the uniform distribution over n-bit strings. For a distribution $\mathcal{D}$ we denote by $x \leftarrow \mathcal{D}$ an element chosen from $\mathcal{D}$ uniformly at random. We denote by $\circ$ the string concatenation operation. A function $\mathsf{negl}\colon \mathbb{N} \to \mathbb{R}^+$ is *negligible* if for every constant $c > 0$, there exists an integer N_c such that $\mathsf{negl}(n) < n^{-c}$ for all $n > N_c$.

2.1 Information Theoretic Tools

Definition 1 (Statistical Distance)**.** *The statistical distance between two random variables X, Y is defined by*

$$\Delta(X, Y) \triangleq \frac{1}{2} \cdot \sum_x |\Pr[X = x] - \Pr[Y = x]|$$

We say that X and Y are δ-close (resp. -far) if $\Delta(X, Y) \leq \delta$ (resp. $\Delta(X, Y) \geq \delta$).

2.1.1 Limited independence

Definition 2 (k-wise independence)**.** *Fix $n, m, k \in \mathbb{N}$. A function family $\mathcal{G} = \{g\colon \{0,1\}^n \to \{0,1\}^m\}$ is k-wise independent if for every distinct $x_1, \ldots, x_k \in \{0,1\}^n$ and every $y_1, \ldots, y_k \in \{0,1\}^m$ it holds that*

$$\Pr_{g \leftarrow \mathcal{G}}[\forall i \in [k]\colon g(x_i) = y_i] = \frac{1}{2^{km}}.$$

It is known that for every $m \leq n$, there exists a k-wise independent family of functions, where each function is described by $k \cdot n$ bits. One well-known construction which is optimal in terms of size is by letting each $g \in \{0,1\}^{k \cdot n}$ describe a degree $k-1$ polynomial over $\mathsf{GF}[2^n]$. The description of the polynomial requires k field elements

so $k \cdot n$ bits are enough. Evaluation of such a function is merely an evaluation of the polynomial.

In some applications (including some of ours) the input size n is very large and we prefer that the description size of the hash function to be much shorter. To circumvent this, it is sometimes enough to use *almost* k-wise independent functions.

Definition 3 (Almost k-wise independence)**.** *Fix $n, m, k \in \mathbb{N}$ and $\delta \in \mathbb{R}$. A function family $\mathcal{G} = \{g\colon \{0,1\}^n \to \{0,1\}^m\}$ is (k,δ)-wise independent if for every distinct $x_1, \ldots, x_k \in \{0,1\}^n$ the distribution of $(g(x_1), \ldots, g(x_k))$ is δ-close to the distribution $(u_1, \ldots, u_k)$, where $g \leftarrow \mathcal{G}$ and each $u_i \leftarrow \{0,1\}^m$ are chosen uniformly at random.*

It is known that for every $m \leq n$, there exists a (k,δ)-wise independent function with each function $g \in \mathcal{G}$ being described by $O(mk + \log(n/\delta))$ bits [NN93, AGHP92] (see also [Ta-17]).

2.1.2 Randomness extractors

We consider random variables supported on n-bit strings. A random variable X is said to have min-entropy $\mathsf{H}_\infty(X) = k$ if for every $x \in \mathsf{Supp}(X)$ it holds that $\Pr[X = x] \leq 2^{-k}$.

Single Source Extractor. We say that a function $\mathsf{Ext}\colon \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ is a (k,ϵ)-*seeded extractor* if for every distribution X over $\{0,1\}^n$ with min-entropy k, it holds that

$$\Delta(\mathsf{Ext}(X, U_d), U_m) \leq \epsilon.$$

The extractor Ext is said to be *strong* if $\mathsf{Ext}'(x,s) = \mathsf{Ext}(x,s) \circ s$ is a (k,ϵ)-seeded extractor. That is, if

$$\Delta((\mathsf{Ext}(X, U_d) \circ U_d), (U_m \circ U_d)) \leq \epsilon.$$

The famous leftover hash lemma [ILL89, HILL99] says that a pairwise independent function family is a strong extractor.

Proposition 1. *Let $\mathcal{G} = \{g\colon \{0,1\}^n \to \{0,1\}^m\}$ be a pairwise independent family of hash functions where $m = k - 2\log(1/\epsilon)$. Then, $\mathsf{Ext}(x,h) = h(x)$ is a strong (k,ϵ)-seeded extractor.*

Note that the seed length in this extractor equals the number of bits required to sample $g \leftarrow \mathcal{G}$ which is $2n$ bits.

We will also need the following standard proposition that says that conditioning does not reduce entropy by more than the information given by the condition.

Proposition 2. *Let X and Y be random variables. Then, if Y is supported on strings of length k, then $\mathsf{H}_\infty(X \mid Y) \geq \mathsf{H}_\infty(X) - k$.*

Two Source Extractor. We say that a function $\mathsf{Ext}\colon \mathcal{B}^n \times \mathcal{B}^n \to \mathcal{B}$ is a (k,ϵ)-*two-source extractor* if given any two independent distributions X and Y with min-entropy k (each), then the distribution $\mathsf{Ext}(X,Y)$ is ϵ-close to the uniform distribution on one bit [CG88].[1]

It is known that every (k,ϵ)-two-source extractor gives a $2^n \times 2^n$ Boolean matrix in which every minor of size at least $2^k \times 2^k$ has roughly the same number of 1's and 0's, namely, it has $1/2 \pm \epsilon$ fraction of 1's and 0's (and vice versa).

The probabilistic method shows that most functions are two-source extractors with very good parameters (in particular, they work for min-entropy $\log n + 2\log(1/\epsilon) + 1$), but obtaining explicit constructions for such functions has been a major open problem for a long time. In the last couple of years there has been remarkable progress [Coh16a, CZ16, BDT16, Coh16b, Li16] and nearly optimal constructions are now known.

We will actually use the first construction of a two-source extractor given by Chor and Goldreich [CG88, Theorem 9]. They showed that the inner product function (also known as a Hadamard matrix) acts as a good two-source extractor for k which is roughly $n/2$:

Proposition 3. *Let $k = k(n)$ and $\epsilon = \epsilon(n)$ be such that $2k \geq n + 2\log(1/\epsilon) + 2$. Then, the inner-product function is a (k,ϵ)-two-source extractor.*

In other words, the $2^n \times 2^n$ inner-products matrix has the property that every minor of size at least $2^k \times 2^k$ has $1/2 \pm \epsilon$ fraction of 1's and 0's.

2.1.3 List-recoverable codes

The classical notion of error correcting codes ensures that for a code $C \subseteq \mathbb{F}^n$, where $\mathbb{F}$ is a finite field, given a somewhat corrupted version of $c \in C$, it is possible to recover c. The model of allowed corruptions is that some fraction of the symbols in the codeword might be adversarially changed. List recoverable codes were introduced to handle a different model of corruptions: they allow an adversary to submit, for every coordinate $i \in [n]$ a small list $S_i \subseteq \mathbb{F}$ of possible symbols. In this model, it is impossible to completely recover a codeword given the lists, but these codes guarantee that there is only a small list of codewords that are consistent with all the lists.

More precisely, a mapping $C\colon \mathbb{F}^k \to \mathbb{F}^n$ from length k messages to length n codewords, is called (α,ℓ,L)-*list-recoverable* if there is a procedure that is given a sequence of lists $S_1,\ldots,S_n \subseteq \mathbb{F}$ each of size ℓ, and is able to output all messages $x \in \mathbb{F}^k$ such that $C(x)_i \notin S_i$ for at most an α fraction of the coordinates $i \in [n]$. The code guarantees that there are at most L such messages.

Definition 4 (List-recoverable codes)**.** *Let $\alpha \in [0,1]$. We say that a tuple $x \in (\{0,1\}^k)^n$ is* α-consistent *with sets $S_1,\ldots,S_n \subseteq \{0,1\}^k$, if $|\{i : x_i \in S_i|\} \geq \alpha n$.*

[1] We only discuss and define extractors that output one bit since it is enough for our purposes.

A function $C\colon \{0,1\}^v \to (\{0,1\}^k)^n$ *is* (α, ℓ, L)*-list recoverable, if for every set* $S_1, \ldots, S_n \subseteq \{0,1\}^k$ *each of size at most* ℓ*, there are at most* L *strings* $x \in \{0,1\}^v$ *such that* $C(x)$ *is* α*-consistent with* $S_1, \ldots, S_n$*. For* $\alpha = 1$*, we omit* α *in the above notation and call* C (ℓ, L)*-list recoverable. The strings in the image of* C *are referred to as codewords.*

These code were initially studied in the context of *list-decoding* (and indeed the latter is just a special case of the former with $\ell = 1$) by [GS99, GI02, GI03, GI04]. More recently, they were proven useful in other areas such as compressed sensing [NPR12], non-adaptive domain extension for hashing [HIOS15], domain extension for public random functions and MACs [MT07, DS11], and more (see Section 6.5, and for example, [HW15] and references therein).

A natural relaxation of the above codes is to require that $S_1 = \ldots = S_n$. This variant is called *weakly* list-recoverable codes. A list-recoverable code is immediately weakly list-recoverable and the converse also holds albeit with a minor loss in parameters: An (ℓ, L)-weakly list-recoverable code is an (ℓ, nL)-list-recoverable code. Looking ahead, this loss will not make much of a difference for us since our L will be polynomial in n.

For our purposes, we will need a list-recoverable code with $\alpha = 1$. It is well-known (see e.g., [HIOS15]) that the notion of weakly list-recoverable codes is equivalent to unbalanced expanders with a certain expansion property. The left set of vertices in the graph is $\{0,1\}^v$, the right set of vertices is $\{0,1\}^k$ and the left degree is n. This graph naturally induces a mapping $C\colon \{0,1\}^v \to (\{0,1\}^k)^n$ which on input $x \in \{0,1\}^v$ (left vertex) outputs n neighbors (right vertices). The mapping C is (ℓ, L)-list-recoverable iff for every set $S \subseteq \{0,1\}^k$ of size larger than L of nodes on the right, the set of left neighbors of S is of size larger than ℓ.

The following instantiation of locally-recoverable codes based on the explicit construction of unbalanced expanders of [GUV09] is taken (with minor modifications) from [HIOS15].

Theorem 1 ([GUV09, HIOS15])**.** *For every* $\alpha \geq 1/2$*, and* $k < v$*, there exists a* $\mathsf{poly}(n)$*-time computable function* $C\colon \{0,1\}^v \to \left(\{0,1\}^k\right)^n$ *for* $n = O(v \cdot k)^2$ *which defines an* (α, ℓ, L)*-list recoverable code for every* $L \leq 2^{k/2}$ *and* $\ell = \Omega(L)$*. The list-recovery algorithm runs in time* $\mathsf{poly}(v, \ell)$*.*

2.1.4 Nisan-Wigderson type pseudorandom generators

We define Nisan-Wigderson type pseudorandom generators [NW94] that fool circuits of a given size.

Definition 5 (NW-type PRGs.)**.** *A function* $\mathsf{G}\colon \mathcal{B}^{d(n)} \to \mathcal{B}^n$ *is an* NW-type PRG against circuits of size $t(n)$ *if it is (i) computable in time* $2^{O(d(n))}$ *and (ii) any circuit* C *of size at most* $t(n)$ *distinguishes* $U \leftarrow \mathcal{B}^n$ *from* $\mathsf{G}(s)$*, where* $s \leftarrow \mathcal{B}^{d(n)}$*, with advantage at most* $1/t(n)$*.*

Theorem 2 ([IW97]). *Assume there exists a function in $E = DTIME(2^{O(n)})$ with circuit complexity $2^{\Omega(n)}$. Then, for any polynomial $t(\cdot)$, there exists a NW-type generator $G\colon \mathcal{B}^{d(n)} \to \mathcal{B}^n$ against circuits of size $t(n)$, where $d(n) = O(\log n)$.*

Note that one can find a specific function f satisfying the above condition. In general, any function that is E-complete under linear-time reductions will suffice, and in particular, one can take the bounded halting function.[2]

The above theorem was used in derandomization to fool polynomial sized circuits. It was observed in [AIKS16] that Theorem 2 can be extended to more powerful circuits such as non-uniform circuits. In particular, they gave the following theorem that is used in this work.

Definition 6 (oracle circuits and Σ_i/Π_i-circuits). *Given a boolean function $f(\cdot)$, an f-circuit is a circuit that is allowed to use f gates (in addition to the standard gates). A Σ_i-circuit (resp., Π_i-circuit) is an f-circuit where f is the canonical Σ_i^p-complete (resp., Π_i^p-complete) language. The size of all circuits is the total number of wires and gates (see [AB09, Chapter 5] for a formal definition of the classes Σ_i^p and Π_i^p).*

Theorem 3 (cf., [AIKS16, Theorem 1.7]). *For every $i \geq 0$, the statement of Theorem 2 also holds if we replace every occurrence of the word "circuits" by "Σ_i-circuits" or alternatively by "Π_i-circuits".*

The assumption underlying the above theorem is a worst-case assumption and it can be seen as a natural generalization of the assumption that $E \not\subseteq NP$. For a further discussion about this type of assumptions see [AIKS16, AASY16].

2.2 Cryptographic Tools

A function f, with input length $m_1(n)$ and outputs length $m_2(n)$, specifies for every $n \in \mathbb{N}$ a function $f_n\colon \{0,1\}^{m_1(n)} \to \{0,1\}^{m_2(n)}$. We only consider functions with polynomial input lengths (in n) and occasionally abuse notation and write $f(x)$ rather than $f_n(x)$ for simplicity. The function f is computable in polynomial time (efficiently computable) if there exists an algorithm that for any $x \in \{0,1\}^{m_1(n)}$ outputs $f_n(x)$ and runs in time polynomial in n.

A function family ensemble is an infinite set of function families, whose elements (families) are indexed by the set of integers. Let $\mathcal{F} = \{\mathcal{F}_n\colon \mathcal{D}_n \to \mathcal{R}_n\}_{n\in\mathbb{N}}$ stand for an ensemble of function families, where each $f \in \mathcal{F}_n$ has domain $\mathcal{D}_n$ and range $\mathcal{R}_n$. An efficient function family ensemble is one that has an efficient sampling and evaluation algorithms.

Definition 7 (Efficient function family ensemble). *A function family ensemble $\mathcal{F} = \{\mathcal{F}_n\colon \mathcal{D}_n \to \mathcal{R}_n\}_{n\in\mathbb{N}}$ is efficient if:*

[2] The function is defined as follows: $BH(M, x, t) = 1$ if the Turing machine M outputs 1 on input x after at most t steps (where t is given in binary), and $BH(M, x, t) = 0$ otherwise.

- *$\mathcal{F}$ is samplable in polynomial time: there exists a probabilistic polynomial-time machine that given 1^n, outputs (the description of) a uniform element in $\mathcal{F}_n$.*
- *There exists a deterministic algorithm that given $x \in \mathcal{D}_n$ and (a description of) $f \in \mathcal{F}_n$, runs in time $\mathsf{poly}(n, |x|)$ and outputs $f(x)$.*

Definition 8 (Computational Indistinguishability)**.** *Two sequences of random variables $X = \{X_n\}_{n\in\mathbb{N}}$ and $Y = \{Y_n\}_{n\in\mathbb{N}}$ such that X_n's and Y_n's lengths are polynomially bounded are computationally indistinguishable if for every probabilistic polynomial time algorithm A there exists an integer N such that for all $n \geq N$,*

$$|\Pr[A(X_n) = 1] - \Pr[A(Y_n) = 1]| \leq \mathsf{negl}(n) ,$$

where the probabilities are over X_n, Y_n and the internal randomness of A.

Definition 9 (One-Way Functions)**.** *A polynomial time computable function $f : \{0,1\}^* \to \{0,1\}^*$ is called* one-way *if for any PPT inverter A (the non-uniform version is polysize circuit) there exists a negligible function $\mu(\cdot)$, such that*

$$\Pr\left[A(f(x)) \in f^{-1}(f(x)) : x \leftarrow \mathcal{B}^n\right] \leq \mu(n) .$$

We say that $\mathcal{F}$ is a family of injective one-way functions if every function in $\mathcal{F}$ is injective.

2.2.1 Universal one-wayness

A one-way function is an efficiently computable function which is hard to invert on a random output for any probabilistic polynomial-time machine. A universal one-way hash function (UOWHF) is a family of compressing functions $\mathcal{H}$ for which any PPT adversary has a negligible chance of winning in the following game: the adversary submits an x and gets back a uniformly chosen $h \leftarrow \mathcal{H}$. The adversary wins if it finds an $x' \neq x$ such that $h(x) = h(x')$. UOWHF were introduced by Naor and Yung [NY89] and were shown to imply secure digital signature schemes. Rompel [Rom90] (see also [KK05]) showed how to construct UOWHF based on the minimal assumption that one-way functions exist.

Definition 10 (Universal one-way hash functions (UOWHF))**.** *An efficient function family ensemble $\mathcal{F} = \{\mathcal{F}_n : \{0,1\}^{m_1(n)} \to \{0,1\}^{m_2(n)}\}_{n\in\mathbb{N}}$ is a universal one-way hash function family if the probability of every probabilistic polynomial-time adversary $\mathcal{A}$ to win in the following game is negligible in n:*

1. *$\mathcal{A}$, given 1^n, submits $x \in \{0,1\}^{m_1(n)}$.*
2. *Challenger responds with a uniformly random $f \leftarrow \mathcal{F}_n$.*
3. *$\mathcal{A}$ (given f) outputs $x' \in \{0,1\}^{m_1(n)}$.*
4. *$\mathcal{A}$ wins iff $x \neq x'$ and $f(x) = f(x')$.*

2.2.2 Lossy functions and collision resistant hash functions

Collision resistant hash. Recall that a family of collision resistant hash (CRH) functions is one such that it is hard to find two inputs that hash to the same output. More formally, a sequence of families of functions $\mathcal{H}_n = \{h\colon \mathcal{B}^{\ell_1(n)} \to \mathcal{B}^{\ell_2(n)}\}$, where ℓ_1 and ℓ_2 are two functions such that $\ell_1(n) > \ell_2(n)$ for every $n \in \mathbb{N}$, is collision resistant if for every probabilistic polynomial-time algorithms A, there exists a negligible function $\mathsf{negl}(\cdot)$ such that

$$\Pr_{h \leftarrow \mathcal{H}_n}[(x, x') \leftarrow A(1^n, h);\ h(x) = h(x')] \leq \mathsf{negl}(n).$$

CRH functions are known to exist under a variety of hardness assumptions such as factoring, discrete-log, and Learning with Errors (LWE). They are *not* known to exist under the assumption that one-way functions exist[3], and there are oracle separation results for the two primitives [Sim98].

By default, unless we say otherwise, when we assume the existence of CRH functions, then we assume a family as above in which every function shrinks its input by one bit. It is known that such an assumption is equivalent to a family in which every function shrinks by any fixed polynomial factor (by iteratively applying the hash polynomially-many times).

Lossy functions. A collection of lossy functions consists of two families of functions. Functions in the first family are injective, whereas functions in the second family are lossy, namely the size of their image is significantly smaller than the size of their domain. The security requirement is that a description of a randomly chosen function from the first family is computationally indistinguishable from a description of a randomly chosen function from the second family.

Lossy functions were introduced by Peikart and Waters [PW11] and shown to be useful for a variety of fundamental cryptographic applications. In particular, they were shown to imply collision resistant hash functions, oblivious transfer protocols, and chosen ciphertext-secure cryptosystems. Since their introduction they have found numerous other applications (see [FGK+13] for references).

Definition 11 ([PW11])**.** *A collection of (n, ℓ)-lossy functions is defined by a pair of algorithms (G, F) such that:*

1. *$G(1^n, b)$, where $b \in \mathcal{B}$, outputs a string $s \in \mathcal{B}^{p(n)}$ for some fixed polynomial $p(\cdot)$. If $b = 0$, then the algorithm $F(s, \cdot)$ computes an injective function $f_s(\cdot)$ over $\mathcal{B}^n$, and if $b = 1$, then the algorithm $F(s, \cdot)$ computes a function $f_s(\cdot)$ over $\mathcal{B}^n$ whose image size is at most $2^{n-\ell}$.*

[3] In contrast, UOWHFs, Universal One-Way Hash Functions, where there is a fixed target x and the goal is to find x' that collides with it are known to exist under the assumption that one-way functions exist.

2. *The distribution of* $G(1^n, 0)$ *is computationally indistinguishable from the distribution of* $G(1^n, 1)$.

Lossy functions are known to exist under a variety of hardness assumptions such as Decisional Diffie-Hellman (DDH), Learning with Errors (LWE), and factoring related assumptions (Quadratic Residuosity and Phi-hiding) with different parameters [PW11, KOS10, HO12, FGK+13]. In our constructions, we will rely on lossy functions with polynomial shrinkage (e.g., $(n, n - n^{0.1})$-lossy functions). Such functions are known to exist based on LWE [PW11], DDH [FGK+13] and Phi-hiding assumptions [KOS10] (but not based on Quadratic Residuosity). The construction of [KOS10] gives a family of functions which are length-preserving.

2.2.3 Commitment schemes

A commitment scheme is a two-stage interactive protocol between a sender $\mathcal{S}$ and a receiver $\mathcal{R}$. The goal of such a scheme is that after the first stage of the protocol, called the commit protocol, the sender is bound to at most one value. In the second stage, called the opening protocol, the sender opens its committed value to the receiver. We also require that the opening protocol allows to open only a single bit of the committed string. More precisely, a commitment scheme for a domain of strings $\{0,1\}^\ell$ is defined via a pair of probabilistic polynomial-time algorithms $(\mathcal{S}, \mathcal{R}, \mathcal{V})$ such that:

- The commit protocol: $\mathcal{S}$ receives as input the security parameter 1^n and a string $s \in \{0,1\}^\ell$. $\mathcal{R}$ receives as input the security parameter 1^n. At the end of this stage, $\mathcal{S}$ outputs $\mathsf{decom}_1 \ldots, \mathsf{decom}_\ell$ (the local decommitments) and $\mathcal{R}$ outputs com (the commitment).

- The local-opening procedure: $\mathcal{V}$ receives as input the security parameter 1^n, a commitment com, an index $i \in [\ell]$, a local-decommitment decom_i, and outputs either a bit b or $\perp$.

A commitment scheme is *public coin* if all messages sent by the receiver are independent random coins.

Denote by $(\mathsf{decom}_1, \ldots, \mathsf{decom}_\ell, \mathsf{com}) \leftarrow \langle \mathcal{S}(1^n, s), \mathcal{R} \rangle$ the experiment in which $\mathcal{S}$ and $\mathcal{R}$ interact with the given inputs and uniformly random coins, and eventually $\mathcal{S}$ outputs a list of ℓ decommitment strings and $\mathcal{R}$ outputs a commitment. The completeness of the protocol says that for all $n \in \mathbb{N}$, every string $s \in \{0,1\}^\ell$, every tuple $(\mathsf{decom}_1, \ldots, \mathsf{decom}_\ell, \mathsf{com})$ in the support of $\langle \mathcal{S}(1^n, s), \mathcal{R} \rangle$, and every $i \in [\ell]$, it holds that $\mathcal{V}(i, \mathsf{decom}_i, \mathsf{com}) = s_i$.

Below we define two security properties one can require from a commitment scheme. The properties we list are *statistical-hiding* and *computational-binding*. These roughly say that after the commit stage, the sender is *bound* to a specific value which remains statistically hidden for the receiver.

Definition 12 (ϵ-binding). *A commitment scheme $(\mathcal{S}, \mathcal{R}, \mathcal{V})$ is $(t(n), \epsilon(n))$-binding if for every probabilistic adversary $\mathcal{S}^*$ that runs in time at most $t(n)$, it holds that*

$$\Pr\left[\begin{array}{c}(i, \mathsf{decom}_i, \mathsf{decom}'_i, \mathsf{com}) \leftarrow \langle \mathcal{S}^*(1^n), \mathcal{R}\rangle \text{ and} \\ \perp \neq \mathcal{V}(i, \mathsf{decom}_i, \mathsf{com}) \neq \mathcal{V}(i, \mathsf{decom}'_i, \mathsf{com}) \neq \perp\end{array}\right] \leq \epsilon(n)$$

for all sufficiently large n, where the probability is taken over the random coins of both $\mathcal{S}^*$ and $\mathcal{R}$.

Given a scheme $(\mathcal{S}, \mathcal{R}, \mathcal{V})$ and an adversary $\mathcal{R}^*$, we denote by $\mathsf{view}_{\langle \mathcal{S}(s), \mathcal{R}^*\rangle}(n)$ the distribution on the view of $\mathcal{R}^*$ when interacting with $\mathcal{S}(1^n, s)$. The view consists of $\mathcal{R}^*$'s random coins and the sequence of messages it received from $\mathcal{S}$. The distribution is take over the random coins of both $\mathcal{S}$ and $\mathcal{R}$. Without loss of generality, whenever $\mathcal{R}^*$ has no computational restrictions, we can assume it is deterministic.

Definition 13 (ρ-hiding). *A commitment scheme $(\mathcal{S}, \mathcal{R}, \mathcal{V})$ is $\rho(n)$-hiding if for every (deterministic) adversary $\mathcal{R}^*$ and every distinct $s_0, s_1 \in \{0,1\}^\ell$, it holds that*

$$\Delta\left(\{\mathsf{view}_{\langle \mathcal{S}(s_0), \mathcal{R}^*\rangle}(n)\}, \{\mathsf{view}_{\langle \mathcal{S}(s_1), \mathcal{R}^*\rangle}(n)\}\right) \leq \rho(n)$$

for all sufficiently large $n \in \mathbb{N}$.

Complexity measures. The parameters of interest are (1) the number of rounds the commit protocol requires, (2) the size of a commitment, and (3) the size of a local opening.

The *size* of a commitment is the size (in bits) of the output of $\mathcal{S}$ denoted above by com. A *short commitment* is such that the size of com is much smaller than ℓ. Preferably, the size of a short commitment depends solely on n, but poly-logarithmic dependence on ℓ is also okay. The size of a local opening is the maximum size of decom_i (in bits). A protocol is said to support local opening if this size depends only on n and at most poly-logarithmically on ℓ.

2.3 Fully black-box constructions

We give a definition of a fully black-box reduction from an MCRH to standard CRH. For this, we generalize the definition of an MCRH to the setting of oracle-aided computation: The generation and evaluation algorithms of an MCRH are given access to an oracle Γ relative to which they can generate a description of a hash function and evaluate an index at a point. The adversary is also given oracle access to Γ in the security game and has to find multiple collisions relative to it.

We focus here on k-MCRH functions with $k = 3$. The following definition of a "black-box construction" is directly inspired by those of [AS16, HHRS15].

Definition 14. *A fully black-box construction of a collision-resistant function family $\mathcal{H}'$ from a 3-MCRH function family $\mathcal{H}$ mapping $2n$ bits to n bits consists of a pair of probabilistic polynomial-time algorithms $(\mathcal{H}.\mathsf{G}, \mathcal{H}.\mathsf{E})$ and an oracle-aided polynomial-time algorithm M such that:*

- **Completeness:** *For any $n \in \mathbb{N}$, for any 3-MCRH function family $\mathcal{H}$ and any function h produced by $h \leftarrow \mathcal{H}'^{\mathcal{H}}(1^n)$, it holds that $h^{\mathcal{H}} \colon \{0,1\}^{2n} \to \{0,1\}^n$.*
- **Black-box proof of security:** *For any collision resistant hash $\mathcal{H}'$, any probabilistic polynomial-time oracle-aided algorithm $\mathcal{A}$, every polynomial $p(\cdot)$, if*

$$\Pr\left[\begin{array}{c} x_1 \neq x_2 \\ h^{\mathcal{H}}(x_1) = h^{\mathcal{H}}(x_2) \end{array} \middle| \begin{array}{c} h \leftarrow h^{\mathcal{H}}(1^n) \\ (x_1, x_2) \leftarrow \mathcal{A}^{\mathcal{H}}(1^n, h) \end{array}\right] \geq \frac{1}{p(n)}$$

for infinitely many values of n, then there exists a polynomial $p'(\cdot)$ such that

$$\Pr\left[\begin{array}{c} x_1, x_2, x_3 \text{ are distinct and} \\ h(x_1) = h(x_2) = h(x_3) \end{array} \middle| \begin{array}{c} h \leftarrow \mathcal{H}(1^n) \\ (x_1, x_2, x_3) \leftarrow M^{\mathcal{A},\mathcal{H}}(1^n, h) \end{array}\right] \geq \frac{1}{p'(n)}$$

for infinitely many values of n.

2.3.1 Witness indistinguishable proof systems

We consider witness indistinguishable proof systems for NP. In particular, these are derandomized versions of Zaps and can be constructed from any Zap by fixing the first message non-uniformly (see [DN07, Section 3]). In the uniform setting, Barak et al. [BOV07] showed that by leveraging a NW-type PRG, they can derandomize the Zap construction and get the same result. In both cases, we get a witness indistinguishable proof system which is defined as follows:

Definition 15. *Let L be an NP language with relation R. A scheme $(\mathsf{Prove}, \mathsf{Verify})$ is a witness indistinguishable proof system between a verifier and a prover if:*

1. **Completeness***: for every $(x, w) \in R$ we have that:*

$$\Pr\left[\mathsf{Verify}(x, \pi) = 1 \mid \pi \leftarrow \mathsf{Prove}(x, w)\right] = 1 \,.$$

2. **Perfect Soundness***: for every $x \notin L$ and for every π it holds that:*

$$\Pr\left[\mathsf{Verify}(x, \pi) = 1\right] = 0 \,.$$

3. **Witness Indistinguishability***: for any sequence of $\{x, w, w'\}$ such that $(x, w) \in R$ and $(x, w') \in R$ it holds that:*

$$\{\mathsf{Prove}(x, w)\} \approx_c \{\mathsf{Prove}(x, w')\} \,.$$

2.4 Average-case complexity

For a language L and an instance x we write $L(x) = 1$ if $x \in L$ and $L(x) = 0$ otherwise.

Definition 16 (Probability distributions)**.** *A probabilistic randomized algorithm $\mathcal{D}$ is said to be a probability distribution if on input 1^n it outputs a string of length n. We denote by $\mathcal{D}(1^n; r)$ the evaluation of $\mathcal{D}$ on input 1^n using the random coins r. We say that $\mathcal{D}$ is efficient if it runs in polynomial time.*

Definition 17 (Hard distributional problem)**.** *Let $L \in$ NP and let $\mathcal{D}$ be an efficient probability distribution. We say that $(L, \mathcal{D})$ is a hard distributional problem if for any probabilistic polynomial time-algorithm A there exist a negligible function $\mathsf{negl}(\cdot)$ such that for all large enough n it holds that:*

$$\Pr_{r,A}[A(x) = L(x) : x \leftarrow \mathcal{D}(1^n; r)] \leq 1/2 + \mathsf{negl}(n) \ ,$$

where the probability is taken over r and the randomness of A.

Remark 1. *We say that a language $L \in$ NP is hard-on-average if there exists an efficient probability distribution $\mathcal{D}$ such that $(L, \mathcal{D})$ is a hard distributional problem.*

Definition 18 (Hard public-coin distributional problem)**.** *Let $L \in$ NP and let $\mathcal{D}$ be an efficient probability distribution. We say that $(L, \mathcal{D})$ is a hard public-coin distributional problem if for any probabilistic polynomial time-algorithm A there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all large enough n it holds that:*

$$\Pr_{r,A}[A(r, x) = L(x) : x \leftarrow \mathcal{D}(1^n; r)] \leq 1/2 + \mathsf{negl}(n) \ ,$$

where the probability is taken over r and the randomness of A. (Notice that in this case, A gets both the instance x and the random coins r used to sample x.)

2.5 Search Problems

The class TFNP of "total search problems" contains a host of non-trivial problems for which a solution always exists.

Definition 19 (TFNP)**.** *A* total NP search problem *is a relation $\mathcal{S}(x, y)$ such that it is (i) computable in polynomial (in $|x|$ and $|y|$) time (ii)* total*, i.e. there is a polynomial q such that for every x there exists a y such that $\mathcal{S}(x, y)$ and $|y| \leq q(|x|)$. The set of all total NP search problems is denoted by TFNP.*

The class TFNP/poly is the non-uniform circuit version of TFNP, similar to NP/poly with respect to NP.

Definition 20 (TFNP/poly)**.** *A* total NP/poly search problem *is a relation* $\mathcal{S}(x,y)$ *such that it is (i) computable polynomial (in* $|x|$ *and* $|y|$*) time with polynomial advice or equivalently there exists a family of polynomial sized circuits that computes* $\mathcal{S}$ *(ii)* total*, i.e. there is a polynomial* q *such that for every* x *there exists a* y *such that* $\mathcal{S}(x,y)$ *and* $|y| \leq q(|x|)$*. The set of all total NP/poly search problems is denoted by TFNP/poly.*

A *polynomial-time reduction* from total search problem $\mathcal{S}$ to total search problem $\mathcal{T}$ is a pair f, g of polynomial-time computable functions such that, for every input x of $\mathcal{S}$, $f(x)$ is an input of $\mathcal{T}$, and furthermore for every $y \in \mathcal{T}_{f(x)}$, $g(y) \in \mathcal{S}_x$.

Since TFNP is a "semantic class", it is unlikely to contain any natural complete problems. Papadimitriou [Pap94] defined various "syntactic" subclasses of TFNP with important complete problems based on combinatorial principles used to show their totality.

The class PPAD (for Polynomial Parity Arguments on Directed graphs) is defined as all the total search problems reducible to the End-of-Line problem.

Definition 21 (End-of-Line)**.** *Given two circuits* $\mathsf{S}, \mathsf{P} \colon \mathcal{B}^n \to \mathcal{B}^n$*, such that* $\mathsf{P}(0^n) = 0^n \neq \mathsf{S}(0^n)$*, find a string* $x \in \mathcal{B}^n$ *such that* $\mathsf{P}(\mathsf{S}(x)) \neq x$ *or* $\mathsf{S}(\mathsf{P}(x)) \neq x \neq 0^n$*.*

The main appeal of the class PPAD is that it has many important complete problems related to algorithmic game theory, such as finding a Nash equilibrium in bimatrix games [DGP09, CDT09] or in constant degree graphical games [Rub15]. The black-box hardness of problems related to PPAD was extensively studied, and we point the interested reader to [SS06, CD08, FISV09] and references within for additional results.

Johnson, Papadimitriou and Yannakakis [JPY88] defined and studied a subclass of TFNP that captures the complexity of local search problems. The class PLS (for Polynomial Local Search) is defined as all the total search problems reducible to the Local-Search problem. In the following we denote by $[n]$ the set of numbers $\{1, 2, \ldots, n\}$:

Definition 22 (Local-Search)**.** *Given two circuits* $\mathsf{S} \colon \mathcal{B}^n \to \mathcal{B}^n$ *and* $\mathsf{V} \colon \mathcal{B}^n \to [2^n] \cup \{0\}$*, find a string* $x \in \mathcal{B}^n$ *such that* $\mathsf{V}(\mathsf{S}(x)) \leq \mathsf{V}(x)$*.*

Note that the above definition is equivalent to an alternative definition where the problem is given by circuits $\mathsf{N} \colon \mathcal{B}^n \to \mathcal{B}^{p(n)}$ and $\mathsf{V} \colon \mathcal{B}^n \to [2^n] \cup \{0\}$ such that for each vertex $x \in \mathcal{B}^n$, the circuit N outputs a polynomial number $p(n)$ of x's neighbors. First, every instance (S, V) can be viewed as an instance (N, V) of the second type with a neighborhood function $\mathsf{N} = \mathsf{S}$ outputting only singleton neighborhoods. Second, any instance (N, V) can be transformed into an equivalent "hill-climbing" instance (S, V) by making the circuit S output a neighbor with the highest value under V.

Additional black-box hardness results for local search can be found in [SS09, SY09]. The relation between the query complexity of local search and approximate fixed point computation was studied by Chen and Teng [CT07].

Continuous Local Search. Even though the LOCAL-SEARCH problem and also the END-OF-LINE problem that define PLS and PPAD respectively are discrete and presented in terms of Boolean circuits, both classes have equivalent definitions in terms of continuous functions over the unit cube. The notion of continuity used in this context is the *Lipschitz continuity*.

Definition 23. *A function $f\colon \mathbb{R}^n \to \mathbb{R}^m$ is λ-Lipschitz for some $\lambda > 0$ if for all $x, x' \in \mathbb{R}^n$ it holds that $|f(x) - f(x')|_\infty \leq \lambda \cdot |x - x'|_\infty$, where $|\cdot|_\infty$ denotes the maximum norm.*

The continuous complete problem for PPAD is finding a Brouwer fixed point. An instance is given by an $\varepsilon > 0$ and a λ-Lipschitz function f mapping the unit cube to itself. The goal is to find an ε-approximate fixed point of f, i.e., some $x \in [0,1]^3$ such that $|f(x) - x| < \varepsilon$. The continuous complete problem for PLS is finding a local optimum in a continuous valuation function. An instance is given by a function f mapping the unit cube to itself (not necessarily Lipschitz) and a λ-Lipschitz function p assigning a real value between zero and one to every point of the unit cube. The goal is to find an ε-approximate local optimum of p under f, i.e., a point $x \in [0,1]^3$ such that $p(f(x)) - p(x) < \varepsilon$.

Daskalakis and Papadimitriou [DP11] suggested to combine the above two definitions in a synergetic way and defined the class CLS (for Continuous Local Search) to contain all the total search problems reducible to the following CONTINUOUS-LOCAL-OPTIMUM problem, where both the transition and the valuation functions are λ-Lipshitz.

Definition 24 (CONTINUOUS-LOCAL-OPTIMUM)**.** *Given two arithmetic circuits $f\colon [0,1]^3 \to [0,1]^3$ and $p\colon [0,1]^3 \to [0,1]$, and two real constants $\varepsilon, \lambda > 0$, find either a point $x \in [0,1]^3$ such that $p(f(x)) \leq p(x) + \varepsilon$ or a pair of points $x, x' \in [0,1]^3$ certifying that either p or f is not λ-Lipschitz.*

We already discussed that the class CLS lies in the intersection of PLS and PPAD and that it contains a multitude of important computational problems at the outskirts of P. We note, however, that there is no known *combinatorial* complete problem for CLS and finding one remains an interesting open problem.

2.5.1 Black-box and white-box models

Let $\mathcal{F}_n = \{f\colon \mathcal{B}^n \to \mathcal{B}^n\}$ be the class of all circuits f mapping n bits into n bits. We give a definition of a search problem for the family $\mathcal{F}_n$.[4]

Definition 25. *A* search problem *$\mathcal{S}$ is a relation on $2q(n)$ tuples. More precisely, $\mathcal{S} = \cup_{n=1}^{\infty} \mathcal{S}_n$, where $\mathcal{S}_n \subseteq (\mathcal{B}^n)^{q(n)} \times (\mathcal{B}^n)^{q(n)}$ for a polynomial $q(\cdot)$, such that: (i) for all $f \in \mathcal{F}_n$, there exist $x_1, \ldots, x_{q(n)} \in \mathcal{B}^n$ for which $\big(x_1, \ldots, x_{q(n)}, f(x_1), \ldots, f(x_{q(n)})\big) \in \mathcal{S}$, and (ii) $\mathcal{S}$ is computable in polynomial time in n. The class of all such search problems is denoted TFNP.*

[4] We restrict our attention to the family $\mathcal{F}_n$ of length-preserving functions for simplicity.

The tuple $(x_1, \ldots, x_{q(n)})$ is called the witness (i.e., the solution). In general, a witness is not necessarily given as a sequence of points in the domain $\mathcal{B}^n$ but notice that any string can be encoded as such a sequence and so our definition is without loss of generality.

We mainly focus on three models of computation that differ either by the representation type of the function $f \in \mathcal{F}_n$ or by the complexity measure of the solver. The models that we define and study are the *black-box* model, the *white-box* model, and a new hybrid model we call *succinct black-box*. We also mention a fourth model we call the *efficient-succinct black-box* model. For the rest of this subsection, fix a polynomial $q = q(n)$ and a search problem $\mathcal{S} \subseteq (\mathcal{B}^n)^q \times (\mathcal{B}^n)^q$.

In the *black-box model*, an algorithm is required to solve the search problem $\mathcal{S}$ while given only oracle access to the function f. That is, the algorithm provides queries x and gets back the results $y = f(x)$. The black-box complexity of a search problem $\mathcal{S}$ is the number queries needed to solve a search problem in the worst-case, while the running time is unbounded. This model was introduced and studied by Lovász et al. [LNNW95].

Definition 26 (Black-box complexity)**.** *The black-box complexity of $\mathcal{S}$, denoted by* $\mathsf{bbc}(\mathcal{S})$*, is bounded by a function $T(\cdot)$ if there exists an algorithm A that for sufficiently large n and any $f \in \mathcal{F}_n$, makes at most $T(n)$ queries to f and outputs $x_1, \ldots, x_q$ such that $(x_1, \ldots, x_q, f(x_1), \ldots, f(x_q)) \in \mathcal{S}$.*

In the *white-box model*, an algorithm is required to solve the search problem $\mathcal{S}$ while given an explicit representation of the function f (as a circuit). The white-box complexity of $\mathcal{S}$ is the *running time* (as opposed to number of queries) needed (measured as a function of the size of the representation) to solve a search problem in the worst-case. In the white-box setting, we are mostly interested in solvers that run in polynomial-time in the size of the function.

Definition 27 (White-box complexity)**.** *The white-box complexity of $\mathcal{S}$, denoted by* $\mathsf{wbc}(\mathcal{S})$*, is bounded by a function $T(\cdot)$ if there exists an algorithm A that for sufficiently large n, given $f \in \mathcal{F}_n$ (as a circuit) runs in time $T(|f|)$, and outputs $x_1, \ldots, x_q$ such that $(x_1, \ldots, x_q, f(x_1), \ldots, f(x_q)) \in \mathcal{S}$.*

In the *succinct black-box model*, an algorithm is required to solve the search problem $\mathcal{S}$ while given only oracle access to the function f, however, as opposed to the black-box model, the succinct black-box complexity of a search problem $\mathcal{S}$ is measured by the number of queries required to solve the problem as a function of the *size of the representation* of f. In particular, if f is represented succinctly by a polynomial-size (in n) circuit, then an efficient algorithm can perform only a polynomial number of queries (but its running time is unbounded).

Definition 28 (Succinct black-box complexity)**.** *The succinct black-box complexity of $\mathcal{S}$, denoted by* $\mathsf{sbbc}(\mathcal{S})$*, is bounded by the function $T(\cdot)$ if there exists an algorithm A that for sufficiently large n and any $f \in \mathcal{F}_n$, makes at most $T(|f|)$ queries to f and outputs $x_1, \ldots, x_q$ such that $(x_1, \ldots, x_q, f(x_1), \ldots, f(x_q)) \in \mathcal{S}$.*

We also consider a model we call the *efficient-succinct black-box model*, which is similar to the succinct black-box model, except that the solver's running is bounded (in the representation size). In Table 2.1 below we summarize the differences between the models.

Model	Function Access	Solver Running Time	Rep. Size
Black-Box (BB)	Oracle	Unbounded	Unbounded
White-Box	Implicit	Bounded	Bounded
Succinct Black-Box	Oracle	Unbounded	Bounded
Efficient-Succinct BB	Oracle	Bounded	Bounded

Table 2.1: A summary of the different models defined.

2.6 Ramsey theory

In this section we recall some basic definitions and facts from Ramsey theory and derive several bounds that will be useful for us later. We refer to Graham et al. [GRS90] for a thorough introduction and history of Ramsey theory.

A Ramsey graph is a graph that contains no clique or independent set of some predefined sizes.

Definition 29 (Ramsey graphs)**.** *A graph on N vertices is called (s,t)-Ramsey if it contains no independent set of size s and no clique of size t. A graph is called k-Ramsey if it is (k,k)-Ramsey.*

The classical result of Ramsey gives an upper bound on the size of a graph that does not contain either an independent set or a clique of some predefined size.

Proposition 4. *For any $s,t > 1$, there exists a number $R(s,t) < \infty$ such that any graph on $R(s,t)$ vertices is not (s,t)-Ramsey. Moreover,*

$$R(s,t) \leq R(s-1,t) + R(s,t-1) \leq \binom{s+t-2}{s-1}.$$

Plugging in $s = t = (\log N)/2$, we get that

$$R\left((\log N)/2, (\log N)/2\right) \leq \binom{\log N}{(\log N)/2} \leq 2^{\log N} = N,$$

where the inequality follows by the inequality $\binom{2k}{k} \leq 2^{2k}$. As a corollary of Proposition 4, we get:

Proposition 5. *Every graph on N vertices has either a clique or an independent set of size $\frac{1}{2}\log N$.*

A well-known (*non-explicit*) construction of a Ramsey graph was given by Erdös [Erd47] as one of the first applications of the probabilistic method. He showed that most graphs on N vertices are $(2\log N)$-Ramsey (see also the book of Alon and Spencer [AS08]). It was observed by Naor [Nao92] that Erdös's proof actually gives a stronger statement: not only are most graphs $(2\log N)$-Ramsey, but such graphs can actually be sampled with relatively few bits of randomness (i.e., via a limited-independent family[5] or a small-bias probability space [NN93]). For completeness, the proof of the next statement is given in Section 5.7.3. No explicit construction of graphs matching these parameters is known, but see Section 5.2.2 for the state of the art. For a function $g\colon \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ we define the corresponding graph G on n vertices where for any $u < v$ (lexicographic order) it holds that (u,v) is an edge in G iff $g(u,v) = 1$.

Proposition 6. *A graph on N vertices sampled via a $(2\log^2 N)$-wise independent hash function is a $(2\log N)$-Ramsey graph with probability $1 - 1/N^{\Omega(\log\log N)}$.*

Given that there are constructions of k-wise independent functions mapping $\mathcal{B}^n \times \mathcal{B}^n \to \mathcal{B}$ that are succinct (the size of the representation is polynomial in n and k even for n output bits), the proposition implies that it is possible to sample a Ramsey graph (w.h.p.) with a succinct representation, i.e., the description length of the graph is polynomial in n. Furthermore, since computing a $(2\log^2 N)$-wise independent function can be done in time proportional to the size of the description, it is possible to sample a circuit that implicitly represents the graph.

The property of a graph being (s,t)-Ramsey can be equivalently phrased as a coloring property of the complete graph K_N on N vertices with two colors. Specifically, the function that defines whether there is an edge or not can be thought of a coloring of the full graph with two colors and the existence of a clique or an independent set of size k is equivalent to the existence of a monochromatic subgraph of size k. This raises a natural generalization of the Ramsey property for graphs with multiple colors.

Definition 30 (Colorful Ramsey graphs)**.** *A coloring $\psi\colon \binom{N}{2} \to [m]$ of the full graph K_N with m colors is called $(k_1,\ldots,k_m)$-Ramsey if there is no monochromatic subgraph of size k_i colored with the color i, for every $i \in [m]$.*

The colorful Ramsey theorem provides, for a given number of colors, an upper bound on the size of a clique such that any coloring must result with a monochromatic subgraph of a predefined size.

Proposition 7. *For any m and $k_1,\ldots,k_m > 1$, there exists a number $R(k_1,\ldots,k_m) < \infty$ such that any graph on $R(k_1,\ldots,k_m)$ vertices is* not *$(k_1,\ldots,k_m)$-Ramsey. Moreover,*

$$R(k_1,\ldots,k_m) \leq 2 + \sum_{i=1}^{m} \left(R(k_1,\ldots,k_{i-1},k_i-1,k_{i+1},\ldots,k_m) - 1\right).$$

[5] A function family $\mathcal{H} = \{h\colon \mathcal{D} \to \mathcal{R}\}$ is k-wise independent, if $\Pr_{h\leftarrow\mathcal{H}}[h(x_1) = y_1 \vee h(x_2) = y_2 \vee \ldots \vee h(x_k) = y_k] = 1/|\mathcal{R}|^k$, for every distinct $x_1,x_2,\ldots,x_k \in \mathcal{D}$ and every $y_1,y_2,\ldots,y_k \in \mathcal{R}$.

Based on Proposition 7, we can upper bound $R(k_1, \ldots, k_m)$ for various values of $k_1, \ldots, k_m$. In particular, in the symmetric case where $k_1 = k_2 \ldots k_{m-1} = k_m$, we get:

Proposition 8. *For every $k > 2$ and $m > 1$, it holds that $R(\underbrace{k, \ldots, k}_{m \text{ times}}) \leq m^{mk}$.*

As a corollary of Proposition 8, we obtain a bound on the number of colors that ensure the existence of a monochromatic subgraph of size k.

Proposition 9. *Consider the full graph on N vertices. For every $k < \log N$, and every coloring $\psi \colon \binom{N}{2} \to [m]$, where $m = \frac{(\log N)/k}{\log \log N - \log k}$, there exists a monochromatic subgraph of size k.*

The proofs of Propositions 8 and 9 appear in Section 5.7.2.

Chapter 3

The Journey from NP to TFNP Hardness

3.1 Introduction

Previously, the hardness of TFNP was only based either on Obfustopia or strong versions of Minicrypt (e.g. one-way permutations). None of the assumptions used to show TFNP hardness are known to be implied simply by the existence of one-way functions, or even from public-key encryption. It is known that one-way permutations cannot be constructed in a black-box way from one-way functions [Rud89, KSS11]. Moreover, indistinguishability obfuscation is a relatively new notion that has yet to find solid ground (see [AJN+16, Appendix A] for a summary of known attacks as of August 2016). Moreover, while indistinguishability obfuscation implies the existence of one-way functions [KMN+14], it has been shown that it cannot be used to construct either one-way permutations or collision-resistant hash in a black-box manner [AS15, AS16]. Other hardness results for TFNP rely on specific number theoretic assumptions such as factoring or discrete log.

Barriers for proving TFNP hardness. Given the lack of success in establishing hardness of TFNP under general assumptions, there has been some effort in trying to explain this phenomenon. From the early papers on PLS and TFNP by Johnson et al. [JPY88] and Megiddo and Papadimitriou [MP91] we know that TFNP hardness cannot be based on worst-case NP hardness, unless NP = coNP.[1] Mahmoody and Xiao [MX10] showed that even a *randomized* reduction from TFNP to worse-case NP would imply that SAT is "checkable" (which is a major open question [BK95]). Buhrman et al. [BFK+10] exhibited an oracle under which all TFNP problems are easy but the polynomial hierarchy is infinite, thus, explaining the failure to prove

[1] The argument is quite simple: Suppose there is a reduction from SAT (for example) to TFNP, in a way that every solution to the TFNP problem translates to a resolution of the SAT instance. This would yield an efficient way to refute SAT: guess a solution of the TFNP problem, and verify that it indeed implies that the SAT instance has no solution.

TFNP hardness based on worst-case problems in the polynomial hierarchy. In a recent work, Rosen et al. [RSS16] showed that any attempt to base TFNP hardness on (trapdoor) one-way functions (in a black-box manner) must result in a problem with exponentially many solutions. This leads us to ask the following natural question:

What is the weakest assumption under which we can show hardness of TFNP?

A possible approach to answering this question is to try to put TFNP hardness in the context of the Impagliazzo's five worlds. In the light of this classification, one can argue that an equally, if not more interesting, question is:

What is the weakest assumption for hardness on average of TFNP?

In this work, we give an (almost) tight answer to this question. Given the negative results discussed above, our results are quite unexpected. While the barriers imply that it is very unlikely to show TFNP hardness under *worst-case* NP hardness, we are able to show that *hard-on-average* TFNP can be based on any *hard-on-average* language in NP (and in particular on the existence of one-way functions). In the terminology of the worlds of Impagliazzo, our results show that hard-on-average TFNP problems exist in `Pessiland` (and beyond), a world in which none of the assumptions previously used to show TFNP hardness exist (not even one-way functions). On the other hand, the barriers discussed above indicate that it would be unlikely to prove worse-case TFNP hardness in `Heuristica`, as in that world the only available assumption is $\mathsf{P} \neq \mathsf{NP}$.

As for techniques, we show an interesting interplay between TFNP and derandomization. We show how to "push" problems into TFNP while maintaining their hardness using derandomization techniques. In the non-uniform settings, our results can be established with no further assumptions. Alternatively, we show how to get (standard) uniform hardness via further derandomization assuming the existence of a Nisan-Wigderson type pseudorandom generator (discussed below).

In correspondence with the black-box impossibility of Rosen et al. [RSS16], the problem we construct might have instances with an exponential number of solutions. Nevertheless, we show that there exists a different problem in TFNP such that its hardness can be based on the existence of one-way functions and *Zaps* (discussed below), and has *either one or two* solutions for any instance. The reason our result does not contradict the impossibility results is (i) that it is not black-box, and (ii) we use an extra object, Zaps; it is unknown whether the latter exists solely under the assumption that one-way functions exist (but it can be shown to exist relative to random oracles). We observe that the fact that our problem has either one or two solutions for every instance is (in some sense) tight: any hard problem with a single solution for every instance would imply hardness of $\mathsf{NP} \cap \mathsf{coNP}$ (by taking a hardcore bit of the unique solution), and thus would have to face the limitations of showing $\mathsf{NP} \cap \mathsf{coNP}$ hardness from unstructured assumptions [BDV16].

Nisan-Wigderson PRGs. Nisan and Wigderson showed how the assumption of a very hard language can be used to construct a specific type of pseudorandom generators (henceforth NW-type PRG) benefitial for tasks of derandomization and in particular to derandomize BPP algorithms [NW94]. Impagliazzo and Wigderson [IW97] constructed a NW-type PRG under the (relatively modest) assumption that E (i.e., $\mathsf{DTIME}(2^{O(n)})$) has a problem of circuit complexity $2^{\Omega(n)}$. Although used mainly in computational complexity, (strong versions of) these generators have found applications in cryptography as well: Barak, Lindell, and Vadhan [BLV06] used them to prove an impossibility for two-round zero-knowledge proof systems, Barak, Ong, and Vadhan [BOV07] showed how to use them to construct a witness indistinguishable proof system for NP, and Bitansky and Vaikuntanathan [BV15] showed how to completely immunize a large class of cryptographic algorithms from making errors. In the examples above, the PRG was constructed to fool polynomial sized (co)non-deterministic circuits. Such NW-type PRGs follow from (relatively modest) assumption that E has a problem of (co)non-deterministic circuit complexity $2^{\Omega(n)}$.

We show that NW-type PRGs have an interesting interplay with TFNP as well. We use strong versions of these generators to show that several different problems can be "pushed" into TFNP by eliminating instances with no solutions (in the uniform setting). Our notion of strong NW-type PRG requires fooling polynomial-sized Π_2-circuits. Again, such PRGs follow from the assumption that E has a problem of Π_2-circuit complexity $2^{\Omega(n)}$ (see [AIKS16] for an example of a use of such PRGs).

Zaps. Feige and Shamir [FS90] suggested a relaxed notion of zero-knowledge called *witness indistinguishability* where the requirement is that any two proofs generated using two different witnesses are computationally indistinguishable. They showed how to construct three-message witness indistinguishable proofs for any language in NP assuming the existence of one-way functions. A Zap, as defined by Dwork and Naor [DN07], is a two-message public-coin witness indistinguishable scheme where the first message can be reused for all instances. They showed that (assuming one-way functions) Zaps are existentially equivalent to NIZKs (non-interactive zero-knowledge proofs), and hence one can use the known constructions (e.g., based on trapdoor permutations). Dwork and Naor also showed that the interaction could be further reduced to a single message witness indistinguishable proof system in the non-uniform setting (i.e., the protocol has some polynomial sized advice). In the uniform setting, Barak et al. [BOV07] showed the same result by leveraging a NW-type PRG for derandomizing the known constructions of Zaps. Our proofs make use of such witness indistinguishable proof systems both in the uniform and non-uniform setting.

3.2 Our results

Some of our results use a derandomization assumption in the form "assume that there exists a function with deterministic (uniform) time complexity $2^{O(n)}$, and Π_2-circuit complexity $2^{\Omega(n)}$". In the description of our results below, we simply call this the *fooling assumption.* Alternatively, instead of this assumption we can consider the non-uniform setting, and get the same results for TFNP/poly (see Definition 20). Some of our results are summarized in Figure 4.1.

Theorem 4 (Informal)**.** *Any hard-on-average NP language (e.g., random SAT, hidden clique, or any one-way function) implies a non-uniform TFNP problem which is hard-on-average.*

Then, using derandomization we get the following corollary for the uniform setting.

Corollary 1 (Informal)**.** *Under the fooling assumption, any hard-on-average NP language implies a (uniform) TFNP problem which is hard-on-average.*[2]

Furthermore, we present an alternative approach for proving totality of search problems using zero-knowledge proofs which allows to build more structured TFNP problems. Specifically, if injective one-way functions exist, and Zaps exist then we construct a total search problem with at most two solutions.

Theorem 5 (Informal)**.** *Assume the existence of Zaps and injective one-way functions. Then, there exists a hard-on-average problem (either non-uniform, or uniform under the fooling assumption) in TFNP such that any instance has at most two solutions.*

3.3 Related work (search vs. decision)

The question of the complexity of search when the solution is guaranteed to exist has been discussed in various scenarios. Even et al. [ESY84] considered the complexity of promise problems and the connection to cryptography. They raised a conjecture which implies that public-key cryptography based on NP-hard problems does not exist. Impagliazzo and Naor [IN88] and Lovasz et al. [LNNW95] considered search in the query complexity model where a solution is guaranteed to exist, similarly to the class TFNP. They showed a separation between the classes of deterministic, randomized, and the size of the object of the search. Bellare and Goldwasser [BG94] considered the issue of self reducibility, i.e. are there languages in NP where given a decision oracle it is hard to solve *any* corresponding FNP search problem. They showed that such languages exist under the assumption that EE $\neq$ NEE (double exponential time is not equal to its non-deterministic version).

[2] By combining this corollary with the result of [MX10] we get a new impossibility for basing average-case NP hardness on worst-case NP hardness. In particular, any such reduction (even adaptive and randomized) would imply that SAT is checkable (under the "fooling" assumption). Previously, [MX10] showed this result with respect to *non-uniform* instance checkers (however, without the "fooling" assumption).

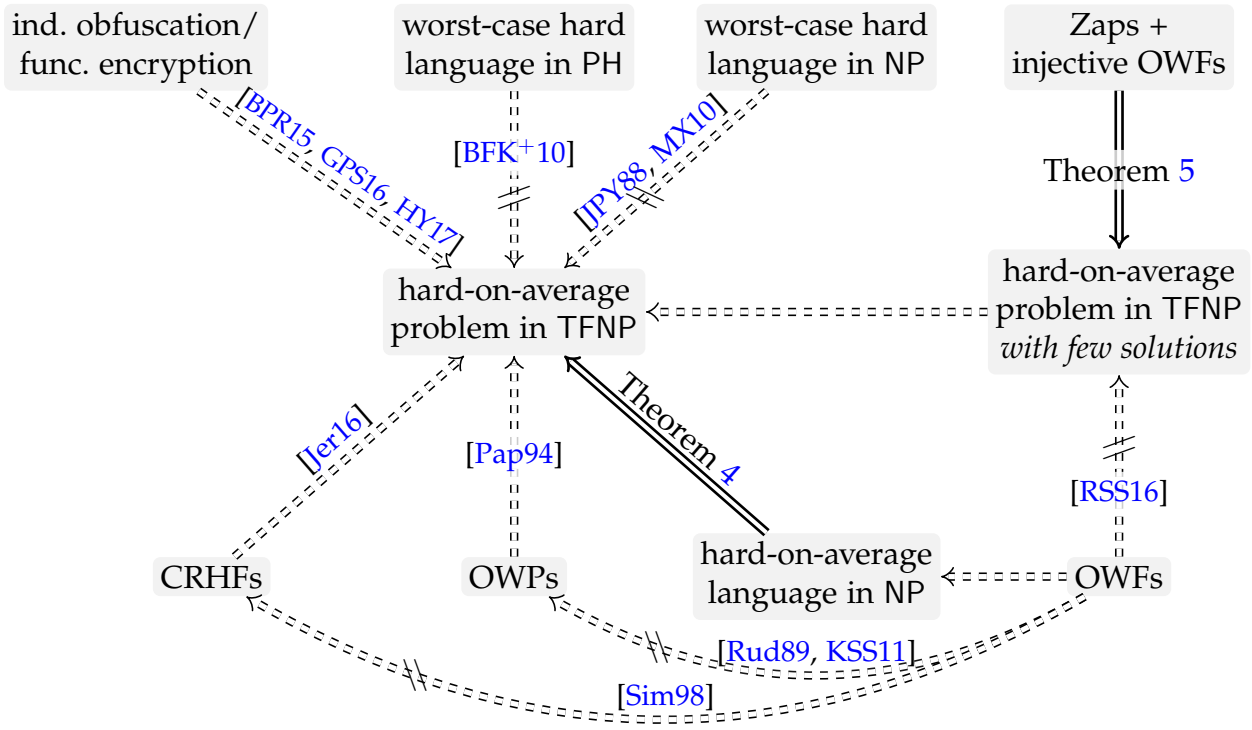


Figure 3.1: An illustration of our results (solid implications) in the context of previously known positive results (dashed implications) and negative results (crossed out dashed implications).

3.4 Technical Overview

3.4.1 TFNP hardness based on average-case NP hardness

Let L be a hard-on-average NP language for an efficiently samplable distribution $\mathcal{D}$, associated with a relation R_L. That is, $(L, \mathcal{D})$ is a pair such that no efficient algorithm can decide L with high probability when instances are drawn from $\mathcal{D}$. Our goal is to construct a (randomized) reduction from an instance for L sampled from $\mathcal{D}$ to a TFNP problem, such that every solution to the TFNP problem can be used to decide the instance given for L. As discussed above, such a reduction cannot rely on worst-case complexity ([JPY88, MP91]), and hence it must use the fact that we have a hard distribution for L.

The first thing to note is that since $(L, \mathcal{D})$ is hard to decide, then it is also hard to find a valid witness. Thus, the corresponding search problem, i.e., given an instance x to find a witness w such that $(x, w) \in R_L$, is hard directly based on the hardness of $(L, \mathcal{D})$. However, this is not sufficient for establishing hardness in TFNP. The issue is that not all instances indeed have a solution, and furthermore, determining whether an instance has a solution is by itself NP-hard.

To show that a problem is in TFNP we need to prove that there exists a solution for *every* instance. Therefore, our goal is to start with $(L, \mathcal{D})$ and end up with $(L', \mathcal{D}')$, such that the distribution $\mathcal{D}'$ always outputs instances in L', but remains a hard distribution nevertheless. We employ a method called *reverse randomization*, which has been used in the context of complexity (for proving that BPP is in the polynomial hierarchy [Lau83]) and in the context of cryptography (for constructing Zaps, commitment schemes, and immunizing cryptographic schemes from errors [Nao91, DN07, DNR04, BV15]). For a string s, let the distribution $\mathcal{D}_s$ be the one that on random coins r samples both $x \leftarrow \mathcal{D}(1^n; r)$ and $x' \leftarrow \mathcal{D}(1^n; r \oplus s)$. We define L_s accordingly as containing the instance (x, x') if *at least one* of x or x' is in L. Since D is a hard distribution, we know that for x' sampled from D the probability that $x' \in L$ is non-negligible. Therefore, we get that for any instance $x \notin L$ with non-negligible probability over the choice of s the shifted version x' of x is such that $x' \in L$. We perform several such random shifts, such that for any $x \notin L$ the probability that *one* of its shifts is an element in L is very high, and define the new language to accept any instance where *at least one of the shifts* is in L. Moreover, we show that for any such collection of shifts, the resulting distribution is still hard even when the shift is given to the solving algorithm.

The result is that there *exists* a collection of shifts such that applying them to $\mathcal{D}$ yields a pair $(L', \mathcal{D}')$ with the following two properties: (1) the support of $\mathcal{D}'$ is entirely contained in L', and (2) the pair $(L', \mathcal{D}')$ is a hard *search* problem, i.e., given a random instance x sampled from $\mathcal{D}'$ the probability of finding a valid witness for x in polynomial-time is negligible. To construct our hard TFNP problem there are a few difficulties we ought to address.

We have merely proven the existence of such a shift for every input length. However, finding such a shift might be a hard task on its own. One possible way to circumvent this issue is to consider non-uniform versions of the problem, where this shift is hardwired into the problem's description (for each input size). Notice that in this case the shift is public and thus it is important that we have proven hardness even given the shift. If we want to stay in the uniform setting, we need to show how to find such shifts efficiently for every n. The main observation here is that we can show that a random shift will be good enough with high probability, and verifying if a given shift is good can be done in the complexity class Π_2. Thus, using a Nisan-Wigderson type pseudorandom generators against Π_2-circuits we show how to (completely) derandomize this process and find the collection of shifts efficiently and deterministically. Such NW-type PRGs were shown to exist under the assumption that E is hard for Π_2 circuits.

At this point, we have an (efficiently) samplable distribution $\mathcal{D}'$ such that its support provably contains only instances in L', and finding any valid witness is still hard for any polynomial-time algorithm. However, how can we use $\mathcal{D}'$ to create a hard TFNP problem? Indeed, we can use $(L', \mathcal{D}')$ to construct a hard search problem. Since any instance in the support of $\mathcal{D}'$ is satisfiable we would claim that the problem is indeed in TFNP. However, this is actually not the case, since it is infeasible to verify

that an instance has actually been sampled from $\mathcal{D}'$! Therefore, the resulting search problem is not in TFNP. To solve this, we need a way to prove that an instance x is in the support of $\mathcal{D}'$ without hurting the hardness of the distribution.

On distributions with public randomness. The straightforward solution to the above problem is to publish the randomness r used to sample the instance. This way, it is easy to verify that the instance is indeed in the support of $\mathcal{D}'$ and thus any instance must have a solution and our problem is in TFNP. But what about hardness? Is the distribution hard even when given the randomness used to sample from it? Note that in many cases the randomness might even explicitly contain a satisfying assignment (e.g., a planted clique or factorization of a number). Thus, we must rely on distributions which remain hard even when given the random coins used for sampling. We denote such distributions as *public-coin distributions*.

If the original distribution $\mathcal{D}$ was public-coin to begin with, then we show that our modifications maintain this property. Thus, assuming that $\mathcal{D}$ is public-coin we get a problem that is provably in TFNP and is as hard as $\mathcal{D}$ (see Section 3.6 for the full proof). Then, we show that in fact any hard distribution can be modified to construct a public-coin distribution while maintaining its hardness, with no additional assumptions. This lets us construct a hard TFNP problem using any hard-on-average language L, even one with a distribution that is not public-coin (see Section 3.7 for discussion of the transformation).

The number of solutions of the TFNP problem we constructed is polynomial in the number of witnesses of the language L we begin with (each shift introduces new solutions). In particular, if we start from a hard decision problem $(L, \mathcal{D})$ where $\mathcal{D}$ is public-coin and every $x \in L$ has a single witness, then our TFNP problem will have only a polynomial number of witnesses. Our transformation from any distribution to a public-coin one increases the number of witnesses (for some instances). In the next section, we discuss an alternative method for proving totality of search problems which, moreover, results in a small number of solutions.

3.4.2 Using zero-knowledge for TFNP hardness

We demonstrate the power of zero-knowledge proofs in the context of TFNP in order to assure totality of search problems. This enables us to get more structured problems in TFNP.

Suppose we have a one-way function f, and we try to construct a hard TFNP problem where any solution can be used to invert f on a random challenge y. The difficulty is that we need to prove that the search problem is total while not ruining its hardness. In particular, we have to prove that given y there exists an inverse x such that $f(x) = y$ without "revealing anything" beyond the existence of x. If f is a permutation this task is trivial, since every y has an inverse by definition. However, for a general one-way function this is not the case.

To solve this issue, we employ as our main tool a Zap: a two-message witness indistinguishable proof system for NP (as discussed above). We construct a problem where an instance contains an image y and a Zap proof π of the statement "y has an inverse under f". The first message of the proof is either non-uniformly hard-wired in the problem description, or uniformly derandomized (see discussion in the introduction). This way, any instance has a solution: if π is a valid proof then by the perfect soundness of the Zap we know that y indeed has an inverse, and otherwise, we consider the all-zero string to be a solution. Our goal is to show that this problem is still hard. However, the Zap proof only guarantees that for any two x, x' the proof π is indistinguishable, which does not suffice. In fact, if f is injective, then y has only a single inverse and π might simply be the inverse x without violating the witness indistinguishability property of the Zap.

Therefore, an instance of our problem will consist of two images y, y' and a Zap proof π that *one* of the images has an inverse under f. The goal is to find an inverse of y or y', where one is guaranteed to exist as long as the proof π is valid. This way, we are able to ensure that the proof π does not reveal any useful information about the inverse x. Finally, by randomly embedding an instance y of a challenge to f we show that any adversary that solves this problem can be used to invert the one-way function f. Thus, we get a total problem that is hard assuming one-way functions, and Zaps. Moreover, notice that when the underlying one-way function is *injective* the problem we constructed has *exactly one or two solutions* for any instance. See Theorem 8 and Section 3.8 for details of the full proof.

3.5 TFNP Hardness from Average-Case NP Hardness

We show that there exists a search problem in TFNP that is hard-on-average under the assumption of existence of a hard-on-average NP language and a Nisan-Wigderson type complexity assumption. An overview of the proof is given in Section 3.4.1. Formally, we prove:

Theorem 6. *If there exists a hard-on-average language in NP then there exists a hard-on-average problem in non-uniform TFNP.*

Under an additional (worst-case) complexity assumption (as discussed in Section 3.2), we also give a uniform version of this result.

Corollary 2. *Assume that there exist functions with deterministic (uniform) time complexity $2^{O(n)}$, and Π_2-circuit complexity $2^{\Omega(n)}$. If there exists a hard-on-average language in NP then there exists a hard-on-average problem in TFNP.*

We split the proof of Theorem 6 and Corollary 2 into two parts. First, we prove the results under the assumption that the distribution $\mathcal{D}$ has a special property we call *public-coin*, i.e., when the distribution remains hard even given the random coins used to sample from it (see Definition 18). Second, we show that this assumption does not

hurt the generality of the statement, since the existence of hard distributional decision problems implies the existence of hard *public-coin* distributional decision problems.

3.6 Hardness based on public-coin distributions

We begin with the proof of the non-uniform version of our result.

Proof (Theorem 6). Fix an input size n. For simplicity of presentation (and without loss of generality), assume that to sample an instance $x \in \mathcal{B}^n$ the number of random coins needed is n (otherwise, one can pad the instances to get the same effect). We begin by showing that any hard *public-coin* distributional decision problem $(L, \mathcal{D})$ implies the existence of a hard *public-coin* distributional search problem. In general, this problem will not be a total one but we will be able to use its structure to devise a total problem.

Let $\mathcal{D}$ be a hard public-coin distribution for some NP language L with associated relation R_L. The distributional search problem associated to $(L, \mathcal{D})$, i.e., given $x \leftarrow \mathcal{D}(1^n)$ to find a w such that $R_L(x, w) = 1$, is also hard. Moreover, there exists a polynomial $p(\cdot)$ such that $\Pr_{x \leftarrow \mathcal{D}}[L(x) = 1] \geq p(n)$, as otherwise, the "constant no" algorithm would contradict $\mathcal{D}$ being a hard distribution for L. For any set of $k = n^2 \cdot p(n)$ strings $s_1, \ldots, s_k \in \mathcal{B}^n$, we define a distributional problem $(L_{s_1,\ldots,s_k}, \mathcal{D}')$, where the language $L_{s_1,\ldots,s_k}$ is defined by the relation

$$R_{L_{s_1,\ldots,s_k}}(r, w) = \bigvee_{i \in [k]} R_L(x_i, w), \text{ where } x_i \leftarrow \mathcal{D}(1^n; r \oplus s_i) \ ,$$

and $\mathcal{D}'$ is the uniform distribution on $\mathcal{B}^n$.

First, we use the following general lemma to argue that the distributional search problem associated with $(L_{s_1,\ldots,s_k}, \mathcal{D}')$ is hard.

Lemma 1. *Let $(L, \mathcal{D})$ be a hard public-coin distributional search problem. Let $(L', \mathcal{D}')$ be a distributional search problem related to $(L, \mathcal{D})$ that satisfies the following conditions:*

1. *L' is in an "OR" form, i.e., there exist efficiently computable functions $f_1, \ldots, f_k$ such that $R_{L'}(x', w) = \bigvee_{i \in [k]} R_L(f_i(x'), w)$ where k is some polynomially bounded function of n.*

2. *For every $i \in [k]$, the marginal distribution of $f_i(x')$ under $x' \leftarrow \mathcal{D}'$ is identical to the distribution of $x \leftarrow \mathcal{D}(1^n)$.*

3. *For any fixed instance $x^* = \mathcal{D}(1^n; r)$, the distribution $x' \leftarrow \mathcal{D}'(1^n)$ conditioned on $f_i(x') = x^*$ is efficiently sampleable (given r).*

Then $(L', \mathcal{D}')$ is a hard public-coin distributional search problem.

The proof of Lemma 1 follows by a reduction to solving the original distributional search problem $(L, \mathcal{D})$, and is deferred to Section 3.9.1. Notice that $(L_{s_1,\ldots,s_k}, \mathcal{D}')$ satisfies the three conditions of Lemma 1 with respect to $(L, \mathcal{D})$: (1) the instances are

in the "OR" form (where $x_i = f_i(r) = r \oplus s_i$), (2) for any $i \in [k]$ it holds that x_i is distributed as $\mathcal{D}$ since $r \oplus s_i$ is uniformly random, and (3) for any $x^* = \mathcal{D}(1^n; r^*)$, sampling from $\mathcal{D}'$ conditioned on $x_i = x^*$ can be done by setting $r = s_i \oplus r^*$. We say that $L_{s_1,\ldots,s_k}$ *is good* if it is total, i.e., if it holds that

$$\forall r \in \mathcal{B}^n : L_{s_1,\ldots,s_k}(r) = 1 \,.$$

We prove that for a random choice of $s_1, \ldots, s_k$ the language $L_{s_1,\ldots,s_k}$ is good with high probability.

Claim 1. $\Pr_{s_1,\ldots,s_k \leftarrow \mathcal{B}^{kn}}[L_{s_1,\ldots,s_k} \text{ is good }] \geq 3/4$.

Proof. Fix any string r. If we pick s at random we get that

$$\Pr_{s \leftarrow \mathcal{B}^n}[L(x) = 1 \mid x \leftarrow \mathcal{D}(1^n; r \oplus s)] \geq 1/p(n) \,.$$

This follows since for any string r the string $r \oplus s$ is uniformly random. Moreover, since for any s_i this event is independent, we get that for any fixed r (and for $k = n^2 \cdot p(n)$) it holds that:

$$\Pr_{s_1,\ldots,s_k \leftarrow \mathcal{B}^{kn}}[\forall i : L(x_i) = 0 \mid x_i \leftarrow \mathcal{D}(1^n; r \oplus s_i)] \leq (1 - 1/p(n))^k \leq 2^{-n^2} \,.$$

Thus, taking a union bound over all possible $r \in \mathcal{B}^n$ we get that

$$\Pr_{s_1,\ldots,s_k \leftarrow \mathcal{B}^{kn}}[L_{s_1,\ldots,s_k} \text{ is not good }] \leq 2^n \cdot 2^{-n^2} \leq 1/4 \,,$$

and thus the statement of the claim follows. □

Thus, for any n we fix a set of k strings non-uniformly to get a good $L_{s_1,\ldots,s_k}$, and get a language L^*, that is a hard-on-average search problem under the uniform distribution $\mathcal{D}^*$. Moreover, we get that the problem is total: for every n, every instance $r \in \mathcal{B}^n$ must have a solution since we choose a good $L_{s_1,\ldots,s_k}$ for every n.[3] □

Getting a uniform version. In the above proof of Theorem 6 we constructed a problem in non-uniform TFNP, i.e., a problem in TFNP/poly (see Definition 20). To get a uniform version of the problem (i.e., a problem in TFNP) we show how to employ derandomization techniques to find $s_1, \ldots, s_k$.

Proof (Corollary 2). Recall the definition of the language $L_{s_1,\ldots,s_k}$ from the proof of Theorem 6. Consider a circuit C that on input $s_1, \ldots, s_k$ outputs 1 if and only if $L_{s_1,\ldots,s_k}$ is good. Notice that C can be implemented by a polynomial-sized Π_2-circuit, since

[3] Actually, this claim works only for large enough input sizes n. For small values of n we simply define the language to accept all instances, and thus to remain total. Notice that this does not harm the hardness of the problem.

$s_1, \ldots, s_k$ is good if for all $r \in \mathcal{B}^n$ there exists a witness (s_i, w) such that $R_L(r \oplus s_i, w) = 1$. Let G be a Nisan-Wigderson type PRG (see Definition 5) against Π_2-circuits of size $|C|$ with seed length $m = O(\log n)$. For any seed $j \in [2^m]$, let $(s_1^j, \ldots, s_k^j) = \mathsf{G}(j)$ and write $L^j = L_{s_1^j \ldots, s_k^j}$. By Claim 1 and by pseudorandomness of G we get that

$$\Pr_{j \in [2^m]}[L^j \text{ is good}] \geq 3/4 - 1/|C| \geq 2/3 \,.$$

To derandomize the choice of $s_1, \ldots, s_k$ we define

$$L^*(r_1, \ldots, r_{2^m}) = \bigvee_{j \in [2^m]} L^j(r_j) \,,$$

where every r_j is of length n. Accordingly, define $\mathcal{D}^*$ to sample $\bar{r} = r_1, \ldots, r_{2^m}$ uniformly at random where $r_j \in \mathcal{B}^n$. Notice that $(L^*, \mathcal{D}^*)$ satisfies the following:

1. $\Pr_{\bar{r} \leftarrow \mathcal{D}^*}[L^*(\bar{r}) = 1] = 1$.
2. For any PPT A for large enough n we have: $\Pr_{\bar{r} \leftarrow \mathcal{D}^*}[L^*(\bar{r}, A(\bar{r})) = 1] \leq \mathsf{negl}(n)$.

The first item follows by the derandomization. The second item follows, since for all $j \in [2^m]$, the search problem $(L^j, \mathcal{D}')$ is hard (which follows from Lemma 1). In particular, any adversary A solving the search problem associated to $(L^*, \mathcal{D}^*)$ with noticeable probability $1/p(n)$ must solve one of the $(L^j, \mathcal{D}')$ with probability at least $1/(2^m p(n)) = 1/\mathsf{poly}(n)$, a contradiction. We get that $(L^*, \mathcal{D}^*)$ is a hard-on-average search problem, which is total, and thus in TFNP. □

Note that it was crucial that $\mathcal{D}$ was a public-coin distribution in order to construct $\mathcal{D}^*$ to be the uniform while maintaining hardness. This, in turn, let us define a total problem since any string is in the support of $\mathcal{D}^*$. In the next section, we show that the assumption that $\mathcal{D}$ is public-coin can be made without loss of generality.

3.7 From private coins to public coins

We prove that we can assume that our underlying distribution is public-coin without loss of generality. That is, we show that it can be (efficiently) converted to a public-coin distribution with same hardness with no additional assumptions.

Theorem 7. *If hard-on-average NP languages exist then public-coin hard-on-average NP languages exist.*

Let $\mathcal{D}$ be a distribution of a hard-on-average NP language. We observe that if $\mathcal{D}$ is the *uniform* distribution then it is easy to get a public-coin version of it. Indeed, while $\mathcal{D}$ itself might not be public-coin, by directly sampling instances uniformly from the range of $\mathcal{D}$ (instead of from the domain) we get a public-coin hard distribution.

Impagliazzo anad Levin [IL90] showed how to transform any hard-on-average NP problem to a different one that is hard under the uniform distribution. An alternative presentation of this result appears in Goldreich [Gol08, Sec. 10.2.2.2] (see also Exercise 10.14). Another proof via the method of inaccessible entropy was given by Haitner et al. [HHR+10, Lemma 43].

We provide a self-contained proof that does not rely on the above notions. Our approach is to use the construction of universal one-way hash functions (UOWHFs) from one-way functions [NY89, Rom90, KK05]. A UOWHF is a weaker primitive than a collision resistant hash function, where an adversary chooses an input x, then it is given a hash function h sampled from the UOWHF family, and its task is to find an input $x' \neq x$ such that $h(x) = h(x')$. Therefore, any UOWHF gives rise to a hard distributional search problem (in fact, a distributional decision problem by [BCGL92]) which is public-coin: given random sample x, h find a colliding x'. We conclude the proof by showing that for any input length, a hard distribution is either already public-coin or it gives rise to a one-way function for this length. See Section 3.9.2 for the complete proof.

3.8 Using Zero-Knowledge for TFNP Hardness

In the previous section, we have shown how to get TFNP hardness from average-case NP hardness. We either settled for a non-uniform version with no additional assumptions or used a NW-type PRG to get a uniform version. In this section we show a different approach to proving hardness of problems in TFNP. Our main technique uses Zap, a two-message witness indistinguishable proof for NP. As discussed in Section 3.5, the Zap can be further reduced to a single message by either fixing the first message non-uniformly or by using a NW-type PRG. In both cases, we get non-interactive witness indistinguishable proof system for NP (see Definition 15), and thus we write the proof in terms of this primitive.

The advantage of this technique is twofold. First, it is a general technique that might be used to "push" other problems inside TFNP. Second, it allows us to construct a hard problem in TFNP from injective one-way functions (see Definition 9) with at most two solutions. Formally, we prove the following theorem:

Theorem 8. *If injective one-way functions and non-interactive witness-indistinguishable proof systems for NP exist, then there exists a hard-on-average problem in TFNP such that any instance has at most two solutions.*

Proof. We define a new total search problem and call it Invert-Either.

Definition 31 (Invert-Either)**.** *Let $f\colon \mathcal{B}^m \to \mathcal{B}^n$ be an efficiently computable function. Let L be an NP language defined by the following relation: $R((y, y'), x) = 1$ if and only if $f(x) \in \{y, y'\}$. Let $(\mathsf{Prove}, \mathsf{Verify})$ be a witness indistinguishable proof system for L. The input to the Invert-Either problem is a tuple (y, y', π), where $y, y' \in \mathcal{B}^n$, and $\pi \in \mathcal{B}^{\mathsf{poly}(n)}$. We ask to find a string $x \in \mathcal{B}^m$ satisfying one of the following:*

1. $x = 0^m$ *if* $\mathsf{Verify}((y, y'), \pi) = 0$.
2. $f(x) \in \{y, y'\}$ *if* $\mathsf{Verify}((y, y'), \pi) = 1$.

We now show that Invert-Either is a total search problem.

Claim 2. *The* Invert-Either *problem is in TFNP.*

Proof. Let (y, y', π) be an instance of Invert-Either. If $\mathsf{Verify}((y, y'), \pi) = 0$ then $x = 0^n$ is a solution. Otherwise if $\mathsf{Verify}((y, y'), \pi) = 1$ then, by the perfect soundness of the witness indistinguishable proof system $(\mathsf{Prove}, \mathsf{Verify})$, it follows that $(y, y') \in L$. Thus, there exists an $x \in \mathcal{B}^n$ such that $f(x) = y$ or $f(x) = y'$ which is a solution for the Invert-Either instance (y, y', π). In either case there exists a solution and Invert-Either is in TFNP. □

We move on to show that the existence of one-way functions implies a hard on average distribution of instances of Invert-Either. Assume that there exists an efficient algorithm A that solves in polynomial time instances of Invert-Either defined relative to some one-way function f. We construct A' that inverts an image of f evaluated on a random input with noticeable probability. Given $y = f(x)$, a challenge for inverting the function f, the inverter A' proceeds as follows:

$\underline{A'(y):}$

1. choose $x' \leftarrow \mathcal{B}^n$ at random and compute $y' = f(x')$.
2. choose $b \leftarrow \mathcal{B}$ at random and set $y_b = y$ and $y_{1-b} = y'$.
3. compute $\pi \leftarrow \mathsf{Prove}((y_0, y_1), x')$.
4. compute $w \leftarrow A(y_0, y_1, \pi)$.
5. output w.

Since A' computes the proof π honestly, any solution w for the Invert-Either instance (y_0, y_1, π) must be a preimage of either y or y', i.e., either $f(w) = y$ or $f(w) = y'$. If A outputs a preimage of y then A' will succeed in inverting f. However, A might output a w which is a preimage of y' which was chosen by A' and it does not help in inverting the challenge y. Our claim is that A must output a preimage of y with roughly the same probability as a preimage of y'. Formally, we show that

$$|\Pr[f(A(y_0, y_1, \pi)) = y'] - \Pr[f(A(y_0, y_1, \pi)) = y]| \leq \mathsf{negl}(n) \ .$$

It is sufficient to argue that the input tuple (y_0, y_1, π) for A produced by A' is computationally indistinguishable from an input triple produced using the actual pre-image x of the challenge y, i.e.,

$$\{y_0, y_1, \pi \leftarrow \mathsf{Prove}((y_0, y_1), x')\} \approx_c \{y_0, y_1, \pi \leftarrow \mathsf{Prove}((y_0, y_1), x)\} \ .$$

Since x and x' are chosen according to the same distribution, and y_0 and y_1 are random labels of y and y', the only way to distinguish between the two ensembles is using the proof π. However, from the witness-indistinguishability property of the proof system we get that $\mathsf{Prove}((y_0, y_1), x)$ is computationally indistinguishable from $\mathsf{Prove}((y_0, y_1), x')$ even given x and x' (and thus also given y_0 and y_1). Altogether, we get that the probability that A outputs a preimage of y is about the same probability as the probability that A outputs a preimage of y'. By our assumption, A must output either type of solution with noticeable probability. Therefore, A' succeeds in inverting the challenge y with noticeable probability. □

Remark 2. *We note that we get hardness of* Invert-Either *from any one-way function, however, the number of solutions is guaranteed to be at most two only when the one-way function is injective.*

3.9 Chapter Appendix

3.9.1 Proof of Lemma 1

We give the full proof of Lemma 1 that we use in Section 3.6. The proof follows by a reduction to the original distributional problem $(L, \mathcal{D})$. Given a challenge x^* for $(L, \mathcal{D})$ we can create an instance σ of the search problem associated with $(L', \mathcal{D}')$ such that the challenge x^* is embedded in a random location among the "ORed" instances in σ. This can be done efficiently by the first and the third item. Then, by the second item we get that if $x^* \in L$ then a solution for σ will contain a solution for x^* with probability at least $1/k$. Overall, we gain a polynomial advantage for solving x^* which contradicts the hardness of $(L, \mathcal{D})$.

Lemma 1. *Let $(L, \mathcal{D})$ be a hard public-coin distributional search problem. Let $(L', \mathcal{D}')$ be a distributional search problem related to $(L, \mathcal{D})$ that satisfies the following conditions:*

1. *L' is in an "OR" form, i.e., there exist efficiently computable functions $f_1, \ldots, f_k$ such that $R_{L'}(x', w) = \bigvee_{i \in [k]} R_L(f_i(x'), w)$ where k is some polynomially bounded function of n.*
2. *For every $i \in [k]$, the marginal distribution of $f_i(x')$ under $x' \leftarrow \mathcal{D}'$ is identical to the distribution of $x \leftarrow \mathcal{D}(1^n)$.*
3. *For any fixed instance $x^* = \mathcal{D}(1^n; r)$, the distribution $x' \leftarrow \mathcal{D}'(1^n)$ conditioned on $f_i(x') = x^*$ is efficiently sampleable (given r).*

Then $(L', \mathcal{D}')$ is a hard public-coin distributional search problem.

Proof. Assume towards contradiction that there exist an adversary A and a polynomial $p(\cdot)$ such that

$$\Pr_{x \leftarrow D'(1^n; r)} [R_{L'}(x, A(x, r)) = 1] \geq 1/p(n) .$$

Then, we construct an adversary A' that solves the search problem $(L, \mathcal{D})$.
$A'(x)$:

1. Choose $i \in [k]$ at random.
2. Sample $x' \leftarrow \mathcal{D}'$ conditioned on $f_i(x') = x$ (this can be performed efficiently due to Item 3).
3. Output $w \leftarrow A(x')$.

Notice that since x is sampled from $\mathcal{D}$ and the marginal distribution of $f_i(x')$ is identical to that of $\mathcal{D}$ (Item 2), we get that x' is distributed exactly as a sample from $\mathcal{D}'$. Therefore, when computing w we have that A sees exactly the distribution it expects, and it will find a valid solution to x' with probability at least $1/p(n)$.

If $x \in L$, then with probability $1/p(n)$ we have that A will give us a solution to x' and since x' is in the "OR" form (Item 1), with probability $1/k$ that solution will solve x (and otherwise it answers at random). Thus, the probability for A' to solve x is at least $1/(k \cdot p(n))$ which contradicts $(L, \mathcal{D})$ being a hard distributional search problem. □

3.9.2 Proof of Theorem 7

We give the proof of the following theorem establishing that private-coin distributional decision problems imply existence of public-coin distributional decision problems.

Theorem 7. *If hard-on-average NP languages exist then public-coin hard-on-average NP languages exist.*

Proof. We begin by showing that if one-way functions exist then there are hard-on-average distributions that are public-coin. This part of the proof follows by combining known transformations. First, it is known that if one-way functions exist then universal one-way hash functions (UOWHFs) exist [NY89, Rom90, KK05]. A UOWHF is a family of compressing functions $\mathcal{H}$ that have the following security requirement: for a random x and $h \in \mathcal{H}$ it is hard for any PPT algorithm to find an $x \neq x'$ such that $h(x) = h(x')$. We note that the constructions of such families are in fact public-coin: to sample h no private coins are used. Thus, we can define the following search problem: given x, h find an appropriate collision x'.[4] Second, we can apply to this search problem a transformation of Ben-David et al. [BCGL92] for converting any average-case search problem into an average-case decision problem. Although it was not mentioned explicitly in their work, we observe when applied to a search problem that has a public-coin distribution the resulting decision problem is public-coin as well.

[4] Notice that this search problem problem is not in TFNP since no such collision x' might exist.

We have shown how to get a hard public-coin distributional problem from one-way function. We want to show how to get the same result from any hard distributional problem. Let $\mathcal{D}$ be a hard-on-average NP distribution for some language L. If $\mathcal{D}$ is public-coin, then we are done. Assume that $\mathcal{D}$ is not public-coin.

If we were able to prove a statement of the form "private-coin distributions imply one-way functions", then by applying the above two transformations we would be done. But what exactly is a "private-coin" distribution? Our only assumption is that $\mathcal{D}$ is not public-coin. It might be the case that for some (infinitely many) input sizes the distribution is public-coin and for some (infinitely many) it is not. Thus, the function that we get will be hard to invert only on the input sizes that the distribution is not public-coin. Then, the distribution that we get from this function will be public-coin only for the same input sizes. However, for the rest of the input sizes, the distribution was already public-coin! Thus, by combining the two we can get a hard public-coin distribution for any input size.

One subtle issue is that we do not necessarily know for which input sizes is the distribution public-coin and for which not. Thus, for any input size n we apply both methods: we two samples using randomness r_1 and r_2. For r_1 we release $r_1, \mathcal{D}(1^n, r_1)$, and we r_2 to sample from the distribution constructed from the one-way function, as described above. Finally, we take the "AND" of the both. For any n we know that one of the two will be hard, and thus overall we get hardness for any input size.

We are left to show how to construct one-way functions that are hard to invert for the non-public input sizes of the distribution. Assume that $\mathcal{D}$ is not public-coin. Then, there exist an efficient algorithm A, and a polynomial p such that for infinitely many input sizes it holds that:

$$\Pr_{r,A}[A(r,x) = L(x) : x \leftarrow D_n(r)] \geq 1/2 + 1/p(n) \ . \tag{3.9.1}$$

Let $\mathcal{N}$ be the infinite set of $n \in \mathbb{N}$ such that Equation (3.9.1) holds. We define the function family $f = \{f_n(r) = D_n(r)\}_{n\in\mathbb{N}}$. We claim that it is infeasible to invert f for the input sizes in $\mathcal{N}$:

Claim 3. *For any PPT A there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $n \in \mathcal{N}$:*

$$\Pr_{r\in\mathcal{B}^n}[A(f_n(r)) \in f_n^{-1}(f_n(r))] \leq \mathsf{negl}(n) \ .$$

Proof. Impagliazzo and Luby [IL89] showed that if one-way functions do not exist, then it is not only possible to invert a function f, but also to get a close to uniform inverse. Formally, if A is an adversary that inverts f then there exists a constant $c > 0$ such that A outputs a distribution that is $1/n^c$-close to a uniform distribution over the inverses.

Thus, suppose that f is not one-way as stated in the claim. Then, given $y = f(r)$ we can run the inverter on y and get r' such that $f(r) = f(r')$ with probability $1/p(n)$ for some polynomial p. Moreover, there exists a constant c such that the distribution of r' is $1/n^c$ close to a uniform one. The high-level idea is that if we run $A(r', x)$ then

we get the correct answer with high probability, thus we are able to decide L relative to $\mathcal{D}$ with high probability.

Formally, we verify that $\mathcal{D}(r) = \mathcal{D}(r')$. If this is not the case then we answer randomly. Assume that $\mathcal{D}(r) = \mathcal{D}(r')$. Let

$$\mathcal{R}_x = \{r : \Pr_A[A(r,x) = L(x)] \geq 1/2 + 1/p(n)\}.$$

We say that x is good if $\Pr_r[r \in \mathcal{R}_x] \geq 1/2$. By Equation (3.9.1) we get that the probability that x is good is at least half. If x is good, then we get that $A(r',x) = L(x)$ with probability at least $1/2 + 1/p(n) - 1/n^c$. Altogether, we get a polynomial advantage above $1/2$ in deciding L. □

The above claim concludes the proof. □

Chapter 4

Hardness of Continuous Local Search: Query Complexity and Cryptographic Lower Bounds

4.1 Introduction

Local search is a widely applicable tool for constructing approximation algorithms when facing NP-hard optimization problems (cf. [WS11]). In local search we seek only for a local optimum rather than insisting on a global one. More formally, an instance of the problem is given by two functions, a neighborhood function that maps each point to a set of its local neighbors and a valuation function that gives each point a real value. The goal is to find a point for which none of its neighbors have a (strictly) greater value. Many of the most popular optimization algorithms apply local search; the simplex algorithm, genetic algorithms, steepest ascent hill climbing, and simulated annealing, to name a few.

Although powerful, local search has its limitations. It is known that there exist instances for which finding a local optimum might take exponential time. A natural approach to avoid the shortcomings of local search is to exploit some additional structure of the optimization problem at hand. In particular, both the neighborhood function and the valuation function are often *continuous*, which might offer advantage when trying to solve such *continuous local search* problems. Motivated by these considerations, we ask the following natural question:

*How hard is **continuous** local search?*

There are two common types of hardness results: white-box and black-box. In the white-box setting, the problem at hand is given explicitly and in a succinct manner (e.g., as a polynomially sized circuit). Then, hardness is shown via a reduction from a different problem for which hardness is already established (or assumed). In the black-box setting (also known as the query model, or the decision tree model) the problem is represented via oracle access, and the complexity is measured by the

number of performed oracle queries. In this model, it is often possible to prove unconditional lower bounds on the number of queries required to find a solution. In this paper, we show hardness of *continuous* local search in terms of *both white-box and black-box*.

Optimization problems amenable to local search, as well as continuous local search, fall into a vast category of problems for which a solution is guaranteed to exist. Indeed, there is a myriad of important natural problems with a straightforward proof certifying the existence of a solution. One simple example is finding a collision in a (compressing) hash function, where a collision is guaranteed by the pigeonhole principle. Another example is the standard game theoretical notion of equilibrium due to Nash [Nas50]. The existence of a Nash equilibrium is guaranteed in any finite strategic game, whereas the problem of finding one is eluding the algorithmic game theory community with no known general efficient algorithm. Motivated by some of the above observations, Megiddo and Papadimitriou [MP91] proposed the complexity class TFNP (for Total Function Nondeterministic Polynomial-time), which is the class of all NP search problems with a guaranteed solution. Papadimitriou [Pap94] subsequently introduced a taxonomy of the problems inside TFNP that gathers the problems into classes based on the type of combinatorial argument establishing their totality.

The problem of finding a local optimum (not necessarily in the continuous setting) defines its own subclass of TFNP called PLS (for Polynomial Local Search [JPY88]). The canonical problem for PLS, called Local-Search, is given by a pair of circuits N and V that assign to any n-bit string x a polynomial sized neighborhood $\mathsf{N}(x)$ and a non-negative integer value $\mathsf{V}(x)$. The goal is to find a local optimum, i.e., an x such that $\mathsf{V}(x) \geq \mathsf{V}(x')$ for all $x' \in \mathsf{N}(x)$. The totality of this problem is guaranteed as a consequence of the observation that every finite directed acyclic graph has a sink. Other natural problems were shown to be complete for PLS; e.g., finding a *pure* Nash equilibrium in congestion games [FPT04], or finding a locally optimal maximum cut [SY91]. As for hardness, a series of exponential query complexity lower bounds for Local-Search [Ald83, LTT89, Aar06, Zha09] established black-box hardness for PLS for deterministic, randomized and even quantum algorithms. On the other hand, no white-box hardness for PLS are currently known under any cryptographic assumptions.

What about the problem of finding a local optimum in the *continuous* setting? Daskalakis and Papadimitriou [DP11] introduced the class CLS (for Continuous Local Search) to capture the exact computational complexity of continuous local search. The canonical problem for CLS is called Continuous-Local-Optimum (CLOpt). Unlike Local-Search, which is a discrete problem over the Boolean hypercube $\mathcal{B}^n$, an instance of Continuous-Local-Optimum is given by two functions f and p over the unit cube $[0,1]^3$ which are assumed to be λ-Lipschitz continuous.[1] The function f maps the unit cube to itself and the function p assigns to every point of the unit cube a real value. The goal is to find an ε-approximate local optimum of p with respect to f

[1] A function f is λ-Lipschitz continuous if for any x, x' it holds that $|f(x) - f(x')| \leq \lambda |x - x'|$.

(i.e., a point x such that $p(f(x)) - p(x) \leq \varepsilon$) or two points that violate the λ-Lipschitz continuity of either f or p. Daskalakis and Papadimitriou [DP11] showed that CLS *contains* many important computational problems for which there is, despite a lot of effort, no known polynomial time algorithm; e.g., finding mixed Nash equilibria in congestion games [Ros73], or solving simple stochastic games of Shapley and Condon [Sha53, Con92]. Moreover, they showed that CLS lies in the intersection of PLS and the class PPAD.

The class PPAD (for Polynomial Parity Argument on Directed graphs) has received particular attention in part due to the fact that one of its complete problems is finding Nash equilibra in strategic games [DGP09, CDT09]. The canonical problem for PPAD, called End-of-Line (EOL), is given by a successor circuit S and a predecessor circuit P that both map n-bit strings to n-bit strings, and thus implicitly define a graph with 2^n vertices of degree at most two (there is an edge between u and v iff $u = \mathsf{P}(v)$ and $v = \mathsf{S}(u)$). Given a source in this graph, the goal is to find a sink or an alternative source. The totality of this problem can be established based on the handshaking lemma.[2] Similarly to PLS, also for PPAD there are known black-box hardness results, e.g., exponential query complexity lower bounds for finding Brouwer fixed points (another important complete problem for PPAD) [HPV89, Bab14]. Recently, Bitanski, Paneth and Rosen [BPR15] constructed hard instances of End-of-Line under cryptographic assumptions and showed the first white-box hardness result for PPAD (the cryptographic assumptions were improved in [GPS16]). An overview of the above subclasses of TFNP and the known hardness results is depicted in Figure 4.1 (see Section 2.5 for the formal definitions of the various subclasses of TFNP).

Given that there are oracle separations showing that in the relativized setting both PLS is not reducible to PPAD (cf. [BM04]) and PPAD is not reducible to PLS (cf. [Mor01]), and since CLS is contained in the intersection of PLS and PPAD, it follows that relative to these oracles CLS is a proper subclass of both PLS and PPAD. This suggests that continuous local search might be an easier problem than the problems of finding approximate local optima or approximate Brouwer fixed points. Moreover, unlike in the case of PPAD and PLS, no hardness result is known for CLS; neither in terms of query complexity (even in the random oracle model), nor under any cryptographic assumption (including even strong notions of obfuscation used to establish the known white-box hardness results for PPAD).

In this work we show the first hardness results for CLS. We stress that among the various subclasses currently defined in TFNP, no non-trivial[3] subclass is known to be contained in CLS. Specifically, we prove an exponential query complexity lower bound, and give hardness from several cryptographic assumptions used in previous works for showing cryptographic hardness for PPAD.

[2] The handshaking lemma states that every finite undirected graph has an even number of vertices with odd degree.

[3] The notion of being "non-trivial" refers to a class that is not known to be solvable in polynomial-time.

4.2 Our Results

In order to study hardness of continuous local search, we introduce a new total search problem we call End-of-Metered-Line (EOML), which is similar in spirit to the End-of-Line problem. An instance of End-of-Metered-Line is given by three circuits S, P, and V. The circuits S and P act exactly like the successor and the predecessor in the End-of-Line problem, i.e., they implicitly define a graph in which every vertex has degree at most two. The goal also remains the same, i.e., given that 0^n is a source, we ask to find a sink or a source different from the trivial one at 0^n. Since 0^n is a source, it follows from the handshaking lemma that there exists a sink with respect to S and P, and thus End-of-Metered-Line is a total problem.

The additional circuit V is intended to aid in solving this task by providing extra information about the structure of the implicit graph. Specifically, V acts as an odometer for the line starting at the initial source 0^n, that is, it reveals how many steps of S it takes to reach any vertex x from 0^n.[4] Recall that the existence of a solution for problems inside TFNP is ensured syntactically and not by a promise on the instances. In particular, we cannot make any promise on the behavior of V. Though, we cannot efficiently verify that V indeed answers as expected for every vertex. Therefore, we enforce that the addition of circuit V aids in finding a solution by letting any vertex attesting that V deviates from its expected behavior to constitute an additional solution. First, the initial source 0^n must be the only vertex with value 1. Second, any vertex x with a non-zero value i must be succeeded with a vertex with value $i+1$ and preceded with a vertex with value $i-1$. Any vertex violating either of these two conditions is defined to be a solution. The formal definition (cf. Definition 32) together with additional discussion is given in Section 4.4.

Our first (and the most technical) result shows that the valuation circuit puts End-of-Metered-Line in a much smaller class than the End-of-Line problem. We show that End-of-Metered-Line is reducible to Continuous-Local-Optimum (see Definition 24), the canonical problem defining the class CLS.

Theorem 9. End-of-Metered-Line *is contained in CLS.*

Next, we show that continuous local search remains hard by proving hardness for End-of-Metered-Line. Our hardness results are in terms of both black-box and white-box. In the black-box case, we show an exponential query complexity lower bound for End-of-Metered-Line.

Theorem 10. *The randomized query complexity of* End-of-Metered-Line *is* $\Omega\left(2^{n/2}/\sqrt{n}\right)$.

By combining Theorem 13 with Theorem 12, we get as a corollary the black-box hardness of CLS:

[4] In other words, the circuit V serves the same purpose as milestones placed along the roads throughout the Roman Empire; it reassures the traveler that the correct path is being followed and indicates the number of steps taken from the ultimate origin at the *Milliarium Aureum*, the golden milestone at the Forum Romanum (in our case at 0^n).

Corollary 3. *In the query model, any randomized algorithm solving the* Continuous-Local-Optimum *problem up to n-digits of precision must perform at least* $2^{\Omega(n)}$ *queries.*

In the white-box case, where the functions are given via circuits of polynomial size in n, we use cryptographic assumptions, specifically, the notion of secure circuit obfuscation. Our results extend the recent result of Bitanski et al. [BPR15]. First, [BPR15] defined a problem called Sink-of-Verifiable-Line (SVL) (see Section 4.6.2), and they showed that it is hard under the above cryptographic assumptions. Second, they gave a reduction from SVL to End-of-Line yielding that any hardness of SVL translate to hardness for End-of-Line.

In this work, we give a reduction from SVL to End-of-Metered-Line, which translates any SVL hardness to End-of-Metered-Line hardness.

Theorem 11 (Informal)**.** *There exists a polynomial-time reduction from* Sink-of-Verifiable-Line *to* End-of-Metered-Line*.*

Garg et al. [GPS16] and Komargodski and Segev [KS17a] showed hardness of SVL from weaker assumptions cryptographic assumptions than [BPR15]. Therefore, combining Theorem 11 and the hardness of [GPS16] we get the following corollary:

Corollary 4. *Assume there exist one-way permutations and indistinguishability obfuscation for P/Poly. Then the class CLS (and thus also PLS) is hard for polynomial-time algorithms.*

We note that prior to this work no white-box hardness results based on cryptographic assumptions were known either for CLS or for PLS. An overview of our results in the context of TFNP is depicted in Figure 4.1.

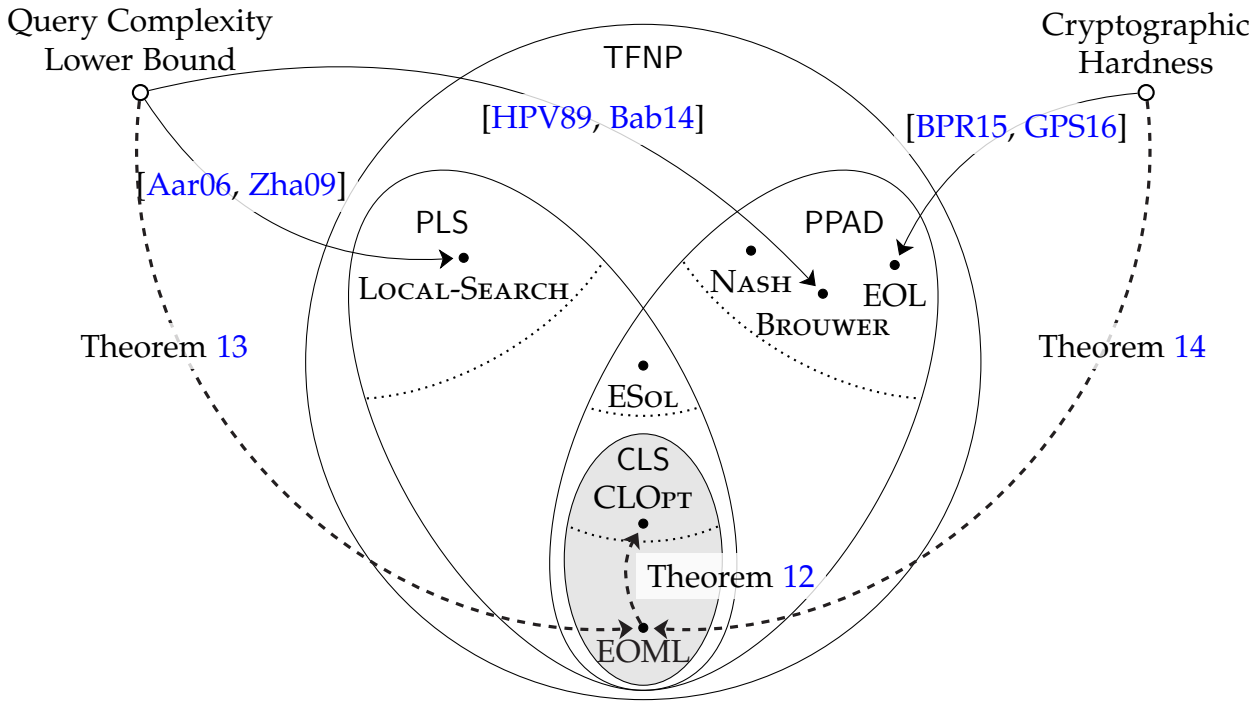


Figure 4.1: Depiction of our and previous results in the context of TFNP. Problems known to be complete for the respective classes are drawn above a dotted line.

Further applications of our reductions. One key property of our reductions is that they allow us to strengthen multiple recent results originally proven for the class PPAD. In particular, we can show that they hold for the class CLS, and thus also for the class PLS. Below we give several such examples.

In a recent work, Rosen, Segev and Shahaf [RSS16] studied the connections between average-case PPAD hardness and standard cryptographic assumptions. They gave several impossibility results for existence of reductions among End-of-Line and various cryptographic objects that treat the primitives as black-box. First, they proved that average-case Sink-of-Verifiable-Line hardness does not imply one-way functions in a black-box way. Thus, since there is a reduction from Sink-of-Verifiable-Line to End-of-Line, they showed that average-case PPAD hardness does not imply one-way functions in a black-box way. Note that our reduction from Sink-of-Verifiable-Line to CLS gives as an immediate corollary that even the stronger assumption of average-case CLS hardness does not imply one-way functions.

Corollary 5. *There is no black-box construction of a one-way function from a hard-on-average distribution of CLS instances.*

Second, [RSS16] showed that there is no *black-box* reduction from End-of-Line to Sink-of-Verifiable-Line. Using our reduction from Sink-of-Verifiable-Line to

End-of-Metered-Line, we get that *any* reduction, either black-box or white-box, from End-of-Line to Sink-of-Verifiable-Line would show that PPAD is contained in CLS, and thus also contained in PLS (recall that there are oracle separations between all the three classes). Formally, we get the following corollary:

Corollary 6. *Assuming that PPAD is not contained in PLS, then there is no reduction (black-box or white-box) from* End-of-Line *to* Sink-of-Verifiable-Line.

In another recent work, Bitansky, Degwekar and Vaikuntanathan [BDV16] studied the relationship between PPAD and statistical zero knowledge (SZK) and the class $\mathsf{NP} \cap \mathsf{coNP}$. They showed that in the fully black-box framework of [AS15] there is no construction of a hard problem in SZK or $\mathsf{NP} \cap \mathsf{coNP}$ from hard problems in PPAD. However, they did not rule out getting such hard problem from hardness of sub-classes of PPAD. Thus, applying our results we get the following stronger statement:

Corollary 7. *There is no fully black-box construction of a hard-on-average problem in SZK or $\mathsf{NP} \cap \mathsf{coNP}$ from a hard-on-average distribution of CLS instances.*

4.3 Our Techniques

In this section we give a high-level description of the main techniques used to establish our results. Formal definitions and complete proofs are provided in the subsequent sections.

4.3.1 Reduction to Continuous-Local-Optimum

The core of our hardness results for continuous local search is a reduction showing that End-of-Metered-Line is contained in the class CLS. Our reduction takes any instance $(\mathsf{S}, \mathsf{P}, \mathsf{V})$ of End-of-Metered-Line over the domain $\mathcal{B}^n$ and builds an instance of Continuous-Local-Optimum consisting of the following: a function f mapping the unit cube to itself, a function p assigning a real value to any point in the unit cube and two constants $\varepsilon, \lambda > 0$, where f and p are λ-Lipschitz. Our reduction guarantees that any ε-approximate local optimum (i.e., a point x such that $p(f(x)) - p(x) \leq \varepsilon$) corresponds to a solution to $(\mathsf{S}, \mathsf{P}, \mathsf{V})$.

Any instance $(\mathsf{S}, \mathsf{P}, \mathsf{V})$ implicitly defines a graph over $\mathcal{B}^n$. The graph is defined similarly to the graph corresponding to any End-of-Line instance, i.e., there is an edge between u and v iff $u = \mathsf{P}(v)$ and $v = \mathsf{S}(u)$, and additionally, we discard all vertices with value 0, and all edges $(x, \mathsf{S}(x))$ such that $\mathsf{V}(\mathsf{S}(x)) - \mathsf{V}(x) \neq 1$ (i.e., all the edges where V does not increment correctly). Notice that there are no cycles in this graph since it is impossible to assign incremental values consistently along a cycle. We show how to embed this discrete graph defined by $(\mathsf{S}, \mathsf{P}, \mathsf{V})$ into a continuous function defined over the unit square.[5]

[5] In order to match the definition of Continuous-Local-Optimum, we should construct functions

Embedding a Single Line. For simplicity, we begin by considering the graph to be a single line: $0^n \to \mathsf{S}(0^n) \to \mathsf{S}^2(0^n) \to \cdots$. Some approaches for embedding such discrete path into a continuous function over the unit square were discussed in the literature [HPV89, CD09]. However, as we explain below, these works do not yield a reduction to Continuous-Local-Optimum since they do not build the corresponding valuation function p, and it is not clear how to extend them to construct one.

For example, Chen and Deng [CD09] showed how to embed any End-of-Line instance into a continuous function over the unit square, an instance of the Brouwer problem. That is, they constructed a function f over the unit square such that any (approximate) fixed point of f corresponds to a solution to the End-of-Line instance. Their reduction works by considering a discrete grid on the unit square, and showing how to embed the End-of-Line graph as a directed path onto the lines of this grid. On the grid points, the function f is defined such that it points towards the direction of the path at the points lying on the path, and such that f points towards $(0,0)$ at any point lying off the path. For any intermediate point (off the grid), f is defined by interpolation of its four neighbors on the grid. The resulting function f creates a flow along the path towards its end and a flow towards $(0,0)$ in the remaining area of the unit square. Moreover, any fixed-point of f lies near the end of the path.

Ultimately, we would want to define p on the grid (and by interpolation off the grid) to match the behavior of f, and such that any local optimum with respect to f and p corresponds to a solution to the End-of-Metered-Line instance. Specifically, we need to assign increasing values along the different flows defined by f in a locally and efficiently computable way. The natural way to achieve this would be to define p as follows: to each point off the path, we assign a value that corresponds to its distance from the point $(0,0)$, and to each point lying on the path, we use V to assign a value corresponding to its distance from the beginning of the path.

The hope is that it is possible to choose appropriate values in such a construction of p that do not introduce local optima except at a fixed point of f. However, it turns out that this approach fails. In fact, one can show that any set of values for p on the grid will introduce a new local optimum that does not correspond to a solution for $(\mathsf{S},\mathsf{P},\mathsf{V})$. The problem is that there are points on the path where f points in the opposite direction than at a nearby point off the path. The interpolation between such two points then, in the combination with the p values, introduces a local optimum at points near the path (but possibly far from its end). Actually, this issue would introduce exponentially many new local optima from which it is not possible to extract a solution.

In order to overcome the above (and other) problems, we construct a new embedding of the discrete path into the continuous function f over the unit square that allows to naturally define the valuation function p. We use some of the building blocks from [CD09], though our construction has major differences. In order to en-

over the unit cube and not over the unit square. However, it is easy to extend any Brouwer function f mapping the unit square to itself to a continuous function over the unit cube by simply copying over the third coordinate (i.e., for all $(x,y,z) \in [0,1]^3$, define $f'(x,y,z) = (f(x,y),z)$).

sure that the change in direction of f is proportional to the change in value of p, we tailor the function f to create an additional transition layer between the path and its surrounding area. This enables us to define values for p that do not introduce unnecessary local optima along *most* parts of the path. In particular, if the path traverses in only two directions (say up and right), then no unnecessary local optimum is introduced at all (yet another problem with using the reduction in [CD09] is that their path traverses the unit square in all four directions).

The Staircase Embedding. The natural step to take, at this point, is to make sure that the path traverses the unit square only in two directions, resulting in a path that resembles a "staircase". To achieve this, we route the staircase through the unit square so that the coordinates of every point on the staircase encode a vertex in the EOML graph. In particular, we split the unit square into square sub-blocks so that we can identify each block along the side of the square with one of the 2^n non-zero values given out by V. Then, we do the same for each sub-block so that we can identify the small squares along the bottom edge of each sub-block with strings in $\mathcal{B}^n$. Thus, in the end we have split the unit square into 2^{4n} small squares. Finally, for every vertex x such that $V(x) = i$ we create a path connecting the point $(x, 0)$ in block (i, i) with the point $(\mathsf{S}(x), 0)$ in the block $(i + 1, i + 1)$. The points are connected via three line segments going through (x, x) in block (i, i) and $(S(x), x)$ in block $(i + 1, i)$ (see Figure 4.2a). Note that for each line segment, it is possible to identify only from the coordinates of a point whether the line passes through it. For example to test that a point (z_1, z_2) in block (i, i) lies on the first line segment, we check that $\mathsf{V}(z_1) = i$ and that $z_2 \leq z_1$. Testing for the other two line segments is performed similarly.

Now that we have defined the function f so that it results in an embedding of the path as a staircase in the unit square, we can assign the p values to the grid points. To avoid local optima except for endpoints of the path, we assign incremental values for all grid points on the path. That is, for any point on the path, its value is the distance from the beginning of the path. Note that we need to be able to compute this distance locally (so that it can be efficiently computed by the valuation circuit V). Here, we exploit again the structure of our staircase embedding: for any point on the staircase, it holds that the distance from the beginning is exactly its Manhattan distance from $(0, 0)$ (i.e., the sum of its coordinates).

Given the modified version of f that results in a staircase embedding, and the above definition of p, it is readily possible to prove that, in the special case where the End-of-Metered-Line graph is a single line, the only local optimum is exactly at the end of the staircase. Thus, by our construction of the embedding, the coordinates of the unique local optimum can be used to extract a solution to the original End-of-Metered-Line instance $(\mathsf{S}, \mathsf{P}, \mathsf{V})$. We give an example of an embedding of a single line in Figure 4.5. Next, we show how to handle general graphs which might correspond to more than a single line.

Handling General Graphs. Consider the case where the graph is not a single line, but a collection of lines. The embedding of such a graph will result in a collection of staircases. The problem is that these staircases might intersect, and any such intersection would introduce a new local optimum. Since the intersection could lie at an arbitrary point on two staircases, it is in general not possible to extract from the local optimum a solution to the End-of-Metered-Line instance. Thus, we use an idea from [CD09] and locally alter the flow of the staircases at any crossing point in a way that eliminates the intersection on one hand and preserves the staircase structure on the other. For any crossing of a horizontal line segment with a vertical line segment, we disconnect the lines so that they do not touch and switch their directions so that afterwards the horizontal line turns up and the vertical line turns right (see Figure 4.2b). Note that even though this transformation alters the topological structure of the staircases in a major way, it preserves the endpoints of the original staircases and does not introduce any new ones.

In the most general case, an arbitrary End-of-Metered-Line graph can contain a collision of two lines, that is, two lines that at some point merge to one. Notice that this merging point is a solution for $(\mathsf{S}, \mathsf{P}, \mathsf{V})$. In terms of our embedding, any such collision will introduce a local optimum from which we can extract the solution. The full details of the reduction, its proof, and complete discussion of how to extract solutions from any local optimum are given in Section 4.5. In Figure 4.6, we give an example of an embedding of the lines near a crossover after the modification. In Figure 4.7, we give an example of an embedding of a graph with multiple lines and collisions into the unit square.

4.3.2 Query Complexity Lower Bound for End-of-Metered-Line

Our query complexity lower bound builds on the techniques used for proving black-box hardness of PLS, and in particular, on query complexity lower bounds for the Local-Search problem over the Boolean hypercube. Our starting point is the tight randomized query complexity lower bound of $\Theta(2^{n/2}\sqrt{n})$ for Local-Search on the Boolean hypercube by Zhang [Zha09]. Specifically, Zhang [Zha09] showed how to construct a distribution of self-avoiding random paths over the n-dimensional hypercube, such that any randomized algorithm that finds the endpoint of the path given only oracle access to the path (i.e., it can learn whether a vertex lies on the path or not) must query exponentially many vertices of the hypercube. We show how to exploit the specific structure of Zhang's construction to build a distribution of End-of-Metered-Line instances that require exponential query complexity.

In particular, the label of every vertex $v \in \mathcal{B}^n$ on the path constructed in [Zha09] can be parsed as a pair $(x, z) \in \mathcal{B}^m \times \mathcal{B}^{n-m}$, where the x component is used to perform a random walk and the z component stores a step-counter for the random walk to avoid intersections of the path with itself. Our EOML oracle defines a graph corresponding to a single line chosen according to the above structured random walk, and we use the step-counter to assign incremental values along the path. The ora-

cle answers for every query hitting the path with the respective predecessor vertex, successor vertex, and the distance from the origin of the random walk. We show that any query to the End-of-Metered-Line instance can be implemented by at most n queries to the path of Zhang. Hence, we get an $\Omega(2^{n/2}/\sqrt{n})$ randomized query complexity lower bound for End-of-Metered-Line. See Section 4.6.1 for a detailed proof.

4.3.3 Cryptographic Hardness for End-of-Metered-Line

Our cryptographic hardness result for End-of-Metered-Line builds upon previous works on cryptographic hardness for PPAD. Bitanski, Paneth and Rosen [BPR15] showed hardness of End-of-Line under the assumption of existence of injective one-way functions and indistinguishability obfuscation (both with sub-exponential security).

A cryptographic obfuscator is a complier that transforms any given circuit to a "scrambled" one, which is functionally equivalent on one hand, but hides its implementation details on the other. The theoretical study of obfuscation was initiated by Barak et al. [BGI+12a], who suggested the notion of virtual black-box obfuscation: anything that can be efficiently computed from the obfuscated circuit, can be also computed efficiently from black-box access to the circuit (see Definition 35). Their main result was that this notion of obfuscation cannot be generally achieved.

As a way to bypass their general impossibility result, they introduced a plausibly weaker notion of indistinguishability obfuscation. An indistinguishability obfuscator is a compiler that guarantees that if two circuits compute the same function, then their obfuscations are computationally indistinguishable (see Definition 34). Furthermore, since the first candidate construction of indistinguishability obfuscation [GGH+13a] was proposed, many other constructions have followed suite [PST14, GLSW14, AB15, BVWW16].

Abbot, Kane and Valiant [AKV04] used the power of secure obfuscation to prove a hardness result for End-of-Line (the canonical complete problem for PPAD) assuming virtual black-box obfuscation. However, given that virtual black-box obfuscation is not achievable in general, it remained an obvious important open problem to show hardness for PPAD under weaker cryptographic assumptions. Bitanski et al. [BPR15] were the first to solve this open problem. They showed that, assuming one-way functions and indistinguishability obfuscation (both with subexponential security), there exist hard instances of the End-of-Line problem. Their proof followed two main steps. First, motivated by [AKV04] they defined a problem called Sink-of-Verifiable-Line (SVL), and they showed that SVL is hard under the above cryptographic assumptions. Second, they gave a reduction from SVL to End-of-Line yielding the conclusion that End-of-Line is hard under the same assumptions.

Subsequently, Garg et al. [GPS16] showed that the cryptographic assumption for the hardness of Sink-of-Verifiable-Line can be weakened. In particular, they showed hardness of Sink-of-Verifiable-Line under the assumption of existence of

one-way permutations and indistinguishability obfuscation both with *polynomial security*, and furthermore they showed an alternative construction of hard instances of Sink-of-Verifiable-Line assuming compact functional encryption (a special kind of public-key encryption which supports restricted secret keys that enable a key holder to learn a specific function of the encrypted data, but learn nothing else), which is a plausibly weaker cryptographic primitive than indistinguishability obfuscation. Using the reduction to End-of-Line of [AKV04, BPR15] their result implies cryptographic hardness for PPAD.

Our white-box hardness result for End-of-Metered-Line is achieved by constructing a reduction from SVL to End-of-Metered-Line. This allows us to use any hardness result for SVL (either [BPR15] or[GPS16]) and get a corresponding hardness for End-of-Metered-Line.

An instance of the SVL problem consists of a successor circuit S, a target index $1 \leq T \leq 2^n$ and a verification circuit V with the promise the $V(x, i) = 1$ if and only if $\mathsf{S}^{i-1}(0^n) = x$. The goal is to find an x such that $V(x, T) = 1$. In this work, we reduce any SVL instance $(\mathsf{S}, \mathsf{V}, T)$ to an equivalent instance $(\mathsf{S}', \mathsf{P}', \mathsf{V}')$ of End-of-Metered-Line. The approach of [BPR15] (suggested in [AKV04]) was to leverage the verification circuit V to create a new *reversible* successor circuit S' (and thus get P' for free). In our approach, we show how to leverage the verification circuit to not only get a reversible successor circuit S' but also a corresponding valuation circuit V' that correctly assigns incremental values along the path starting at the initial source. Thus, we build an equivalent instance $(\mathsf{S}', \mathsf{P}', \mathsf{V}')$ of End-of-Metered-Line.

Our work clarifies the implications of the recent hardness results for SVL based on cryptographic notions of software obfuscation. In particular, we show that the cryptographic hardness for SVL of [GPS16] based on indistinguishability obfuscation and one-way permutations (or compact functional encryption) implies hardness for problems inside the class CLS, currently one of the lowest known non-trivial classes inside TFNP. Since End-of-Metered-Line is contained in CLS which lies in the intersection of PLS and PPAD, we also get a hardness results of PLS for which there were previously no known hardness results based on cryptographic assumptions. See Section 4.6.2 for the complete construction and proof.

4.3.4 Organization of the Paper

The rest of the paper is organized as follows. In Section 2.5 we give the formal definitions and an overview of related work on complexity of total search problems and their classes. In Section 4.4 we define the End-of-Metered-Line problem and discuss some of its basic properties. In Section 4.5 we describe our reduction from End-of-Metered-Line to Continuous-Local-Optimum. In Section 4.6.1 we show a query complexity lower bound for End-of-Metered-Line. In Section 4.6.2 we describe a cryptographic hardness result for End-of-Metered-Line.

4.4 End-of-Metered-Line

In this section we define a new total search problem, which we call End-of-Metered-Line, that will serve as a basis of our hardness results for CLS. Our new problem extends End-of-Line, given by a successor circuit S and a predecessor circuit P, by including an additional valuation circuit V. The purpose of the valuation circuit is to act as an odometer for line starting at the initial source 0^n, that is, it reveals how many steps of S it takes to reach any vertex x from 0^n. In particular, we would like to have the promise that $\mathsf{V}(x) = i$ if and only if $\mathsf{S}^{i-1}(0^n) = x$. Of course, it is not possible to efficiently verify that indeed V answers as expected for every vertex. Moreover, TFNP (see Definition 43) is a class of problems where a solution must exist, without additional promises. Thus, we introduce a compromise between certifying correctness and having a promise: We make no promises on the behavior of V but let any vertex attesting that V deviates from its expected behavior to be an additional solution. The formal definition is given below.

Definition 32 (End-of-Metered-Line)**.** *Given circuits* $\mathsf{S}, \mathsf{P} \colon \mathcal{B}^n \to \mathcal{B}^n$*, and* $\mathsf{V} \colon \mathcal{B}^n \to [2^n] \cup \{0\}$ *such that* $\mathsf{P}(0^n) = 0^n \neq \mathsf{S}(0^n)$ *and* $\mathsf{V}(0^n) = 1$*, find a string* $x \in \mathcal{B}^n$ *satisfying one of the following:*

1. *either* $\mathsf{P}(\mathsf{S}(x)) \neq x$ *or* $\mathsf{S}(\mathsf{P}(x)) \neq x \neq 0^n$*,*
2. $x \neq 0^n$ *and* $\mathsf{V}(x) = 1$*,*
3. *either* $\mathsf{V}(x) > 0$ *and* $\mathsf{V}(\mathsf{S}(x)) - \mathsf{V}(x) \neq 1$ *or* $\mathsf{V}(x) > 1$ *and* $\mathsf{V}(x) - \mathsf{V}(\mathsf{P}(x)) \neq 1$*.*

Given an End-of-Metered-Line instance $(\mathsf{S}, \mathsf{P}, \mathsf{V})$, we say that a vertex x is a solution of the first, the second, or the third type if it satisfies the corresponding item in the above definition. Notice that solutions of the *first type* are also solutions to the End-of-Line instance (S, P) defined by the successor and the predecessor circuit of the End-of-Metered-Line instance. The solutions of the second and the third type correspond to some indication that V is "untruthful", that is, V is not answering according to the distance of the vertex from 0^n. In particular, solutions of the *second type* indicate that V is giving values to a different line other than the one starting from 0^n. Solutions of the *third type* indicate that V is reporting a "miscount" at some vertex x.

The Intuition Behind our Definition. There are several ways how to define the "expected behavior" of V, where for each we would obtain a possibly different set of solutions, and thus a variation of the problem which, in general, is not equivalent to Definition 32. In our definition, we require V to give additional information only for vertices on the line, and thus we allow V to set the zero value to vertices off the line starting at 0^n. That is, a vertex x such that $\mathsf{V}(x) = 0$ and $\mathsf{V}(\mathsf{S}(x)) = 0$ is *not* considered as a solution. In particular, V can set the zero value to all vertices on any cycle without introducing new solutions. This property is crucial for our

hardness results (cf. Section 4.6.1 and Section 4.6.2). In both cases, we show hardness by a reduction from a certain problem to End-of-Metered-Line. To maintain the hardness, it is crucial for the valuation functions constructed in our reductions that we are able to assign the zero value without introducing trivial solutions.

Note that in our definition of End-of-Metered-Line, solutions of the third type do not include solutions of the second type. In particular, an intermediate vertex x on some path such that $V(x) = 1$ and all predecessors of x have value 0 would not constitute a solution of the third type. Thus, it is important that we include also solutions of the second type to enforce that information provided by V is useful.

Moreover, we do not require V to distinguish whether a vertex lies on the line or not. Consider two vertices $x \neq x'$ such that $V(x) = V(x') \neq 0$. It is clear that both cannot be at the same distance from 0^n, and thus they are a witness of V deviating from its expected behavior. However, we do not consider such pairs to be a solution, as there is no way to distinguish which vertex is not on the line originating at 0^n. It is possible to add an additional type of solution corresponding to such x and x'. We note that all of our results naturally extend to this variant of End-of-Metered-Line.

Warm-Up: EOML lies in PPAD ∩ PLS. We show that End-of-Metered-Line lies at the intersection of PLS (see Definition 22) and PPAD (see Definition 21). As pointed out by Daskalakis and Papadimitriou [DP11], proving that a problem lies in the intersection of PLS and PPAD is equivalent to reducing the problem both to a complete problem in PLS and to a complete problem in PPAD. They showed that Either-Solution, i.e., given an instance of Local-Search and an instance of End-of-Line to find a solution to either one of them, is a complete problem for PLS ∩ PPAD. We have already shown that End-of-Metered-Line is contained in PPAD since for any instance (S, P, V) of End-of-Metered-Line, the circuits (S, P) constitute an instance of End-of-Line. Additionally, End-of-Metered-Line can also be reduced to Local-Search, which together with the above straightforward reduction to End-of-Line completes the reduction to Either-Solution. Thus, End-of-Metered-Line is contained in PLS ∩ PPAD.

Lemma 2. End-of-Metered-Line *is reducible to* Local-Search.

Proof. Given an End-of-Metered-Line instance (S, P, V) we define a Local-Search instance (S', V), where $S'(x)$ outputs 0^n if $V(x) = 0$ and otherwise outputs $S(x)$. Let x be any solution to (S', V). Note that $V(x) \neq 0$, since every vertex x such that $V(x) = 0$ is under S' followed by 0^n and it cannot be a local optimum (recall that $V(0^n) = 1$). Moreover, $V(x) \geq V(S'(x)) = V(S(x))$ which implies that $V(x) \neq V(S(x)) - 1$. Hence, x is a solution of the third type to the original End-of-Metered-Line instance (S, P, V). □

4.5 End-of-Metered-Line is in CLS

In this section we show that End-of-Metered-Line is contained in CLS (see Section 2.5). In particular, we give a reduction from any instance $(\mathsf{S}, \mathsf{P}, \mathsf{V})$ of End-of-Metered-Line (cf. Definition 32) to an equivalent instance $(f, p, \varepsilon, \lambda)$ of the problem Continuous-Local-Optimum (cf. Definition 24). Our reduction takes the discrete graph implicitly defined by any End-of-Metered-Line instance and embeds it inside the unit square in a continuous manner. In particular, our embedding defines the function f for the new Continuous-Local-Optimum instance that resembles a staircase (or a collection of staircases in case the valuation circuit V outputs non-zero values on more than one line). Additionally, we define a continuous valuation function p for all the points of the unit square with the property that from any local optimum of p under f it is possible to reconstruct a solution to the original EOML instance.

Theorem 12. End-of-Metered-Line *is reducible to* Continuous-Local-Optimum.

Our reduction takes any instance $(\mathsf{S}, \mathsf{P}, \mathsf{V})$ of End-of-Metered-Line and produces an instance of Continuous-Local-Optimum, defined by functions $f\colon [0,1]^3 \to [0,1]^3$, and $p\colon [0,1]^3 \to [0,1]$ and constants $\varepsilon, \lambda > 0$. We begin by describing the functions $f\colon [0,L]^2 \to [0,L]^2$ and $p\colon [0,L]^2 \to [0,4L]$, where $L \in \mathbb{N}$ will be defined later. Afterwards, we show how to convert f and p to the appropriate domains and we define the constants ε and λ.

Consider a tessellation of the domain $[0,L]^2$ with $L \cdot L$ unit squares. Similarly to some previous works [HPV89, Rub15], we call the border area of the domain the frame, and the area inside the frame the picture. We set the frame to be of width 5 unit squares. That is, the picture covers exactly the area $[5, L-5]^2$. The picture is comprised of square blocks, where the side of each block is of length $10 \cdot 2^n$ unit squares. The picture contains 2^n such blocks. Thus, we get that $L = 10(2^{2n}+1)$, where $10 \cdot 2^{2n}$ is the number of squares along the side of the picture and 10 is added to account for the width of the frame on both sides of the picture.

The coordinates of any point $(X, Y) \in [5, L-5]^2$ within the picture can be parsed as a pair of triplets: $X = (b_x, s_x, x)$ and $Y = (b_y, s_y, y)$. First, $(b_x, b_y) \in [2^n] \times [2^n]$ are the coordinates of the square block containing (X, Y). Second, $(s_x, s_y) \in [10 \cdot 2^n] \times [10 \cdot 2^n]$ are the coordinates of the unit square within this block. Third, $(x, y) \in [0,1]^2$ are the coordinates of (X, Y) within this square. Formally, we have that

$$X = 5 + (10 \cdot 2^n)(b_x - 1) + (s_x - 1) + x$$

and

$$Y = 5 + (10 \cdot 2^n)(b_y - 1) + (s_y - 1) + y.$$

We embed the line defined by the End-of-Metered-Line instance into the picture. The line starts at the bottom left corner of the picture and traverses the borders of the unit squares inside the picture in a staircase-like pattern. Below we use a mapping

from $\mathcal{B}^n$ to the set $[10 \cdot 2^n]$, and for any $x = (x_n, \ldots, x_1) \in \mathcal{B}^n$ denote by $\langle x \rangle = 10\left(\sum_{i=1}^{n} x_i 2^{i-1}\right) + 1$ its corresponding value (however, when clear from the context we simply write x). The line passes through points $((i, \langle x_i \rangle, 0), (i, \langle 0^n \rangle, 0))$ for all $(x_i, i) \in \mathcal{B}^n \times [2^n]$ such that $\mathsf{V}(x_i) = i$. We connect all pairs of points $((i, \langle x_i \rangle, 0), (i, \langle 0^n \rangle, 0))$ and $((i+1, \langle \mathsf{S}(x_i) \rangle, 0), (i+1, \langle 0^n \rangle, 0))$ via three line segments:

$$
\begin{aligned}
&((i, \langle x_i \rangle, 0), (i, \langle 0^n \rangle, 0)) \longleftrightarrow \\
&((i, \langle x_i \rangle, 0), (i, \langle x_i \rangle, 0)) \longleftrightarrow \\
&((i+1, \langle \mathsf{S}(x_i) \rangle, 0), (i, \langle x_i \rangle, 0)) \longleftrightarrow \\
&((i+1, \langle \mathsf{S}(x_i) \rangle, 0), (i+1, \langle 0^n \rangle, 0)).
\end{aligned}
$$

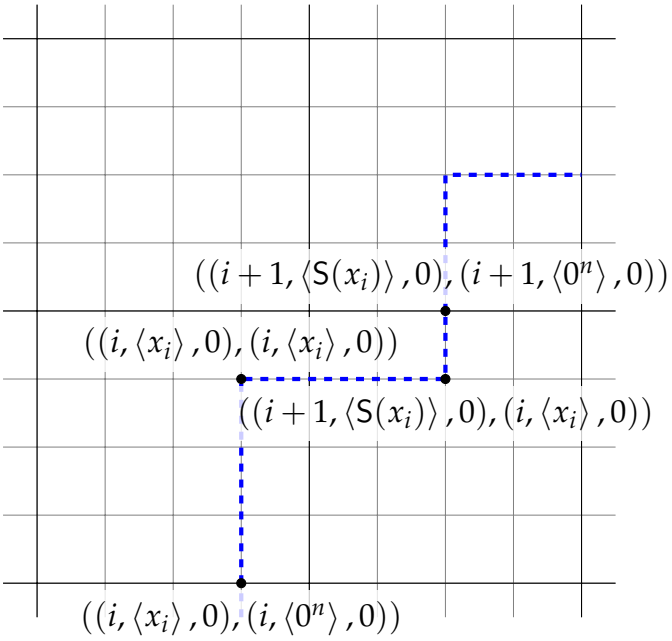
See Figure 4.2a for an illustration.

Note that in general it could be the case that $\mathsf{V}(x) = \mathsf{V}(x') = i$ for some $x \neq x'$ and there are crossing line segments inside the blocks (i, i) and $(i+1, i)$. To avoid such crossing of lines, we alter the behavior of the line in the vicinity of a crossing point as illustrated in Figure 4.2b. In particular, the neighborhood of any crossing point is changed according to a fixed crossing gadget inspired by the work of Chen and Deng [CD09]. Even though the lines are altered after applying the crossover section, the most important property of this transformation is that it neither removes existing ends of line nor it introduces additional ones.

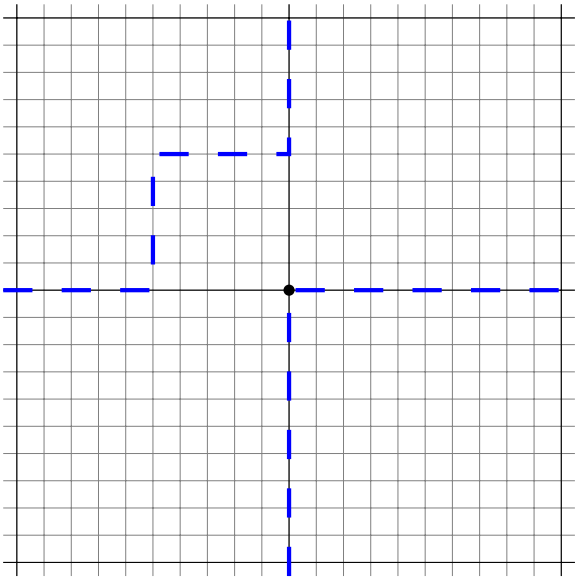
The function f is defined to create a "flow" along the above staircase. Formally, the function f is defined by a different displacement at each point. Each unit square inside $[0, L]^2$ is given a "color" corresponding to the displacement at the center of the square. For any other point z, the displacement is defined as the Cartesian interpolation of the f-values of the four centers surrounding z. That is, f is of the form $f(z) = z + \delta d(z)$ where $\delta > 0$ is some constant and $d(z) \in \mathbb{R}^2$ is the displacement function, which we describe next.

There are five basic displacements (depicted in Figure 4.3): blue $(-1, -1)$, green $(0, -1)$, purple $(-1, 0)$, yellow $(1, 0)$, and red $(0, 1)$. By default the color of each square is blue, which corresponds to the displacement $(-1, -1)$ (i.e., pointing down and left). The squares on the frame are colored according to a fixed assignment of colors. First, the squares in the leftmost column get yellow color and the squares in the second left column get the green color. Second, the squares in the bottom row get red color and the square in the row above it get purple color. Then, we color the bottom left corner of the frame according to Figure 4.4. Finally, we color the squares inside and below the picture corresponding to their distance from the line. Any square touching the line from left or above is colored yellow. Any square touching the line from right or below is colored red. Any square that is distance one from the line from left or above is colored green. Any square that is distance one from the line from right or below is colored purple.

The function $p \colon [0, L]^2 \to [0, 4L]$ is defined similarly to f. Each square is assigned an integer value corresponding to the value of p at the middle of the square, and the value of p at any other point is the Cartesian interpolation of the p-values at the



(a) A staircase segment connecting points representing x_i and $\mathsf{S}(x_i)$.



(b) The fixed template for avoiding crossing at a point that lies on two lines.

Figure 4.2: Details of our embedding of an End-of-Metered-Line instance into the unit square.

four centers surrounding it. We number every unit square in the tiling of $[0, L]^2$ by coordinates in $[L] \times [L]$, where the bottom left unit square gets coordinates $(1, 1)$ and

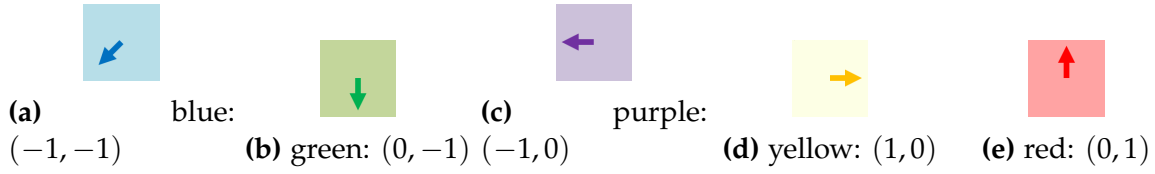


Figure 4.3: The five basic displacements.

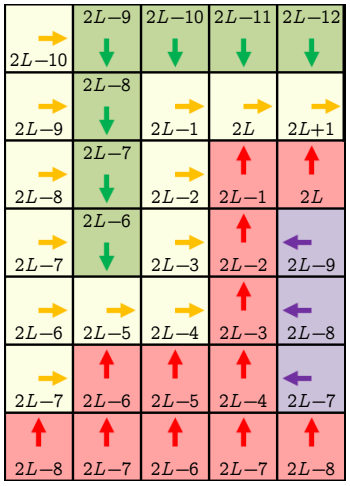


Figure 4.4: The f-values and the p-values of the squares close to the origin. The bottom left square is aligned at the bottom left corner of the frame.

the top left unit square gets coordinates (L, L). We assign the p-values to each square according to its color under f. Every square $(i, j) \in [L] \times [L]$ that is green, blue or purple gets value $2L - i - j$ (that is, the top right square gets value 0 and the bottom left square gets value $2L - 2$). Now we assign the p-values of the yellow and red squares. First, any square $(1, j)$ in the leftmost column gets value $2L - j - 2$. Second, any square $(i, 1)$ in the bottom row gets value $2L - i - 2$. Then, we assign the p-values in the bottom left corner of the frame according to Figure 4.4.

To assign the values to the remaining yellow and red squares, we begin by assigning a value corresponding to each step on the line inside the picture. The start of the line gets value $M = 2L + 1$, and after performing ℓ steps of unit length the value is $M + \ell$. Note that all the remaining yellow and red squares are adjacent to the line. If the square is below the line (i.e., it has red color) then it gets the value of its top left corner. If the square is above the line (i.e., it has yellow color) then it gets the value of its bottom right corner. For concrete example of a specific embedding with the corresponding values for f and p see Figure 4.5.

We note that the evaluation of f and p on any point can be performed locally. In particular, the values follow a simple fixed pattern on the frame. The values inside the picture are computed based on the proximity of the staircase that embeds the End-of-Metered-Line instance. Note that coordinates of any point on the staircase are sufficient to verify that the point indeed lies on the line. We give the complete descrip-

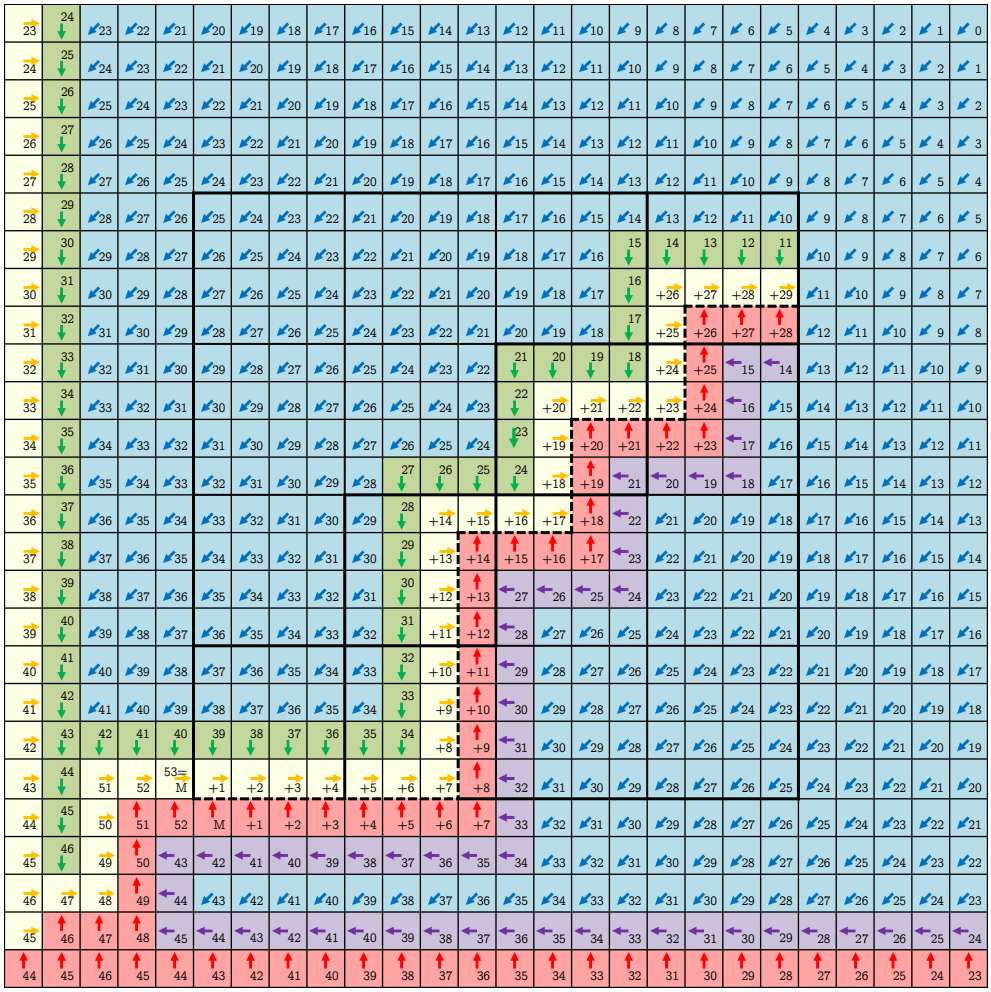


Figure 4.5: A simplified depiction (omitting the ten unit square buffer inside the blocks) of an instance of Continuous-Local-Optimum obtained by applying our reduction on an instance of End-of-Metered-Line. The dashed line is the staircase corresponding to embedding a single line $00 \to 11 \to 10 \to 01$ implicitly defined by the successor circuit S of the End-of-Metered-Line instance. The arrows indicate the displacement at the center of each square and the numbers indicate the corresponding p-values ("$+j$" is a shorthand for "$M + j$").

tion of procedure that decides whether a point lies on the staircase in Algorithm 4. Given this procedure it is easy to decide whether a square touches the staircase from above or below. We can simply check whether the square touches the staircase with its bottom right corner or its top left corner, which splits the squares adjacent to the staircase to those that lie above and those that lie below (see Algorithm 5). Knowing that a square lies above or below the line allows us to immediately assign the yellow and red colors. Otherwise, we test the neighbours of the square for being above or below to assign the green and purple color; in case all of these tests fail, the square

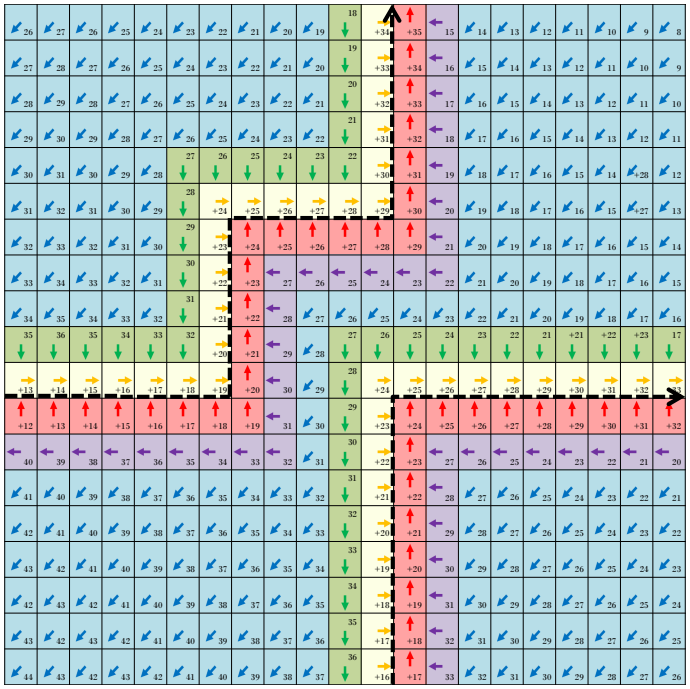


Figure 4.6: The embedding of the line near a crossover.

gets the default blue color (see Algorithm 6).

The p-values are easy to compute based on the f-colors. Note that the value of any blue, green, or purple square depends only on its Manhattan distance d from the bottom right corner of the picture, i.e., the value is $2L - 10 - d$. Moreover, the value of any yellow or red square depends only on the number of steps from the base of the staircase up to that square. Hence, it is independent of the exact shape of the staircase and it can also be expressed in terms of the Manhattan distance from the bottom left corner of the picture, i.e., the value is $M - 1 + d$ (see Algorithm 7).

Finally, we set the parameter according to the construction. The actual values are determined by the analysis in the proof. In terms of correctness, we only need to verify that the parameter can be described in $\mathsf{poly}(n)$ bits. Concretely, we set $\lambda = 2^{3n}$, $\delta = \frac{1}{10\lambda n}$, and $\varepsilon = \frac{\delta}{10}$ (and recall that $L = 10(2^{2n} + 1)$).

4.5.1 Proof of Theorem 12

We prove the correctness of the above reduction. First, we show that the result of the reduction is a valid instance of Continuous-Local-Optimum. That is, we show that f is 4-Lipschitz and p is 2^{3n}-Lipschitz. Second, we show that any local optimum of the resulting instance of Continuous-Local-Optimum can efficiently be transformed to a solution for the EOML instance.

The proof that f and p are Lipschitz follows from the way they are defined, i.e.,

by Cartesian interpolation. The proof is standard and rather technical, and thus we defer it to the appendix (see Lemma 4 and Lemma 5 in Section 4.7.1). The core of our proof is to demonstrate that we did not introduce any local optima far from any end of a line. That is, we show that any unit square that contains a local optimum can be efficiently transformed into a solution of the EOML instance.

We consider the picture to be divided into square templates, each of size 1×1 such that its corners are at the middle of the squares in the picture. The different types of templates correspond to the different colors assigned to the four squares covering the template. Chen and Deng [CD09] showed that no template composed of only two colors contains a fixed point. We show a stronger statement: such templates do not contain a local optimum either.

Lemma 3. *There is no ε-local optimum in any template that contains only two colors.*

Proof. For any point z we define $\Delta = \Delta(z) = p(f(z)) - p(z)$ to be the improvement in p-value after taking a single step in f originating from z. For each template we prove that for any point z inside this template the improvement is greater than ε, i.e.,

$$\Delta(z) = p(f(z)) - p(z) > \varepsilon.$$

Given a point in a specific template, we consider its coordinates relative to the template (the x axis is positioned along the bottom of the square template, so that the origin is at the bottom-left corner, and the top-right corner of the template has coordinates (1,1)). We prove that $\Delta(z) > \varepsilon$ for all $z = (x, y) \in [0, 1]^2$. Let $a, b, c, d \in \mathbb{N}$ be the values of p on the top-left, top-right, bottom-left, and bottom-right corners respectively. We define $p_\perp(x) = xd + (1 - x)c$, and $p_\top(x) = xb + (1 - x)a$. Then, we can express p in terms of $p_\perp$ and $p_\top$ as

$$\begin{aligned} p(x, y) &= y p_\top(x) + (1 - y) p_\perp(x) \\ &= (-a + b + c - d)xy + (a - c)y + (-c + d)x + c \\ &= a'xy + b'y + c'x + d', \end{aligned}$$

where $a' = -a + b + c - d$, $b' = a - c$, $c' = -c + d$, and $d' = c$. Recall that $f(z) = f(x, y) = z + \delta d(x, y) = (x + \delta d_x(x, y), y + \delta d_y(x, y))$, where d_x (respectively d_y) is the x component (respectively the y component) of the displacement function d. Thus, we can express $\Delta(z)$ as

$$\begin{aligned} \Delta(x, y) &= p(f(x, y)) - p(x, y) \\ &= p(x + \delta d_x(z), y + \delta d_y(z)) - p(x, y) \\ &= \delta(c' d_x(z) + b' d_y(z) + a'(\delta d_x(z) d_y(z) + d_y(z)x + d_x(z)y)). \end{aligned}$$

Since f is also defined by Cartesian interpolation, we can plug in the specific values for p and for f at the four corners of the template to the above expression, and we get an explicit formula for $\Delta(x, y)$ in the template.

We begin by considering all the points $z \in [0,1]^2$ such that $f(z) \in [0,1]^2$, i.e., points such that their image under f lands in the same template. In this case we use the explicit formula for $\Delta(z)$ to directly argue that $\Delta(z) > \varepsilon$. In the rest of this part of the proof we go over all the possible templates that can occur in our reduction. For each template we plug in the corner values of f and p for the template, derive the explicit formula for $\Delta(x,y)$, and prove that it is always larger than $\varepsilon' = 2\varepsilon$. The complete analysis of all the different templates is given in Section 4.7.2.

It remains to show that also the points that are brought by f outside their template do not constitute local optima. Notice that any such point must be in a distance of at most δ from the border of the template since f makes steps of length at most δ (in the maximum norm). Recall that by considering the points that under f stay inside the template, we have already shown that all the points that lie inside the template at least δ-far from its border are not local optima. We use this fact together with the following claim to prove that also none of the points δ-close to the boundary of the template constitute a local optimum.

Claim 4. *Let $f\colon \mathbb{R}^3 \to \mathbb{R}^3$ and $p\colon \mathbb{R}^3 \to \mathbb{R}$ be λ_f-Lipschitz and λ_p-Lipschitz respectively, and let $\varepsilon, \gamma > 0$. Let $R \subset \mathbb{R}^3$ be such that for all $z \in R$ it holds that*

$$p(f(z)) - p(z) > \varepsilon.$$

Then for all $z \in R_\gamma = \{x : \exists y \in R, |x-y| \leq \gamma\}$ it holds that

$$p(f(z)) - p(z) > \varepsilon - \lambda_p\lambda_f\gamma - \lambda_p\gamma.$$

Proof. Let $z \in R_\gamma$. By the definition of R_γ, there exists some $z' \in R$ at most γ-far from z, i.e., such that $|z - z'| \leq \gamma$. First, we derive the following bound on the difference in the p-values at the images of z and z' under f:

$$|p(f(z)) - p(f(z'))| \leq \lambda_p|f(z) - f(z')| \leq \lambda_p\lambda_f|z - z'| \leq \lambda_p\lambda_f\gamma,$$

where we used the λ_p-Lipschitz continuity of p, the λ_f-Lipschitz continuity of f, and the bound on the distance of z and z'. Since no $z' \in R$ can be an ε-approximate local optimum, we get that

$$p(f(z)) \geq p(f(z')) - \lambda_p\lambda_f\gamma > p(z') + \varepsilon - \lambda_p\lambda_f\gamma. \tag{4.5.1}$$

Additionally, from the λ_p-Lipschitz continuity of p and the bound on distance of z and z' we get that $|p(z) - p(z')| \leq \lambda_p|z - z'| \leq \lambda_p\gamma$. Thus, $p(z') \geq p(z) - \lambda_p\gamma$, which we can plug into Equation (4.5.1) and we get that

$$p(f(z)) > p(z') + \varepsilon - \lambda_p\lambda_f\gamma \geq p(z) - \lambda_p\gamma + \varepsilon - \lambda_p\lambda_f\gamma,$$

as claimed. □

We have proved that any point z within a template that is at least δ-far from the border satisfies $\Delta(z) > \varepsilon' = 2\varepsilon$. Thus, we can define R to be all such points that are at least δ-far from the border of the template, and using Claim 4 we get that for *all* points z in the template it holds that

$$\Delta(z) > \varepsilon' - \lambda_p \lambda_f \delta - \lambda_p \delta \geq 2\varepsilon - 5\lambda_p \delta \geq 2\varepsilon - 1/n \geq \varepsilon.$$

Which concludes the proof of Lemma 3. □

Extracting a Solution. To finish the proof of Theorem 12, we show how to translate any solution to the resulting Continuous-Local-Optimum instance to a solution for the original End-of-Metered-Line instance. Suppose that we are given an arbitrary local optimum $(X, Y) \in [0, L]^2$ inside the picture (there are no local optima in the frame by our construction). We parse the coordinates of (X, Y) as $((b_x, s_x, x), (b_y, s_y, y))$, where $(b_x, b_y) \in [2^n] \times [2^n]$ is the block, $(s_x, s_y) \in [10 \cdot 2^n] \times [10 \cdot 2^n]$ is the square inside this block and $(x, y) \in [0, 1]^2$ is the relative position of (X, Y) within this square. We have proved that all templates containing only two colors contain no local optimum (Lemma 3). Thus, any local optimum must be inside a template in the vicinity of the beginning or the end of some line. Recall that each block contains a number of squares that is a multiple of 10, and thus let $s'_x = \lfloor s_x/10 \rfloor$ and $s'_y = \lfloor s_y/10 \rfloor$. Our main claim is that if (X, Y) is a local optimum in the Continuous-Local-Optimum instance then either s'_x or s'_y is a solution for the End-of-Metered-Line instance. Since verifying a solution can be done efficiently, given that the claim is true, extracting a solution is easy. We elaborate more on why the claim is true.

First, consider the simplest case where the implicit graph defined by S and P from the End-of-Metered-Line instance contains a single line, and some additional cycles. Moreover, V gives the correct incremental values for vertices on the line, and it assigns the zero value to any other vertex. Let w be the sink of this line, i.e., the only solution to the End-of-Metered-Line instance. In this case, the reduction results in an instance with a single staircase (cf. Figure 4.5) with exactly one local optimum that occurs at block $(\mathsf{V}(w), \mathsf{V}(w) + 1)$ at square $(10 \cdot \mathsf{S}(w), 10 \cdot w)$, and thus the claim follows.

Now consider a graph with an additional line (or many additional lines), where V outputs the values according to the distance of the vertices on the additional line with respect to its actual source. Then we will have two (or more) staircases traveling through the blocks. Here comes the importance of the crossover template (cf. Figure 4.6) that ensures two staircases never cross (which would create a local optimum at the crossing point). The only local optima are in the vicinity of the beginning/end of these staircases, and thus we get the sink/source of the corresponding line, which is a solution to the End-of-Metered-Line instance.

In a more general case, it might be that V gives inconsistent values for vertices on a line. Again, any deviation from the "correct" value (i.e., the exact distance from the

source) will cut the line embedded at that block. Let w be the last vertex on which V answers consistently. Then, there will be a local optimum at block $(\mathsf{V}(w), \mathsf{V}(w)+1)$ at square $(10 \cdot \mathsf{S}(w), 10 \cdot w)$.

In the most general case, we might have collisions in S, collisions in P and have V output inconsistent values for many vertices. Suppose that $\mathsf{S}(x) = \mathsf{S}(x') = y$ and $\mathsf{P}(y) = \mathsf{S}(x)$. Then, x' is a solution (a sink), and indeed we will have a local optimum at block $(\mathsf{V}(x), \mathsf{V}(x)+1)$ at square $(10 \cdot y, 10 \cdot x')$ which can be used to extract x'. Now suppose that $\mathsf{P}(y) = \mathsf{P}(y') = x$ and $\mathsf{S}(x) = y$. Then y' is a solution (a source), and indeed we will have a local optimum at block $(\mathsf{V}(y'), \mathsf{V}(y')-1)$ at square $(10 \cdot y', 10 \cdot x)$. Moreover, if $\mathsf{V}(x) = 1$ and $x \neq 0^n$ then x is a solution and we will have a local optimum at block $(1,1)$ at square $(10 \cdot x, 0)$. In general, the End-of-Metered-Line instance might have many different solutions which will yield many different local optima. However, from coordinates of any such local optimum it is easy to recover the original solution. To illustrate the result of the embedding we give an example for a specific End-of-Metered-Line instance in Figure 4.7.

Normalizing the Domain. Finally, we show how to convert f and p to the appropriate domain and range. First, we scale the domain of the two functions to be the unit square, and we define $f'(z) = \frac{1}{T} \cdot f(Tz)$ and $p'(z) = \frac{1}{T} \cdot p(Tz)$ for $T = 4L$.

Claim 5. *Let $f\colon \mathbb{R}^n \to \mathbb{R}^m$ be a λ-Lipschitz function. Then for any $T > 0$ the function $f'(z) = \frac{1}{T} \cdot f(Tz)$ is λ-Lipschitz.*

Proof. We get by a simple calculation that

$$\begin{aligned} |f'(z) - f'(z')| &= \left| \frac{1}{T} \cdot f\left(Tz\right) - \frac{1}{T} \cdot f\left(Tz'\right) \right| \\ &= \frac{1}{T} \left| f\left(Tz\right) - f\left(Tz'\right) \right| \leq \frac{1}{T} \lambda \left| Tz - Tz' \right| \\ &= \lambda |z - z'|, \end{aligned}$$

as claimed. □

Recall that $L = 10(2^{2n}+1)$. Since f is 4-Lipschitz, by Claim 5, we get that f' is 4-Lipschitz. Similarly, since p is 2^{3n}-Lipschitz we get that p' is 2^{3n}-Lipschitz. Thus, for the final construction we set $\lambda' = 2^{3n}$, $\delta' = \delta/T$ and $\varepsilon' = \varepsilon/T$, such that any ε-local optimum with respect to f and p is an ε'-local optimum with respect to f' and p'.

Finally, notice that f' and p' are defined over $[0,1]^2$ and not over $[0,1]^3$ as in the definition of Continuous-Local-Optimum. They can be easily extended to $[0,1]^3$ by copying over the third dimension. Namely, define $f''(x,y,z) = (f'(x,y), z)$ and $p''(x,y,z) = p'(x,y)$. It is easy to see that this transformation does not change the Lipschitz constants of the functions. This concludes the proof of Theorem 12.

4.6 On the Hardness of End-of-Metered-Line

4.6.1 Query Complexity Lower Bound

In this section, we show an exponential query complexity lower bound for End-of-Metered-Line. This, in turn, shows a query complexity lower bound for CLS. In more general terms, we show that finding a local optimum even in a continuous domain is exponentially hard.

The Query Model. Let $\Pi(n)$ be a search problem defined by a function parametrized by n. In the query model, an algorithm is given oracle access to the function, and we measure the number of oracle queries performed by the algorithm, whereas its running time may be unbounded. We denote by $\mathrm{QC}_p(\Pi(n))$ the number of queries required for any randomized algorithms to solve $\Pi(n)$ with probability at least p, and we denote $\mathrm{QC}(\Pi(n)) = \mathrm{QC}_{2/3}(\Pi(n))$.

We consider the End-of-Metered-Line in the query model. Formally, let $\mathrm{EOML}(n)$ be given by an oracle that on query $v \in \mathcal{B}^n$ outputs a triple consisting of the predecessor of v, the successor of v, and the value of v. That is, the EOML oracle will answer for any vertex $v \in \mathcal{B}^n$ with the triple $(\mathsf{S}(v), \mathsf{P}(v), \mathsf{V}(v)) \in \mathcal{B}^n \times \mathcal{B}^n \times \mathbb{N}$. The goal is to find a vertex v that satisfies one of the three conditions for a solution of End-of-Metered-Line (cf. Definition 32). We show that the randomized query complexity of $\mathrm{EOML}(n)$ is exponential in n.

Theorem 13. $\mathrm{QC}(\mathrm{EOML}(n)) = \Omega\left(2^{n/2}/\sqrt{n}\right)$.

In order to show a lower bound for probabilistic algorithms, by Yao's minmax principle it is enough to show that there exists a distribution over instances such that every *deterministic* algorithm fails with high probability.

For the problem Local-Search, i.e., finding a (non-continuous) local optimum on the n-dimensional Boolean hypercube, Zhang [Zha09] gave a tight query complexity lower bound of $\Theta(2^{n/2}\sqrt{n})$. In this section we show how to adapt his techniques to get a matching lower bound for End-of-Metered-Line. In particular, Zhang [Zha09] defined a query complexity problem Path, and showed that

$$\mathrm{QC}(\textsc{Path}) \leq 2\mathrm{QC}(\textsc{Local-Search}).$$

Finally, to argue the tight query complexity lower bound for Local-Search, Zhang proved the following lower bound for Path (matching upper bound for Local-Search was given by Aldous [Ald83]):

Claim 6 ([Zha09]). $\mathrm{QC}(\textsc{Path}) = \Omega\left(2^{n/2} \cdot \sqrt{n}\right)$.

The Path Game. We begin by describing the Path problem that asks to find an endpoint on a random self-avoiding path over the Boolean hypercube. In particular, consider a decomposition of the Boolean hypercube $\mathcal{B}^n$ as $\mathcal{B}^m \times \mathcal{B}^{n-m}$, where

$m = \lfloor (n + \log n)/2 \rfloor$. We construct a self-avoiding path over $\mathcal{B}^n$ that corresponds to following a random walk on the first component $\mathcal{B}^m$, while using the second component $\mathcal{B}^{n-m}$ to keep a step-counter for the random walk in order to avoid self-loops.

Formally, fix any Hamiltonian path of length $2^{n-m} - 1 = 2T + 1$ over $\mathcal{B}^{n-m}$. We denote the vertices on the Hamiltonian path as

$$(z_{0,0}, z_{1,0}, z_{1,1}, z_{2,1}, z_{2,2}, \ldots, z_{T,T}).$$

Next, we build a random walk over $\mathcal{B}^m$ that starts at an arbitrary vertex x_0 (e.g., $x_0 = 0^m$), and proceeds for T steps as follows. For all $i \in [T]$, we choose $x_i \in \mathcal{B}^m$ by flipping a random coordinate in x_{i-1}. Finally, given the random walk $(x_0, x_1, \ldots, x_T)$ and the Hamiltonian path $(z_{0,0}, z_{1,0}, \ldots, z_{T,T})$, we define a path X over $\mathcal{B}^n = \mathcal{B}^m \times \mathcal{B}^{n-m}$ by making a step in the random walk and two subsequent steps on the Hamiltonian path, i.e.,

$$X = ((x_0, z_{0,0}), (x_1, z_{0,0}), (x_1, z_{1,0}), (x_1, z_{1,1}), \ldots, (x_T, z_{T-1,T-1}), (x_T, z_{T,T-1}), (x_T, z_{T,T})).$$

The PATH problem is given by an oracle access to the above path X, i.e., the PATH oracle answers 1 for any vertex $v \in \mathcal{B}^n$ that lies on the path X and 0 for any point that lies off the path X. The goal is to find the endpoint of the path X, i.e., the vertex $v = (x_T, z_{T,T})$.

We show that any query complexity lower bound for the PATH problem implies a query complexity lower bound for END-OF-METERED-LINE.

Claim 7. $\mathrm{QC}(\textsc{Path}) \leq n\mathrm{QC}(\mathrm{EOML})$.

Proof. For any PATH oracle, we implement an equivalent EOML oracle.

EOML-Oracle(v)**:**

1. Query $\textsc{Path}(v)$ to learn whether v lies on the path X.
2. If v lies off the path X, output $(v, v, 0)$.
3. If $v = (x, z_{T,T})$ then output $((x, z_{T,T-1}), (x, z_{T,T}), 3T + 1)$.
4. If $v = (x, z_{k,k})$ for some $k \in [T] \cup \{0\}$ and $\textsc{Path}(x, z_{k+1,k}) = 0$ then:
   (a) For all x' that differ from x in a single coordinate, query $\textsc{Path}(x', z_{k,k})$ to find $(x_{k+1}, z_{k,k})$, the next vertex on the path X.
   (b) If $k = 0$ then output $((x, z_{0,0}), (x_1, z_{0,0}), 1)$.
   (c) Otherwise output $((x, z_{k,k-1}), (x_{k+1}, z_{k,k}), 3k + 1)$.
5. If $v = (x, z_{k,k})$ for some $k \in [T] \cup \{0\}$ and $\textsc{Path}(x, z_{k,k-1}) = 0$ then:
   (a) For all x' that differ from x in a single coordinate, query $\textsc{Path}(x', z_{k,k})$ to find $(x_{k-1}, z_{k,k})$, the previous vertex on the path X.

(b) Output $((x_{k-1}, z_{k,k}), (x, z_{k+1,k}), 3k-1)$.

6. If $v = (x, z_{k,k-1})$ for some $k \in [T]$ then output $((x, z_{k-1,k-1}), (x, z_{k,k}), 3k)$.

The unique solution to the new EOML instance is the local optimum at the vertex $v = (x_T, z_{T,T})$, which constitutes also a solution to the original Path problem. Notice that to answer each End-of-Metered-Line query we perform at most $m + 2 \leq n$ queries to the Path oracle. Thus, we get that $\mathrm{QC}(\textsc{Path}) \leq n\mathrm{QC}(\mathrm{EOML})$, as claimed. □

By combining Claim 7 with Claim 6 we get the following query complexity lower bound for End-of-Metered-Line:

$$\mathrm{QC}(\mathrm{EOML}) \geq 2^{n/2}/\sqrt{n},$$

as claimed in Theorem 13.

Query Complexity of Continuous-Local-Optimum. To get the exponential query complexity lower bound for Continuous-Local-Optimum, we combine the above lower bound for End-of-Metered-Line with our reduction from Theorem 12. It follows that one must perform $2^{n/2}/\sqrt{n}$ queries to get accuracy ε, where in the reduction we set $\varepsilon \approx 2^{-5n}$ (up to a polynomial factor). In other words, to solve Continuous-Local-Optimum with n-digits of precious (i.e., up to accuracy $\varepsilon = 2^{-n}$) one must perform approximately $2^{n/10} = 2^{\Omega(n)}$ queries.

Alternative Approach Based on Lower Bounds for PPAD. We note that we could have directly proved exponential query complexity lower bound for CLS by combining our reduction from End-of-Metered-Line to Continuous-Local-Optimum from Theorem 12 with the previous works on query complexity of computing approximate Brouwer fixed points (e.g., [HPV89, LNNW95]). However, our query complexity lower bound for End-of-Metered-Line presented in Theorem 13 is a stronger result, since End-of-Metered-Line is not complete for CLS, and hence we give a lower bound for potentially easier problem than continuous local search. For completeness, we sketch below the direct approach for showing black-box hardness for CLS.

In the case of Brouwer fixed points for functions over the unit square, Hirsch, Papadimitriou and Vavasis [HPV89] showed an exponential query complexity lower bound for any deterministic algorithm treating the function as a black-box. Their work was based on reducing the problem of finding an end of a staircase traversing square grid with N^2 points to finding a fixed point of a continuous function over the unit square. Specifically, they showed that any query complexity lower bound for the staircase problem with N^2 points implies a matching query complexity lower bound for finding $\frac{1}{N}$-approximate Brouwer fixed points. They showed that the deterministic query complexity of the staircase problem is $\Theta(N)$ (Lovász et al. [LNNW95] showed a randomized query complexity lower bound of $\Omega(N^{\frac{1}{3}})$), which implies an exponential

query complexity lower bound for finding approximate Brouwer fixed points and thus for PPAD.

Similarly to [HPV89], our reduction from End-of-Metered-Line to the problem Continuous-Local-Optimum also provides an embedding of any staircase into the unit square resulting in a continuous function f that has a fixed point only at the end of the staircase. The additional property of our construction is that we are able to define a valuation function p assigning to any point of the unit square a real value such that any local optimum of p under f is in the vicinity of the fixed point of f. Moreover, the p-value at any point x is computed based solely on its coordinates and its f-value. In other words, any oracle call to p does not provide any additional information since it can be simulated based on oracle calls to f. Any algorithm that can solve our continuous local optimum instance given oracle access to f and p can solve it only given the oracle access to f. Hence, our staircase embedding can readily be used in place of the embedding of Hirsch et al. [HPV89] to obtain a lower bound for CLS from any query complexity lower bound for the above grid problem.

4.6.2 Cryptographic Hardness of End-of-Metered-Line

In this section, we show that under cryptographic assumptions End-of-Metered-Line (EOML) is hard. Formally, we prove the following theorem.

Theorem 14. *Assume there exist one-way permutations and indistinguishability obfuscation for P/Poly. Then the* End-of-Metered-Line *problem is hard for polynomial-time algorithms.*

Our cryptographic hardness result for End-of-Metered-Line builds upon previous works on cryptographic hardness for PPAD. Recently, Bitanski et al. [BPR15] were able to show that assuming one-way functions (see Definition 9) and indistuinguishability obfuscation (both with subexponential security) the End-of-Line problem is hard. Their proof followed two main steps. First, they defined a problem called Sink-of-Verifiable-Line (motivated by the work of Abbott, Kane and Valiant [AKV04]), and they showed that Sink-of-Verifiable-Line is hard under the above cryptographic assumptions. Second, they gave a reduction from Sink-of-Verifiable-Line to End-of-Line yielding the conclusion that End-of-Line is hard under the same assumptions.

We start by giving an overview of the original reduction to End-of-Line and then describe our modifications. The following formal definition of Sink-of-Verifiable-Line was given in Bitanski et al. [BPR15].

Definition 33 (Sink-of-Verifiable-Line [BPR15])**.** *An instance* $(\mathsf{S}, \mathsf{V}, x_s, T)$ *consists of a source* $x_s \in \mathcal{B}^n$*, a target index* $T \in [2^n]$*, and a pair of circuits* $\mathsf{S}\colon \mathcal{B}^n \to \mathcal{B}^n$*,* $\mathsf{V}\colon \mathcal{B}^n \times [T] \to \mathcal{B}$*, with the guarantee that, for all* $(x, i) \in \mathcal{B}^n \times [T]$*, it holds that* $\mathsf{V}(x, i) = 1$ *iff* $x = x_i := \mathsf{S}^{i-1}(x_s)$*, where* $x_1 := x_s$*. A string* $w \in \mathcal{B}^n$ *is a valid witness iff* $\mathsf{V}(w, T) = 1$*.*

As discussed in the previous works [AKV04, BPR15, GPS16], the above problem is not necessarily total without the promise about the behavior of the verification circuit V. In particular, V might just for all $x \in \mathcal{B}^n$ reject any pair (x, T) and such behavior cannot be efficiently checked.[6] Notice that there is no need for an explicit source vertex x_s in the above definition. The source can be without loss of generality labeled 0^n, and we use this convention from now on as it is standard in definitions of other problems inside TFNP (see Section 2.5).

Warm-up Reduction to End-of-Line. Consider a Sink-of-Verifiable-Line instance denoted $(\mathsf{S}, \mathsf{V}, T)$. In order to reduce it to an End-of-Line instance it is necessary to implement the predecessor circuit P'. Notice that it is easy to construct the predecessor circuit in an *inefficient* way. One can simply modify the labels of the vertices to contain the entire history of the previous steps on the line. That is, we construct circuits S' and P' such that if $0^n \to x_2 \to x_3 \to \cdots \to x_T$ is a line according to S then there is a line according to S' of the form

$$0^n \to (0^n, x_2) \to (0^n, x_2, x_3) \to (0^n, x_2, x_3, x_4) \to \cdots \to (0^n, x_2, \ldots, x_T) \,.$$

Given these labels, implementing the predecessor is easy: simply remove the last element from the label. However, the obvious issue of this transformation is that the size of the labels becomes eventually exponentially large which prevents it from being a polynomial reduction. To reduce the size of the labels to give an efficient construction of the predecessor circuit, [AKV04, BPR15] utilized techniques used for implementing *reversible computation* [Ben89] where only a small number of states is stored in order to be able to revert previous steps in the computation. The general approach can be explained via a simple *pebbling game* that we describe next.

The Pebbling Game. There are n pebbles that can be placed on positions indexed by positive integers. The rules of the game are as follows: a pebble can be placed in or removed from position i if and only if either there is a pebble in position $i-1$ or $i = 1$. The goal of the game is to place a pebble in position $2^n - 1$.

As shown by Chung, Diaconis and Graham [CDG01], the optimal efficient strategy achieves the goal of the game in a recursive manner. The main idea is that since the rules for placing and removing pebbles are symmetric, it is always possible to reverse any sequence of moves. Suppose then there is a way to get to $2^{n-1} - 1$ using $n-1$ pebbles. Then, place a pebble at 2^{n-1}. Next, free the first $n-1$ pebbles by reversing the original sequence of moves performed in the first part. Finally, perform the same sequence starting from 2^{n-1}. This strategy will end with a pebble at position $2^n - 1$.

The predecessor circuit in the reduction from Sink-of-Verifiable-Line to End-of-Line is implemented by simulating the optimal strategy in the pebbling game. Each vertex

[6] In fact, the hardness results of [BPR15, GPS16] strongly exploit this fact. Both works show that under cryptographic assumptions there exist instances of Sink-of-Verifiable-Line that are computationally indistinguishable from instances that do not obey the promise on behavior of V and have no solutions.

has a label representing the states of the n pebbles. The efficient strategy demonstrates that by storing only n intermediate states we can implement S' and P' that can traverse an exponential number of steps. The resulting End-of-Line instance $(\mathsf{S}', \mathsf{P}')$ corresponds to a graph with a single line traversing the sequence of all the configurations visited by the optimal pebbling strategy. In particular, every vertex corresponding to an intermediate state of the pebbling strategy is followed by the subsequent state, and the final step of the pebbling strategy is a self-loop under S'. Any state describing an illegal configuration of the pebbling game is defined to be a self loop both under S' and P'. Therefore, the resulting instance has a unique solution, a sink that identifies a solution to the original Sink-of-Verifiable-Line instance. For completeness, the pseudocode of S' and P' are given in Section 4.7.4.

Our Reduction. In order to give a reduction to End-of-Metered-Line, we must provide not only the predecessor circuit P' but also the valuation circuit V' that meters the line starting at 0^n. Recall the inefficient construction of the predecessor circuit that uses the labels of the vertices to store the complete history of line. Given these labels the implementation of V' is simple: the distance of a vertex from the start of the line is exactly the number of elements in its label. Our crucial observation is that even in the efficient reduction based on simulating the pebbling game the predecessor circuit does not come at the expense of the verification circuit V from the Sink-of-Verifiable-Line instance.

Since the label of each vertex corresponds to a configuration of the pebbling game, the value we assign to each vertex is the number of steps performed in the optimal strategy until reaching the specific configuration corresponding to its label. To construct the valuation circuit V' we need to show that, given a configuration, it is possible to efficiently compute where exactly in the pebbling strategy we are without simulating the entire game. We start by computing the total number of steps in the efficient pebbling strategy.

Claim 8. *Let $g(n)$ be the number of steps performed in the pebbling game by the optimal strategy until a pebble is put in position $2^n - 1$. Then the following hold:*

1. $g(n) = 3g(n-1) + 1$,
2. $g(n) = \frac{3^n - 1}{2}$.

Proof. We begin by proving the first item. By the definition of the pebbling strategy, we first perform $g(n-1)$ steps to put a pebble at position $2^{n-1} - 1$. Then we put the n^{th} pebble at position 2^{n-1}. Then, we reverse the initial $g(n-1)$ moves to free all the first $n-1$ pebble. Finally, we use the same strategy again starting from $2^{n-1} + 1$, and perform another $g(n-1)$ steps to get to position $2^n - 1$. Hence, the total number of steps is $3g(n-1) + 1$.

The second item follows by the recursive definition given in the first item. We have that

$$g(n) = 3g(n-1)+1 = 3^2 g(n-2)+3+1 = \ldots = 3^{n-1}+3^{n-2}\ldots+3+1 = \frac{3^n-1}{2},$$

as claimed. □

The circuit V' can be built based on the recursive expression for $g(n)$ from Claim 8. Note that $g(n)$ was computed as an addition of three phases: (1) getting to the half, (2) reversing the first half, and (3) completing the second half. To check if phase 1 has finished, we check if position 2^{n-1} is occupied. This will indicate that we have already performed at least the initial $g(n-1)+1$ steps. Otherwise, phase 1 has not be finished and we return a recursive call with the first half of the board. Next, to check if phase 2 (reversing the computation) we check whether all the pebbles are to the right of position 2^{n-1}. If there is even 1 pebble of the left, then we are still in phase 2. The number of steps taken in this phase is $g(n-1)$ minus a recursive call with $n-1$ on the first half of the board. If we are in phase 3 then we have performed $2g(n-1)+1 = 3^n$ steps plus the number of steps currently in the phase which is computed by a recursive call to the second half of the board.

For completeness, any vertex with label describing an illegal configuration of the pebbling game gets a default value 0. The formal description of this procedure is given in Algorithm 1.

Correctness of our Reduction. The key point of our reduction is that the resulting End-of-Metered-Line instance $(\mathsf{S}', \mathsf{P}', \mathsf{V}')$ has a single line corresponding to the line of the SVL instance, and all other nodes are self-loops. This restricts the set of possible solutions of $(\mathsf{S}', \mathsf{P}', \mathsf{V}')$ as follows. There is a unique solution of the first type corresponding to the sink at the last state of the pebbling game. There is no solution of the second type, since the only vertex that is assigned value 1 by V' is the initial source. The unique solution of the first type is also the unique solution of the third type, since the only vertex satisfying $0 \neq \mathsf{V}'(x) \neq \mathsf{V}'(\mathsf{S}'(x)) - 1$ is the sink corresponding to the last state of the pebbling game. Moreover, from the unique solution it is easy to extract a vertex x that is a solution to the original SVL instance. In particular, one of the pebbles at the final configuration of the pebbling strategy is exactly in a position corresponding to an x such that $\mathsf{V}(x, T) = 1$.

function $\mathsf{V}'(N = (u_1, \ldots, u_n))$
  if all $u_1, \ldots, u_n$ are valid states **then**
    return $1+\mathsf{V}'_n(1, 2^n - 1, n, N)$
  else
    return 0
  end if
end function

function $\mathsf{V}'_j(\mathsf{start}, \mathsf{end}, N = (u_1, \ldots, u_n))$
  if $j = 0$ **then return** 0
  end if
  $\mathsf{mid} \leftarrow \mathsf{start} + 2^{j-1} - 1$
  if position mid is free **then**
    return $\mathsf{V}'_{j-1}(\mathsf{start}, \mathsf{mid} - 1, N)$
  else if there is a pebble at position $i \in [\mathsf{start}, \mathsf{mid} - 1]$ **then**
    return $3^{j-1} - \mathsf{V}'_{j-1}(\mathsf{start}, \mathsf{mid} - 1, N)$
  else return $3^{j-1} + \mathsf{V}'_{j-1}(\mathsf{mid} + 1, \mathsf{end}, N)$
  end if
end function

Algorithm 1: The algorithm V'.

4.7 Chapter Appendix

In this section, we give proofs deferred from Section 4.5.1.

4.7.1 Proof of f and p being Lipschitz

Lemma 4. *The function f described in the reduction in Section 4.5 is λ_f-Lipschitz, for $\lambda_f = 4$.*

Proof. By the definition of f we have that $f(z) = z + \delta d(z)$, where $d(z)$ is the displacement function defined by interpolation of the different displacements at the centers of squares surrounding z. First, we show that $d(z)$ is λ_d-Lipschitz for $\lambda_d = 4$. Consider $d(z)$ on a single dimension, that is, fix one of the coordinates to any constant. Suppose that $x \in [a, b]$ where a, b are two adjacent center points with displacement values d_a, d_b. Then the function $d_{a,b}(x) = (x - a)d_b + (b - x)d_a$ is the function $d(z)$ on

this region over one dimension. Suppose that $x, x' \in [a, b]$. Then we get that

$$\begin{aligned}|d_{a,b}(x) - d_{a,b}(x')| &= |(x-a)d_b + (b-x)d_a - (x'-a)d_b - (b-x')d_a| \\ &= |(x-x')d_b + (x'-x)d_a| \\ &\leq |d_b - d_a| \cdot |x - x'| \\ &\leq 2|x - x'|,\end{aligned}$$

where we used that the maximum norm of both displacement values d_a and d_b is one. Thus, $d_{a,b}(x)$ is λ_d-Lipschitz on $[a, b]$ for $\lambda_d = 2$. Now suppose that $x \in [b, c]$ and $x' \in [a, b]$. Then we get that:

$$\begin{aligned}|d_{b,c}(x) - d_{a,b}(x')| &\leq |d_{b,c}(x) - d_{b,c}(b)| + |d_{a,b}(b) - d_{a,b}(x')| \\ &\leq 2\delta|x - b| + 2|b - x'| \\ &= 2|x - x'|.\end{aligned}$$

Now suppose that $x \in [a, b]$ and $x' \in [c, d]$ where $b < c$. Then we get that

$$|d_{a,b}(x) - d_{c,d}(x')| \leq 2 \leq 2|x - x'|.$$

Altogether, we got that for any a, b it holds that $d_{a,b}$ is 2-Lipschitz. Going back to two dimensions we get that for the four center points a, b, c, d representing the top-left, top-right, bottom-left, and bottom-right corners of a template respectively we have

$$d(x, y) = d_{a,b,c,d}(x, y) = (y - c_y)d_{c,d}(x) + (a_y - y)d_{a,b}(x).$$

Thus, we get that

$$\begin{aligned}|d(x, y) - d(x', y')| &\leq |d(x, y) - d(x, y')| + |d(x, y') - d(x', y')| \\ &\leq 2|y - y'| + 2\delta|x - x'| \\ &\leq 4 \max\{|x - x'|, |y - y'|\} \\ &\leq 4|(x, y) - (x', y')|.\end{aligned}$$

We can now compute the Lipschitz constant of f:

$$\begin{aligned}|f(z) - f(z')| &= |z + \delta d(z) - z' - \delta d(z')| \\ &\leq |z - z'| + \delta|d(z) - d(z')| \\ &\leq |z - z'| + 4\delta|z - z'| \\ &\leq 4|z - z'|,\end{aligned}$$

which concludes the proof. □

We now establish the Lipschitz continuity of p.

Lemma 5. *The function p described in the reduction in Section 4.5 is λ_p-Lipschitz, for $\lambda_p = 2^{3n}$.*

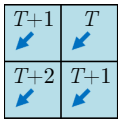
Proof. The analysis of p is simpler. The square with the highest value is at the end of the line, and its value is at most $4L$. The square with the smallest value is 0. The distance between 2 square is 1, and between the square we p is defined by linear interpolation. Therefore, is suffices to bound $|p_\perp(x) - p_\perp(x')|$ for $x, x' \in [0,1]$ such that $p_\perp(0) = 0$ and $p_\perp(1) = 4L$. For these values we get that

$$|p_\perp(x) - p_\perp(x')| \leq |x \cdot 4L - x' \cdot 4L| = 4L|x - x'| \leq 2^{3n}|x - x'|,$$

where we use the fact that $L = 10(2^{2n} + 1)$. □

4.7.2 Analysis of Templates from Lemma 3

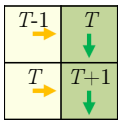
Case 1. The template is of the form:



for some $T \in \mathbb{N}$. Plugging these values for the formula for Δ we get that

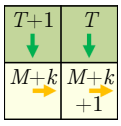
$$\Delta = (1 + 3x(1 - y))\delta \geq \delta > \varepsilon'.$$

Case 2. The template is of the form:



for some $T \in \mathbb{N}$. Plugging these values for the formula for Δ we directly get that $\Delta = \delta > \varepsilon'$.

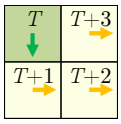
Case 3. The template is of the form:



for some $T, k \in \mathbb{N}$ such that $M + k > T + 1$. It suffice to prove for the case where $M + k = T + 2$. For these values we get that

$$\begin{aligned}
\Delta &= \delta\left(1 + 2y^2(1 - \delta) + 2y(-1 + x + \delta)\right) \\
&= \delta(1 + 2y(y(1 - \delta) + (-1 + x + \delta))) \\
&\geq \delta(1 + 2y((1 - \delta) + (-1 + x + \delta))) \\
&= \delta + 2xy\delta \geq \delta > \varepsilon'.
\end{aligned}$$

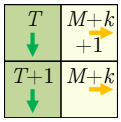
Case 4. The template is of the form:



for some $T \in \mathbb{N}$. Plugging these values for the formula for Δ we that

$$\begin{aligned}
\Delta &= \delta\left(1+2y\left(1+x^2+x(-1+\delta)-\delta\right)+2(-1+x)y^2(1+(-1+x)\delta)\right)\\
&= \delta\left(1+2y\left(1+x^2+x(-1+\delta)-\delta+(-1+x)y(1+(-1+x)\delta)\right)\right)\\
&\geq \delta\left(1-2xy\delta+2x^2y(1+\delta)\right)\\
&\geq \delta\left(1-2\delta)\right) \geq \delta/2 > \varepsilon'.
\end{aligned}$$

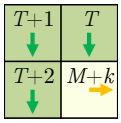
Case 5. The template is of the form:



for some $T, k \in \mathbb{N}$ such that $M+k > T+1$. It suffice to prove for the case where $M+k = T+2$. For these values we get that

$$\begin{aligned}
\Delta &= \delta\left(1+2x(-1+y-\delta)+2x^2(1+\delta)\right)\\
&= \delta(1+2x(-1+x+y-\delta+x\delta))\\
&\geq \delta(1+2x(-1+x-\delta))\\
&\geq \delta(3/4-2x+2x^2)\\
&\geq \delta(3/4-1/2) = \delta/4 > \varepsilon'.
\end{aligned}$$

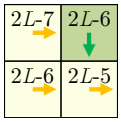
Case 6. The template is of the form:



for some $T, k \in \mathbb{N}$ such that $M+k > T+2$. It suffice to prove for the case where $M+k = T+3$. For these values we get that

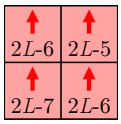
$$\begin{aligned}
\Delta &= \delta\left(1+2x^2(1-y)(-1+(-1+y)\delta)+2x\left(1+y^2+\delta-y(1+\delta)\right)\right)\\
&\geq \delta\left(1+2x\left((1-y)(-1+(-1+y)\delta)+\left(1+y^2+\delta-y(1+\delta)\right)\right)\right)\\
&= \delta(1+2xy(y+\delta-y\delta))\\
&\geq \delta > \varepsilon'.
\end{aligned}$$

Case 7. The template is of the form:



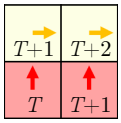
Such a template can be at exactly one position (in the bottom left position of the frame). For these values we directly get that $\Delta = \delta > \varepsilon'$.

Case 8. The template is of the form:



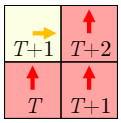
Such a template can be at exactly one position (in the bottom left position of the frame). For these values we get directly that $\Delta = \delta > \varepsilon'$.

Case 9. The template is of the form:



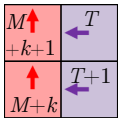
for some $T \in \mathbb{N}$. Plugging these values for the formula for Δ we directly get that $\Delta = \delta > \varepsilon'$.

Case 10. The template is of the form:



for some $T \in \mathbb{N}$. Plugging these values for the formula for Δ we directly get that $\Delta = \delta > \varepsilon'$.

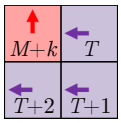
Case 11. The template is of the form:



for some $T, k \in \mathbb{N}$ such that $M + k > T + 2$. It suffice to prove for the case where $M + k = T + 3$. For these values we get that

$$\begin{aligned}\Delta &= \delta\left(1 - 2x^2(-1+\delta) + 2x(-1+y+\delta)\right)\\ &\geq \delta(1 + 2x(-1+x+y+\delta-x\delta))\\ &\geq \delta(1+2x(-1+x))\\ &= \delta(1-2x+2x^2) \geq \delta/2 > \varepsilon'.\end{aligned}$$

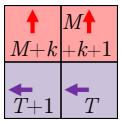
Case 12. The template is of the form:



for some $T, k \in \mathbb{N}$ such that $M + k > T + 2$. It suffice to prove for the case where $M + k = T + 3$. For these values we get that

$$\begin{aligned}\Delta &= \delta\left(1 - 2(-1+x)y^2(-1+(-1+x)\delta) + 2y\left(1 + x^2 + \delta - x(1+\delta)\right)\right)\\ &= \delta\left(1 + 2y\left(1 + x^2 + \delta - x(1+\delta) + (1-x)y(-1+(-1+x)\delta)\right)\right)\\ &\geq \delta\left(1 + 2y\left(1 + x^2 + \delta - x(1+\delta) + (1-x)(-1+(-1+x)\delta)\right)\right)\\ &= \delta\left(1 + 2x^2y(1-\delta) + 2xy\delta\right) \geq \delta > \varepsilon'.\end{aligned}$$

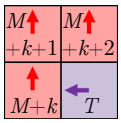
Case 13. The template is of the form:



for some $T, k \in \mathbb{N}$ such that $M + k > T + 1$. It suffice to prove for the case where $M + k = T + 2$. For these values we get that

$$\begin{aligned}\Delta &= \delta\left(1 + 2y(-1+x-\delta) + 2y^2(1+\delta)\right)\\ &= \delta(1 + 2y((-1+x-\delta) + y(1+\delta)))\\ &\geq \delta(1 + 2y((-1+x-\delta) + (1+\delta)))\\ &= \delta + 2xy\delta \geq \delta > \varepsilon'.\end{aligned}$$

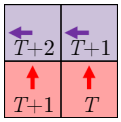
Case 14. The template is of the form:



for some $T, k \in \mathbb{N}$ such that $M + k > T$. It suffice to prove for the case where $M + k = T + 1$. For these values we get that

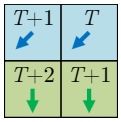
$$\begin{aligned}
\Delta &= \delta\left(1 + 2x\left(1 + y^2 + y(-1+\delta) - \delta\right) + 2x^2(-1+y)(1+(-1+y)\delta)\right) \\
&= \delta\left(1 + 2x\left(\left(1 + y^2 + y(-1+\delta) - \delta\right) + x(-1+y)(1+(-1+y)\delta)\right)\right) \\
&\geq \delta\left(1 + 2x\left(\left(1 + y^2 + y(-1+\delta) - \delta\right) + (-1+y)(1+(-1+y)\delta)\right)\right) \\
&= \delta(1 + 2xy(y - \delta + y\delta)) \geq \delta > \varepsilon'.
\end{aligned}$$

Case 15. The template is of the form:



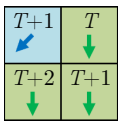
for some $T \in \mathbb{N}$. Plugging these values for the formula for Δ we directly get that $\Delta = \delta > \varepsilon'$.

Case 16. The template is of the form:



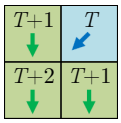
for some $T \in \mathbb{N}$. Plugging these values for the formula for Δ we directly get that $\Delta = (1 + y)\delta \geq \delta > \varepsilon'$.

Case 17. The template is of the form:



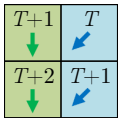
for some $T \in \mathbb{N}$. Plugging these values for the formula for Δ we directly get that $\Delta = (1 + y - xy)\delta \geq \delta > \varepsilon'$.

Case 18. The template is of the form:



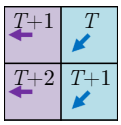
for some $T \in \mathbb{N}$. Plugging these values for the formula for Δ we directly get that $\Delta = \delta(1 + xy) \geq \delta > \varepsilon'$.

Case 19. The template is of the form:



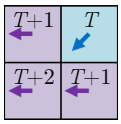
for some $T \in \mathbb{N}$. Plugging these values for the formula for Δ we directly get that $\Delta = (1 + x)\delta \geq \delta > \varepsilon'$.

Case 20. The template is of the form:



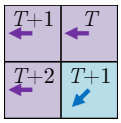
for some $T \in \mathbb{N}$. Plugging these values for the formula for Δ we directly get that $\Delta = (1 + x)\delta \geq \delta > \varepsilon'$.

Case 21. The template is of the form:



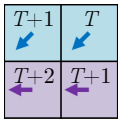
for some $T \in \mathbb{N}$. Plugging these values for the formula for Δ we directly get that $\Delta = \delta(1 + xy) \geq \delta > \varepsilon'$.

Case 22. The template is of the form:



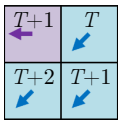
for some $T \in \mathbb{N}$. Plugging these values for the formula for Δ we directly get that $\Delta = (1 + x - xy)\delta \geq \delta > \varepsilon'$.

Case 23. The template is of the form:



for some $T \in \mathbb{N}$. Plugging these values for the formula for Δ we directly get that $\Delta = (1+y)\delta \geq \delta > \varepsilon'$.

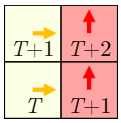
Case 24. The template is of the form:



for some $T \in \mathbb{N}$. Plugging these values for the formula for Δ we directly get that

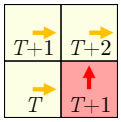
$$\Delta = (2+(-1+x)y)\delta \geq \delta > \varepsilon'.$$

Case 25. The template is of the form:



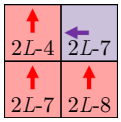
for some $T \in \mathbb{N}$. Plugging these values for the formula for Δ we directly get that $\Delta = \delta > \varepsilon'$.

Case 26. The template is of the form:



for some $T \in \mathbb{N}$. Plugging these values for the formula for Δ we directly get that $\Delta = \delta > \varepsilon'$.

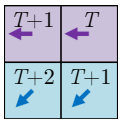
Case 27. The template is of the form:



for some $T \in \mathbb{N}$. Plugging these values for the formula for Δ we directly get that

$$\begin{aligned}\Delta &= \delta\left(3 + 2x\left(-1 + y^2 + y(-1+\delta)\right) - 2x^2y(-1+y\delta)\right)\\ &\geq \delta\left(3 + 2x\left(-1 + y^2 - y\right)\right)\\ &\geq \delta\left(3 - 2.5x\right)\\ &\geq \delta/2 > \varepsilon'.\end{aligned}$$

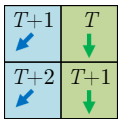
Case 28. The template is of the form:



for some $T \in \mathbb{N}$. Plugging these values for the formula for Δ we directly get that

$$\Delta = (2-y)\delta \geq \delta > \varepsilon'.$$

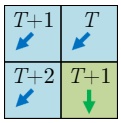
Case 29. The template is of the form:



for some $T \in \mathbb{N}$. Plugging these values for the formula for Δ we directly get that

$$\Delta = (2-x)\delta \geq \delta > \varepsilon'.$$

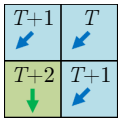
Case 30. The template is of the form:



for some $T \in \mathbb{N}$. Plugging these values for the formula for Δ we directly get that

$$\Delta = (2 + x(-1+y))\delta \geq \delta > \varepsilon'.$$

Case 31. The template is of the form:



for some $T \in \mathbb{N}$. Plugging these values for the formula for Δ we directly get that

$$\Delta = (1 + x + y - xy)\delta \geq \delta > \varepsilon'.$$

4.7.3 Our Algorithms' Pseudocode

```
1: function IsPoint(b_x, b_y, c_x, c_y, q_x, q_y)
2:     if q_x > 1 and q_y > 1 then return No
3:     if b_x = b_y then
4:         if V(c_x) = b_x then
5:             if 10c_y + q_y ≤ 10c_x + 1 and q_x = 1 return Yes
6:         else if V(c_y) = b_x then
7:             if 10c_x + q_x > 10c_y + 1 and q_y = 1 then return Yes
8:         end if
9:     else if b_x − 1 = b_y then
10:        if V(c_y) = b_y then
11:            if 10c_x + q_x ≤ 10S(c_y) + 1 and q_y = 1 then return Yes
12:        else if V(c_x) = b_x then
13:            if 10c_y + q_y > 10P(c_x) + 1 and q_x = 1 then return Yes
14:        end if
15:    else return No
16:    end if
17: end function
```

Algorithm 2: Identifies whether the bottom left corner point of square $(10(c_x - 1) + q_x, 10(c_y - 1) + q_y)$ in block (b_x, b_y) lies on the line.

```
1: function IsCross(b_x, b_y, c_x, c_y, q_x, q_y)
2:     if IsPoint(b_x, b_y, c_x, c_y, q_x, q_y) then
3:         if IsPoint(b_x, b_y, c_x, c_y, q_x − 1, q_y) and IsPoint(b_x, b_y, c_x, c_y, q_x, q_y + 1) then
4:             if IsPoint(b_x, b_y, c_x, c_y, q_x + 1, q_y) then return Yes
5:             if IsPoint(b_x, b_y, c_x, c_y, q_x, q_y − 1) then return Yes
6:         end if
7:     else return No
8:     end if
9: end function
```

Algorithm 3: Identifies whether the bottom left corner point of square $(10(c_x - 1) + q_x, 10(c_y - 1) + q_y)$ in block (b_x, b_y) lies on a crossing of two lines.

```
1: function OnTheLine(b_x, b_y, s_x, s_y)
2:     c_x ← 1 + (s_x div 10)
3:     c_y ← 1 + (s_y div 10)
4:     q_x ← 1 + (s_x mod 10)
5:     q_y ← 1 + (s_y mod 10)
6:     if q_x = q_y = 1 then
7:         return IsPoint(b_x, b_y, c_x, c_y, q_x, q_y)
8:     else if IsCross(b_x, b_y, c_x + 1, c_y, 1, 1) then
9:         if q_y = 1 and q_x ≤ 6 then return Yes
10:        if q_x = 6 and q_y ≤ 6 then return Yes
11:        if q_y = 6 and q_x ≥ 6 then return Yes
12:        if q_x = 1 and q_y ≥ 6 and IsCross(b_x, b_y, c_x, c_y, 1, 1) then return Yes
13:        else return No
14:    else if IsCross(b_x, b_y, c_x, c_y, 1, 1) then
15:        if q_x = 1 and q_y ≥ 6 then return Yes
16:        if q_x = 1 and q_y ≤ 5 then return No
17:        if q_y = 1 then return IsPoint(b_x, b_y, c_x, c_y, q_x, q_y)
18:    else return IsPoint(b_x, b_y, c_x, c_y, q_x, q_y)
19:    end if
20: end function
```

Algorithm 4: Identifies whether the bottom left corner point of square (s_x, s_y) in block (b_x, b_y), lies on the line.

```
1: function AboveOrBelow(b_x, b_y, s_x, s_y)
2:     if OnTheLine(b_x, b_y, s_x + 1, s_y) then return Above
3:     else if OnTheLine(b_x, b_y, s_x, s_y + 1) then return Below
4:     else return ⊥
5:     end if
6: end function
```

Algorithm 5: Identifies whether a square (s_x, s_y) in block (b_x, b_y) touches the line from above or below.

```
1: function Color(b_x, b_y, s_x, s_y)
2:     if AboveOrBelow(b_x, b_y, s_x, s_y) ≠ ⊥ then
3:         if AboveOrBelow(b_x, b_y, s_x, s_y) = Above then return Yellow
4:         else if AboveOrBelow(b_x, b_y, s_x, s_y) = Below then return Red
5:         end if
6:     else if AboveOrBelow(b_x, b_y, s_x, s_y − 1)= Above then return Green
7:     else if AboveOrBelow(b_x, b_y, s_x + 1, s_y − 1)= Above then return Green
8:     else if AboveOrBelow(b_x, b_y, s_x, s_y + 1)= Below then return Purple
9:     else if AboveOrBelow(b_x, b_y, s_x − 1, s_y + 1)= Below then return Purple
10:    else return Blue
11:    end if
12: end function
```

Algorithm 6: Assigns the f-color to any square (s_x, s_y) in block (b_x, b_y).

```
1: function Value(b_x, b_y, s_x, s_y)
2:     d ← 2^n(b_x − 1 + b_y − 1) + s_x + s_y.
3:     if Color(b_x, b_y, s_x, s_y) ∈ {Yellow,Red} then return M − 1 + d
4:     else return 2L − 10 − d.
5:     end if
6: end function
```

Algorithm 7: Assigns the p-value to any square (s_x, s_y) in block (b_x, b_y).

4.7.4 Pseudocode for S' and P'

For completeness, we provide the pseudocode of S' and P' as given by [BPR15]. Let $(\mathsf{S}, \mathsf{V}, T)$ be an SVL instance and let $t = \lceil \log(T+1) \rceil$. We construct an instance $(\mathsf{S}', \mathsf{P}')$ for the EOL problem where $m = t \cdot (n+t)$. We interpret every node in $\mathcal{B}^m$ as a sequence $(u_1, \ldots, u_t)$ of t *states* where, for every $j \in [t]$, the state u_j is of the form $(x, i) \in \mathcal{B}^n \times [T]$. We say that a state (x, i) is *valid* if $\mathsf{V}(x, i) = 1$ and denote the i-th valid state $(\mathsf{S}^{i-1}(0^n), i)$ by $v(i)$. Given $u = (x, i)$, we abuse notation (overloading the function S) and denote $\mathsf{S}(u) := (\mathsf{S}(x), i+1)$. Given $u = (x, i)$ and $u' = (x', i')$, we say that $u < u'$ if $i < i'$.

The functions S' and P' are defined below.

```
1: function S′(N = (u_1, ..., u_t))
2:     return S_t((0^n, 1), N)
3: end function
4:
5: function S_1(u_b = (x, i), N = (u_1, ..., u_t))
6:     if N contains an invalid state then
7:         return N unchanged
8:     end if
9:     if u_j is free then
10:        Set u_1 ← S(u_b)
11:        return (u_1, ..., u_t)
12:    else
13:        return N unchanged
14:    end if
15: end function
16:
17: function S_j(u_b = (x, i), N = (u_1, ..., u_t))
18:    if N contains an invalid state then
19:        return N unchanged
20:    end if
21:    if u_j is free then
22:        N′ ← S_{j−1}(u_b, N)
23:        if N′ ≠ N then
24:            return N′
25:        else if for all k ∈ [j − 1], u_k = v(i + 2^{j−1} − 2^{k−1}) then
26:            Set u_j ← S(u_1)
27:            return (u_1, ..., u_t)
28:        end if
29:        return N unchanged
30:    else if u_j = v(i + 2^{j−1}) then
31:        if for every k ∈ [j − 1], u_k is either free or u_k < u_j then
32:            N′ ← P_{j−1}(u_b, N)
33:            if N′ ≠ N then
34:                return N′
35:            else if for all k ∈ [j − 1], u_k is free then
36:                return S_{j−1}(u_j, N)
37:            end if
38:            return N unchanged
39:        else if for every k ∈ [j − 1], u_k is either free or u_k > u_j then
40:            return S_{j−1}(u_j, N)
41:        end if
42:    end if
43:    return N unchanged
44: end function
```

Algorithm 8: The function S′.

function $\mathsf{P}'(N = (u_1, \ldots, u_t))$
    return $\mathsf{P}_t((0^n, 1), N)$
end function

function $\mathsf{P}_1(u_b = (x, i), N = (u_1, \ldots, u_t))$
    if N contains an invalid state **then**
        return N unchanged
    end if
    if $u_j = v(i+1)$ **then**
        Set $u_1 \leftarrow v(1)$
        return $(u_1, \ldots, u_t)$
    else
        return N unchanged
    end if
end function

function $\mathsf{P}_j(u_b = (x, i), N = (u_1, \ldots, u_t))$
    if N contains an invalid state **then**
        return N unchanged
    end if
    if u_j is free **then**
        return $\mathsf{P}_{j-1}(u_b, N)$
    else if $u_j = v(i + 2^{j-1})$ **then**
        if for every $k \in [j-1]$, u_k is either free or $u_k < u_j$ **then**
            $N' \leftarrow \mathsf{S}_{j-1}(u_b, N)$
            if $N' \neq N$ **then**
                return N'
            else if for all $k \in [j-1]$, $u_k = v(i + 2^{j-1} - 2^{k-1})$ **then**
                Set $u_j \leftarrow v(1)$
                return $(u_1, \ldots, u_t)$
            end if
            return N unchanged
        else if for every $k \in [j-1]$, u_k is either free or $u_k > u_j$ **then**
            $N' \leftarrow \mathsf{P}_{j-1}(u_j, N)$
            if $N' \neq N$ **then**
                return N'
            else if for all $k \in [j-1]$, u_k is free **then**
                return $\mathsf{S}_{j-1}(u_b, N)$
            end if
        end if
    end if
    return N unchanged
end function

Algorithm 9: The pseudocode for circuit P'.

4.7.5 Obfuscation Definitions

We say that two circuits C and C' are *equivalent* and denote it by $C \equiv C'$ if they compute the same function (i.e., $\forall x : C(x) = C'(x)$).

Indistinguishability Obfuscation

Definition 34 (Perfect/Imperfect Indistinguishability Obfuscator)**.** *Let $\mathcal{C} = \{\mathcal{C}_n\}_{n\in\mathbb{N}}$ be a class of polynomial-size circuits, where $\mathcal{C}_n$ is a set of circuits operating on inputs of length n. A uniform algorithm $i\mathcal{O}$ is called an (*imperfect*) indistinguishability obfuscator for the class $\mathcal{C}$ if it takes as input a security parameter and a circuit in $\mathcal{C}$ and outputs a new circuit so that following properties are satisfied:*

1. *(Perfect/Imperfect) Preserving Functionality:*

   There exists a negligible function α such that for any input length $n \in \mathbb{N}$, any λ and any $C \in \mathcal{C}_n$ it holds that

   $$\Pr_{i\mathcal{O}}\left[C \equiv i\mathcal{O}(1^\lambda, C)\right] \geq 1 - \alpha(\lambda),$$

   where the probability is over the internal randomness of $i\mathcal{O}$. If $\alpha(\cdot) = 0$, then we say that $i\mathcal{O}$ is perfect*.*

2. *Polynomial Slowdown:*

   There exists a polynomial $p(\cdot)$ such that: For any input length $n \in \mathbb{N}$, any λ and any circuit $C \in \mathcal{C}_n$ it holds that $\left|i\mathcal{O}(1^\lambda, C)\right| \leq p(|C|)$.

3. *Indistinguishable Obfuscation:*

   For any probabilistic polynomial-time algorithm D, any $n \in \mathbb{N}$, any two equivalent *circuits $C_1, C_2 \in \mathcal{C}_n$ of the same size and large enough λ, it holds that*

   $$\left|\Pr_{i\mathcal{O},D}\left[D\left(i\mathcal{O}\left(1^\lambda, C_1\right)\right) = 1\right] - \Pr_{i\mathcal{O},D}\left[D\left(i\mathcal{O}\left(1^\lambda, C_2\right)\right) = 1\right]\right| \leq \mathsf{negl}(\lambda).$$

We say that $i\mathcal{O}$ is efficient if it runs in polynomial-time.

Virtual Black-Box Obfuscation

Definition 35 (Perfect/Imperfect VBB Obfuscator)**.** *Let $\mathcal{C} = \{\mathcal{C}_n\}_{n\in\mathbb{N}}$ be a class of polynomial-size circuits, where $\mathcal{C}_n$ is a set of circuits operating on inputs of length n. A uniform algorithm $\mathcal{O}$ is called an (*imperfect*) VBB obfuscator for the class $\mathcal{C}$ if it takes as input a security parameter and a circuit in $\mathcal{C}$ and outputs a new circuit so that following properties are satisfied:*

1. *(Perfect/Imperfect) Preserving Functionality:*

   There exists a negligible function α such that for any input length $n \in \mathbb{N}$, any λ and any $C \in \mathcal{C}_n$ it holds that

   $$\Pr_{\mathcal{O}}\left[C \equiv \mathcal{O}(1^\lambda, C)\right] \geq 1 - \alpha(\lambda),$$

   where the probability is over the internal randomness of $\mathcal{O}$. If $\alpha(\cdot) = 0$, then we say that $\mathcal{O}$ is perfect*.*

2. *Polynomial Slowdown:*

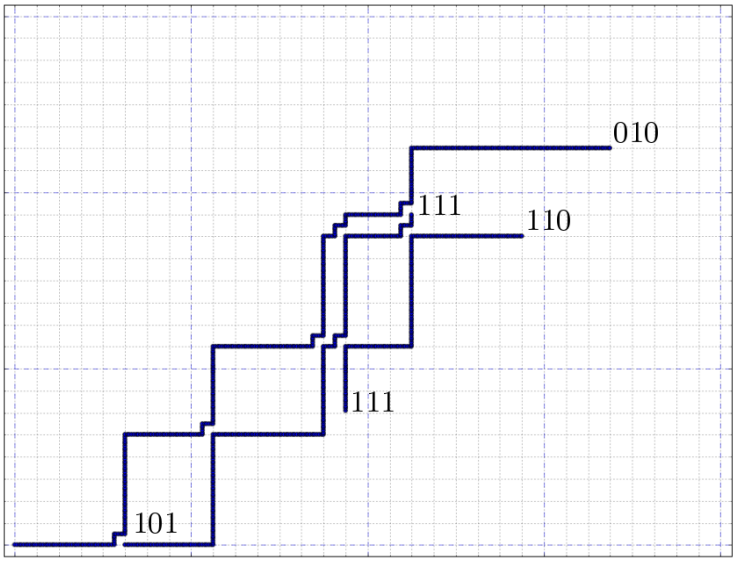
   There exists a polynomial $p(\cdot)$ such that: For any input length $n \in \mathbb{N}$, any λ and any circuit $C \in \mathcal{C}_n$ it holds that $\left|\mathcal{O}(1^\lambda, C)\right| \leq p(|C|)$.

3. *Virtual Black-Box:*

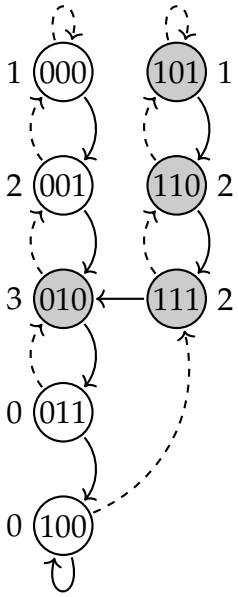
   For any probabilistic polynomial-time algorithm D, any predicate $\pi : \mathcal{C}_n \to \mathcal{B}$, any $n \in \mathbb{N}$ and any circuit $C \in \mathcal{C}_n$, there is a polynomial-size simulator S such that for large enough λ it holds that

   $$\left|\Pr_{\mathcal{O},D}\left[D\left(\mathcal{O}\left(1^\lambda, C\right)\right) = \pi(C)\right] - \Pr_{S}\left[D\left(S^C\left(1^\lambda\right)\right) = \pi(C)\right]\right| \leq \mathsf{negl}(\lambda).$$

We say that $\mathcal{O}$ is efficient if it runs in polynomial-time.



(a)



(b)

Figure 4.7: (a) A collection of staircases obtained by our reduction from a specific EOML instance. Each end point of a staircase is marked by the label of the corresponding vertex that constitutes a solution in the EOML instance. (b) The EOML instance. A solid (respectively dashed) arrow from i to j denotes that j is a successor (respectively predecessor) of i. The value next to a vertex denotes its valuation under V. The vertices with dark background are the solutions obtained via our reduction: 101 is a solution of the second type ($\mathsf{V}(101) = 1$), 110 and 010 are solutions of the third type since the value of their successors is not incremented, 111 is a solution of the third type since the value of its predecessor is not decremented, and additionally also a solution of the first type because it constitutes a sink in the graph ($\mathsf{P}(\mathsf{S}(111)) = 001 \neq 111$).

Chapter 5

White-Box vs. Black-Box Complexity of Search Problems: Ramsey and Graph Property Testing

5.1 Introduction

Consider a setting where one is given a large object (e.g., a graph) and the goal is to find some local pattern (e.g., a certain subgraph) in the object or determine whether it satisfies some property. We investigate the relationship between the black-box setting, where access to the object is via oracle queries, and the white-box setting, where access to the object is given by a program or a circuit, in the context of search problems in which a solution is guaranteed[1] to exist and in the context of property testing.

The Ramsey problem. The Ramsey number $R(n)$ is the minimal number such that any graph on $R(n)$ vertices contains a clique or independent set of size n. The Ramsey theorem states that for any n, it holds that $R(n)$ is finite and moreover that $R(n) \leq 2^{2n}$. This guarantee raises the following question: *Given a graph with 2^{2n} nodes, how difficult is it to find n nodes that are either a clique or an independent set?*

The standard proof of Ramsey's theorem is actually constructive and yields an algorithm that finds the desired clique or independent set, but explores a linear (in the graph size) number of nodes and edges. Is it necessary to explore a large portion of the graph? This of course depends on the representation of the graph and the computational model. In the black-box model, where the access to the graph is merely by oracle queries, Impagliazzo and Naor [IN88] observed that any randomized algorithm must make at least $\Omega(2^{n/2})$ queries before finding the desired clique or independent set. This was based on the fact that a random graph on 2^{2n} vertices

[1] We are not talking about promise problems, but rather when there is a proof that the pattern exists.

has no clique or independent set of size $4n$ with high probability (see Section 2.6).

In this work we are interested in the white-box model[2], where the above question is phrased as: *Given a Boolean circuit encoding the edges of a graph with 2^{2n} nodes, how difficult is it to find n nodes that are either a clique or an independent set?* This question has been explicitly asked by Buss [Bus09] and Goldberg and Papadimitriou [GP17] in the context of search problems in the complexity class TFNP. The class TFNP, defined by Megiddo and Papadimitriou [MP91], is the class of all search problems for which a solution is guaranteed to exist for every instance and verifying a solution can be done efficiently. Thus, the problem where the input is a graph defined by a circuit and the target is to find a clique or an independent set (of appropriate sizes) belongs to the class TFNP.

Our first result is an answer to this question. We show that under the assumption that collision resistant hash functions[3] exist, there exists an efficiently samplable distribution of circuits (circuits on $4n$ inputs representing graphs on 2^{2n} vertices), for which finding a clique or independent set of size n is impossible for any polynomial-time (in n) algorithm. The proof goes along the lines of Krajícek [Kra05], which showed a similar result in the context of the proof complexity of the Ramsey problem.

We also prove a white-box lower bound of a similar flavor for a related problem known as the colorful Ramsey problem. While a graph can be viewed as the edges colored in one color and the non-edges in another, (a simple version of) the colorful Ramsey theorem says that given the complete graph on 2^{2n} vertices and any coloring of its edges using roughly $n/\log n$ colors, there must exist a monochromatic triangle (see Section 2.6 for the precise statement). The question is: given a circuit that represents such a colored graph, what is the computational complexity of finding a monochromatic triangle? We show that this is also hard: assuming collision resistant hash functions, finding a monochromatic triangle is impossible for polynomial-time (in n) algorithms.

Is collision resistance necessary? We complement our result by showing that a form of collision resistance is necessary for the hardness of a variant of the Ramsey problem. Concretely, we consider the bipartite version of the Ramsey problem where the goal is to find a bi-clique or bi-independent set in a bipartite graph. We show that the hardness of this problem implies the existence of a new notion of collision resistance we call *multi-collision resistant hash functions*. These functions guarantees that it is hard to find *multiple* inputs that hash to the same output.[4] In additional, we show the other direction: the hardness of the bipartite Ramsey problem (and the

[2] An example of a graph given as a white-box is the Hadamard graph, where the two inputs are treated as vectors over GF[2] and there is an edge if and only if the inner product between them is 1.

[3] A collision resistant hash is a hash function that shrinks by one bit such that it is hard to find two inputs that hash to the same output.

[4] Any collision resistant hash function is also multi-collision resistant, but the other direction is not known.

standard Ramsey problem) can be based on the existence of multi-collision resistant hash functions. That is, there is an equivalence between the hardness of the bipartite Ramsey problem and the existence of multi-collision resistant hash functions.

Recently, the notion of multi-collision resistance was studied in several work [BDRV17, BKP17, KNY17] showing that this primitive is useful for various cryptographic applications, including statistically-hiding succinct commitment schemes and round-efficient zero-knowledge protocols.

Impossibility of a generic transformation. In the context of search problems, the black-box model (in which the algorithm has only query access to the function) has been extensively studied as it gives hope to prove *unconditional* query lower bounds (see Lovász et al. [LNNW95] for example)[5]. It is tempting to try and translate any query lower bound (in the black-box model) into a white-box lower bound using cryptographic assumption. We show that such a transformation is impossible to achieve in general for search problems in TFNP.[6] Specifically, we present a search problem in TFNP for which the black-box complexity is exponential but for *any* white-box implementation, there exists an algorithm that finds the solution in polynomial time. Our impossibility result is unconditional and does not rely on any cryptographic assumption. It is based on ideas stemming from Canetti et al. [CGH04] concerning limitations of transferring cryptographic schemes that use random oracles to ones that do not appeal to them (see below). Specifically, the construction utilizes the work of Goldwasser and Kalai [GK03] on signature schemes using the Fiat-Shamir paradigm.

The succinct black-box model. In the black-box model, as we have discussed, solving the Ramsey problem requires polynomially many queries in the size of the graph (i.e. exponential in the subgraph we are looking for) and this is also the case for many other problems in TFNP, such as PPP, PLS, PPAD and CLS (see [BCE+98] and [HY17]). In this model, the size of the representation of the function is unbounded and the *running time* of the algorithm accessing the object via queries is unbounded. In contrast, in the white-box model the size of the representation of the object is limited. We consider the question of whether the representation of the function should indeed be unbounded in order to obtain hardness results and study the succinct black-box model (see Definition 28). In this model, the function is represented succinctly but the algorithm is unbounded and has only black-box access to the function.

[5] Over the years several "lifting" techniques were developed in order to translate query lower bounds into lower bounds in other models. Perhaps the most famous example is the lifting technique of [RM99, She11, GPW15] that has been very useful in translating query lower bounds into communication complexity lower bounds.

[6] We note that our impossibility result only rules out a general transformation for all search problem in TFNP. It is an interesting question to find specific problems in TFNP that admit such a transformation.

For this model we show that *any problem in TFNP is easy* (and in particular, the Ramsey problem). That is, there exist a (deterministic) algorithm that performs only a *polynomial* number of queries (in the size of the representation of the function) and finds a solution. One interesting take-away from this result is that any exponential query lower bound (in the black-box model) for a problem in TFNP must use instances of functions (i.e., "boxes") of exponential size. In Table 5.1 we give a short summary of the above results.

Model	$\mathcal{R}$ Problem	**Ramsey-like**	**TFNP**
Black-Box (BB)	Hard (Sec. 5.4)	Hard [IN88]	Hard [HPV89]
White-Box	Easy (Sec. 5.4)	Hard (Sec. 5.2)	Hard [Pap94, HNY17]
Succinct Black-Box	Easy (Sec. 5.5)	Easy (Sec. 5.5)	Easy (Sec. 5.5)

Table 5.1: A summary of our results on the complexity of search problems. $\mathcal{R}$ is the problem defined in Section 5.4. The term "Ramsey-like" problems refers to the problems we consider in Section 5.2, including Ramsey's problem, the colorful Ramsey problem, and the bipartite Ramsey problem.

White-box graph property testing lower bounds. Property testing studies problems of the type: given the ability to perform queries concerning local properties of an object, decide whether the object has some (predetermined) global property, or it is *far* from having such a property. The complexity of a problem is determined by the number of queries required for an algorithm to decide the above correctly.

In all classical works in this field, access to the tested object is given via queries to a black-box. We study the complexity of property testing given a white-box representation. The object is represented implicitly as a program or a circuit and is given to the solver. The solver has to decide whether the object that is encoded in the circuit has a predefined property or not.

We show that cryptographic assumptions can be useful to prove that meaningful properties of graphs are hard to test in the white-box model by any efficient algorithm. The cryptographic assumption we rely on is the existence of a *collection of lossy functions* [PW11]. A collection of lossy functions consists of two families of functions. Functions in the first family are injective, whereas functions in the second family are lossy, namely the size of their image is significantly smaller than the size of their domain. The security requirement is that a description of a randomly chosen function from the first family is computationally indistinguishable from a description of a randomly chosen function from the second family.

We show that there exists a graph property such that, assuming a collection of lossy functions, there exists an efficiently samplable distribution over implicitly represented graphs over 2^n vertices for which testing whether the graph has the property or is far from having it cannot be decided by any polynomial-time (in n) algorithm. The property is whether the graph is a two-source extractor.

5.1.1 Graph-hash product

Our white-box hardness results are based on a technique we call "the graph-hash product", where we generate a new graph from an existing one by embedding the nodes of the new graph via a hash function (see Definition 36). Depending on the properties of the hash function we get various results. The key property of this product operation is that if the hash function is collision resistant, we get that the new graph looks locally as the original one. All of our hardness results, including the hardness of (all variants of) the Ramsey problem and the hardness of the graph property testing, are based on variants of this technique.

We mention that additional hardness results can be shown using our product technique. Also, the technique is not restricted to graph problems. For example, assuming collision resistant hash functions, we prove hardness for finding a sunflower configuration in a large family of sets of the same size. This is a natural (total) search problem that arises from the famous sunflower lemma of Erdös and Rado [ER60]. We refer to Section 5.7.1 for more information.

A similar graph-hash product was used by Krajícek [Kra01] relating the proof complexity of the weak pigeonhole principle and the proof complexity of the Ramsey theorem.

5.1.2 Cryptographic assumptions and white-box lower bounds

For some search problems it is known how to obtain hardness in the white-box model under certain cryptographic assumptions. One of the first examples is due to Papadimitriou [Pap94] who showed that the hardness of the class PPP (a subclass in TFNP) can be based on the existence of one-way *permutations* (the hardness can be also based on the existence of collision resistant hash functions). We refer to [HNY17] for more information about the assumptions that lead to white-box hardness in TFNP.

Obfuscation. It has been recently shown that program obfuscation is very useful for proving white-box lower bounds for search problems. An obfuscator transforms a given program (say described as a Boolean circuit) into another "scrambled" circuit which is functionally equivalent by "hiding" its implementation details. One could hope to take the underlying black-box instance, obfuscate it and use this obfuscated version as the white-box instance. Obfuscation is a strong and (still) somewhat controversial assumption (see Ananth et al. [AJN+16] for a discussion), but if it could be used for a general transformation, then we would get a large class of white-box hardness results. However, there are a few obstacles in applying such an approach: First, Canetti et al. [CGH04] (followed by the work of Goldwasser and Kalai [GK03]) showed that it is impossible to generically translate security of cryptographic primitives in the random oracle model into primitives in the standard setting. Second, ideal program obfuscators ("virtual black-box") do not exist for general functionalities [Had00, BGI+12b], so we have to work with weaker primitives such as indistinguishability obfuscation [BGI+12b, GGH+13b, SW14b]. One prominent instance of

using indistinguishability obfuscation in order to prove white-box lower bounds was shown in the context of PPAD-hardness [BPR15, GPS16, HY17, KS17b], but it is hard to see how to use indistinguishability obfuscation for a more general transformation from black-box hardness to white-box hardness.

Our white-box hardness results do *not* use obfuscation at all and as such bypass the above issues. Furthermore, our techniques show that weaker (and much better studied) primitives can be used to hide information in a meaningful way.

5.2 Hardness of The Ramsey Problem

We show a hard distribution for the Ramsey problem. In this problem, one is given an implicit and efficient representation of the adjacency matrix of a graph on 2^n vertices, and the goal is to find either a clique of size $n/2$ or an independent set of size $n/2$. The implicit representation of the graph is by a circuit $C\colon \mathcal{B}^n \times \mathcal{B}^n \to \mathcal{B}$ that represents the adjacency matrix of a graph on 2^n vertices.

In terms of Definition 25, we have the $q(n) = \binom{n/2}{2}$ and the relation $\mathcal{S}$ is such that $\left(x_1, \ldots, x_{q(n)}, f(x_1), \ldots, f(x_{q(n)})\right) \in \mathcal{S}$ if and only if the edges $x_1, \ldots, x_{q(n)}$ form a clique or an independent set of size n. That is, the set of vertices touched by some edge in $x_1, \ldots, x_{q(n)}$ is of size exactly n, and $f(x_1) = \ldots = f(x_{q(n)}) = b$ for some $b \in \mathcal{B}$.[7]

Hardness of the Ramsey problem. We say that the Ramsey problem is *hard* if there exists an efficiently samplable distribution $\mathcal{D} = \{C\colon \mathcal{B}^n \times \mathcal{B}^n \to \mathcal{B}\}$ over circuits of size polynomial in n that represent graphs on 2^n vertices, such that for every probabilistic polynomial-time algorithm A, there exists a negligible function $\mathsf{negl}(\cdot)$ such that

$$\Pr_{C \leftarrow \mathcal{D}}[v_1, \ldots, v_{n/2} \leftarrow A(1^n, C)\ ;\ v_1, \ldots, v_{n/2} \text{ form a clique or an independent set}] \leq \mathsf{negl}(n),$$

where the probability is over the uniform choice of $C \leftarrow \mathcal{D}$ and the randomness of A. All the efficiency requirements are polynomial in n.

The above problem is indeed in TFNP as it is guaranteed by Proposition 5 that there always exists a monochromatic clique or independent set of size $n/2$. We show that under a certain cryptographic assumptions, the existence of collision resistant hash (CRH) functions (see Section 2.2.2), there exists an efficiently samplable distribution over instances of the Ramsey problem for which no efficient algorithm can find a solution. Recall that if CRH functions compressing by one bit exist, then CRH functions compressing by any polynomial factor (i.e. from n bits to n^δ for any fixed constant $\delta > 0$) exist. We will use a collision resistant hash function family $\mathcal{H} = \{h\colon \mathcal{B}^n \to \mathcal{B}^{n/4}\}$.

[7] We say that an edge (u, v) touches the vertices u and v.

Theorem 15. *The Ramsey problem is hard assuming the existence of collision resistant hash functions.*

In the proof of Theorem 15 we use a construction of Ramsey graphs given in Proposition 6 as well as a type of graph product operation: the operation takes as input a graph G on 2^n vertices and a hash function $h\colon \mathcal{B}^{n+\ell} \to \mathcal{B}^n$, where $\ell \geq 1$ and outputs a graph $G \otimes h$ on $2^{n+\ell}$ vertices, whose edges depend on the edges in G and the hash function.

Definition 36 (The graph-hash product)**.** *Given a graph $G = (V, E)$, where $|V| = \mathcal{B}^n$, and a hash function $h\colon \mathcal{B}^{n+\ell} \to \mathcal{B}^n$, define the graph $G \otimes h = (V', E')$ as a graph on vertices $V' = \mathcal{B}^{n+\ell}$ with edges E' such that $(u, v) \in E'$ if and only if $(h(u), h(v)) \in E$.*

Observe that given an efficient representation of G and an efficient representation of h, we have an efficient representation of the graph $G \otimes h$. We are now ready to give the proof of Theorem 15.

Proof of Theorem 15. Let $k = n/4$, let $\mathcal{H}$ be a family of collision resistant hash function from n bits to k bits; such a family $\mathcal{H}$ exists under the assumption that collision resistant hash functions that compress by one bit exist. Let $\mathcal{G} = \{g\colon \mathcal{B}^k \times \mathcal{B}^k \to \{0,1\}\}$ be a $2k^2$-wise independent hash function family, where each member $g \in \mathcal{G}$ defines a graph G on 2^k vertices in the natural way (see below). By Proposition 6, most $g \in \mathcal{G}$ define a graph G that does not contain any clique or independent set of size $2k = n/2$. The following sampling procedure yields a graph (V', E'), where $|V'| = 2^n$:

1. Sample a collision resistant hash function $h \leftarrow \mathcal{H}$ and a function $g \leftarrow \mathcal{G}$.
2. Set $G = (V, E)$ to be the graph with $|V| = 2^k$ vertices induced by g (see Proposition 6).
3. Output h and g as representing the graph-hash product $G \otimes h = (V', E')$. That is, for any $x, y \in V'$ s.t. $x < y$ we have that edge (x, y) exists iff $g(h(x), h(y)) = 1$.

The Ramsey challenge on (V', E') is to find a clique or independent set of size $n/2$ (since $|V'| = 2^n$). We reduce the ability of an adversary to solve the Ramsey problem to an adversary that breaks the collision resistance of $h \leftarrow \mathcal{H}$.

Suppose that there exists an adversary A that, given an instance of the distribution above, finds a clique or an independent set of size $n/2 = 2k$ in $G \otimes h$ with probability $\epsilon > 0$ (over the choice of h, g, and the randomness of A). Denote this event by $\mathcal{A}\text{-}(\text{-wins}A, g, h)$. That is,

$$\Pr[\mathcal{A}\text{-}(\text{-wins}A, g, h)] \geq \epsilon.$$

Let $(v_1, \ldots, v_{2k})$ the solution found by A, and let $v'_i \triangleq h(v_i)$ for $i \in [2k]$. Let Distinct be the event in which in the solution output by A, the values $v'_1, \ldots, v'_{2k}$ are distinct. Then, by the assumption it holds that

$$\begin{aligned}\Pr[\mathcal{A}\text{-}(\text{-wins}A, g, h)] = &\Pr[\mathcal{A}\text{-}(\text{-wins}A, g, h) \mid \mathsf{Distinct}] \cdot \Pr[\mathsf{Distinct}] + \\ &\Pr[\mathcal{A}\text{-}(\text{-wins}A, g, h) \mid \neg\mathsf{Distinct}] \cdot \Pr[\neg\mathsf{Distinct}] \geq \epsilon \qquad (5.2.1)\end{aligned}$$

We first argue that

$$\Pr[\mathcal{A}\text{-}(\text{-wins}A, g, h) \mid \mathsf{Distinct}] \leq \exp(-n).$$

Indeed, by the definition of the event Distinct, it holds that $v'_1, \ldots, v'_{2k}$ are distinct, and by the definition of our graph-hash product, the sequence of vertices $(v'_1, \ldots, v'_{2k})$ must form a clique or an independent set of size $2k$ in G. However, by Proposition 6 we know that with probability $1 - \exp(-n)$ over g, the graph G *does not contain* any such independent set or clique.

Plugging this back into Equation (5.2.1), we get that

$$\Pr[\mathcal{A}\text{-}(\text{-wins}A, g, h) \mid \neg\mathsf{Distinct}] \cdot \Pr[\neg\mathsf{Distinct}] \geq \epsilon - \exp(-n)$$

and, in particular,

$$\Pr[\neg\mathsf{Distinct}] \geq \epsilon - \exp(-n)$$

Recall that ϵ is a non-negligible term and thus we obtain an algorithm A' that finds a collision in h with probability $\epsilon - \exp(-n)$, which contradicts the collision resistance property of the hash function h. To summarize, the algorithm A' gets as input a hash function h, samples a function $g \leftarrow \mathcal{G}$, as above, and simulates the execution of A on the graph-hash product graph $G \otimes h$. Given the output of A, it searches the output for a pair of values that form a collision relative to h and outputs them (it outputs $\perp$ in case no such pair was found). By the above, two such colliding values will appear in the output with non-negligible probability, resulting in a collision relative to h. □

Hardness for finding a smaller clique or independent set. We showed that it is hard to find a clique or independent set of size $n/2$ in an implicitly represented graph of size 2^n. We can show that *finding a clique or independent set of size* n^δ for any $0 < \delta \leq 1$ is hard, by following the proof of Theorem 15 and using a hash function that maps n bits into n^δ bits (which is implied by the existence of the hash function we used in Theorem 15).

We can even go below a fixed δ to, say, $n^{1/\sqrt{\log n}}$ by using a hash function that compresses a super-polynomial amount (from n bits to $n^{1/\sqrt{\log n}}$ bits). This is known to be implied by a hash function that compresses a single bit albeit with a super-polynomial loss in security, but it is not known with only a polynomial loss.

Ramsey theory and proof complexity. Ramsey theory has been extensively studied in the context of proof complexity. In particular, it is known that Ramsey's theorem has a polynomial-size bounded-depth Frege proof [Pud90] and it is related to the weak pigeonhole principle [Jer09].

5.2.1 Hardness of the colorful Ramsey problem

The colorful Ramsey problem asks, given an implicit and efficient representation of a coloring using m colors of the edges of a graph on 2^n vertices, to find a monochromatic clique of size k. We will see a hard distribution for the colorful Ramsey problem. We focus in the case where the goal is to find a monochromatic triangle (i.e., $k = 3$ above) for simplicity and remark that the proof generalizes for larger values of k.

Hardness of the colorful Ramsey problem. We say that the colorful Ramsey problem is *hard* if there exists an efficiently samplable distribution $\mathcal{D} = \{\psi \colon \binom{2^n}{2} \to [m]\}$ over colorings of the full graph on 2^n vertices, such that for every probabilistic polynomial-time algorithm A, there exists a negligible function $\mathsf{negl}(\cdot)$ such that

$$\Pr_{C \leftarrow \mathcal{D}}[v_1, v_2, v_3 \leftarrow A(1^n, C) \; ; \; v_1, v_2, v_3 \text{ form a monochromatic triangle}] \leq \mathsf{negl}(n),$$

where the probability is over the uniform choice of $C \leftarrow \mathcal{D}$ and the randomness of A.

The above problem is indeed in TFNP whenever $m \leq n/(3 \log n)$, since it is guaranteed by Proposition 9 that there always exists a monochromatic triangle if there are only $n/(3 \log n)$ colors. The theorem below shows that there exists a distribution over instances of the colorful Ramsey problem for which no efficient algorithm can find a solution. As before, the security of the distribution relies on the existence of collision resistance hash functions.

Theorem 16. *The colorful Ramsey problem is hard assuming the existence of a collision resistant hash function family.*

In the proof of Theorem 16, we use an explicit construction of a colorful Ramsey coloring.

Lemma 6. *Fix $k > 2$ and let $m = 2n/\log k$. There exist an efficient and explicit coloring $\psi \colon \binom{2^n}{2} \to [m]$ for which there is no monochromatic complete subgraph of size k.*

Proof. We show a recursive construction. Assume we have a coloring $\psi' \colon \binom{2^n/k}{2} \to [m-1]$ of the full graph on $2^n/k$ vertices with $m-1$ colors such that there is no monochromatic complete subgraph of size k. We show how to get a coloring $\psi' \colon \binom{2^n}{2} \to [m]$ of the full graph on 2^n vertices with m colors. Split the latter into k full graphs $K_1, \ldots, K_k$ each of size 2^n and color each of them (internally) using ψ'. For any edge (u, v) that crosses between copies, that is, $u \in K_i$ and $v \in K_j$, where $i < j$, we assign the color $m-1$ if $i = 1$ and with the color m otherwise.

We show that ψ is a valid coloring, namely, that there is no monochromatic complete subgraph of size k. Consider any k vertices. If they are all from the same copy K_i for some $i \in [k]$ then by the recursive construction they are not monochromatic. If each vertex is in a different copy, then the clique is colored with colors $m-1$ and m and thus not monochromatic. Lastly, if there is a mix of edges in between copies

and among different copies, then we must have at least two different colors since the coloring withing a copy is only from the colors $[m-2]$ while the coloring among different copies uses colors within $\{m-1, m\}$.

We iterate the construction above, starting from a trivial 1-vertex graph. One can see that in each iteration we add 2 colors but multiply by k the number of vertices. Thus, after $\log_k(2^n) = n/\log k$ iteration we obtain a graph with 2^n vertices colored with $m = 2n/\log k$ colors. An explicit algorithm (not in a recursive form) for coloring the edges of the graph is given next. Note that if we write n in the base k representation then it has $m/2$ coordinates.

$\mathsf{Color}(u, v)$:

1. Let $(u_1, \ldots, u_{m/2})$ and $(v_1, \ldots, v_{m/2})$ be the base k representation of u and v, respectively.
2. Let i be the minimal index such that $u_i \neq v_i$.
3. If $u_i = 1$, then output color $2i - 1$.
4. Otherwise, output color $2i$.

□

We proceed with the proof of Theorem 16.

Proof of Theorem 16. We adapt the graph-hash product operation (see Definition 36) to support more general coloring functions (rather than graphs). Given a coloring $\psi\colon \binom{2^n}{2} \to [m]$ of the full graph with 2^n vertices using m colors, and a hash function $h\colon \mathcal{B}^{n+\ell} \to \mathcal{B}^n$, we define a coloring of the full graph on $2^{n+\ell}$ vertices $\psi \otimes h$ as follows. For any edge $e = (u, v) \in \mathcal{B}^{n+\ell} \times \mathcal{B}^{n+\ell}$, let $u' = h(u)$ and $v' = h(v)$. The color of e is $\psi(u', v')$ (i.e., the color of (u', v') according to ψ) if $u' \neq v'$ and it is 1 if $u' = v'$.

We proceed with the main construction. Let $n' = \frac{n \log 3}{6 \log n}$ and let $\psi\colon \binom{2^{n'}}{2} \to [m]$ be a coloring of the full graph on $2^{n'}$ vertices with $m \triangleq 2n'/\log 3 = n/(3 \log n)$ colors that does not contain a monochromatic triangle given by Lemma 6. Sample a collision resistant hash function $h \leftarrow \mathcal{H}$, where $\mathcal{H} = \{h\colon \mathcal{B}^n \to \mathcal{B}^{n'}\}$ is a collision resistant hash function compressing n bits to n' bits, and color the full graph on 2^n vertices by $\psi \otimes h$. This coloring, that consists of a description of ψ and h, is the output of the sampling procedure of the distribution.

Suppose there exist an adversary A that, given an instance of the distribution above, finds a monochromatic triangle in $\psi \otimes h$ with probability $\epsilon > 0$. Denote by $(i_1, v_1), (i_2, v_2), (i_3, v_3) \in \mathcal{B}^{n/2} \times \mathcal{B}^{n/2}$ the solution found by A, and let $v'_i = h(i, v_i)$ for $i \in [3]$. We first observe that by Lemma 6 it must be that the v'_i's are not distinct. Indeed, if they are distinct then (by the definition of our coloring-hash product) the sequence of vertices (v'_1, v'_2, v'_3) correspond to a monochromatic triangle in ψ, which cannot happen (since ψ does not have any such monochromatic triangle). Given that

the v_i''s are not distinct, they must contain a collision relative to h. Thus, we obtain an algorithm A' that finds a collision for h with the same probability probability ϵ. □

5.2.2 The relation of Ramsey and the WeakPigeon classes

The weak pigeon class (PWPP^n_k) is a subclass in TFNP defined by the collision finding problem for circuits that compress their input.

Definition 37 (PWPP^n_k complete problem)**.** *Given a circuit $C\colon \mathcal{B}^n \to \mathcal{B}^k$, find $x \neq y$ such that $C(x) = C(y)$. Moreover, we define $PWPP = PWPP^n_{n-1}$.*

In [Jer16] it has been shown that the class PWPP^n_{n-1} is equivalent to the class $\mathsf{PWPP}^n_{n^\delta}$ for any positive constant δ. Moreover, any hard problem for PWPP^n_k naturally gives rise for a collision resistant hash function.

In Theorem 15 we showed a reduction from the hardness of the Ramsey problem to the hardness of collision resistant hash functions, that is, to PWPP. Let RAMSEY be the set of all search problems which are reducable in polynomial-time to the Ramsey problem. Then, we get that the class PWPP is contained under *randomized* reductions in the class RAMSEY. The only source of randomness is our reduction is sampling the limited-independence hash function $g \in G$ that defines a Ramsey graph. We observe that we can overcome this issue (i.e., get a deterministic reduction) by relying on explicit constructions of Ramsey graphs. The currently best explicit constructions of Ramsey graphs do not get the same parameters as a random graph, and hence to use it we need a stronger hash function (i.e. with better compression rate).

The explicit construction that we use comes from an exciting line of recent works in the area of randomness extractors (for example [Coh16a, CZ16, BDT16, Coh16b, Li16]).[8] We will use the following theorem.

Proposition 10. *There exists an explicit k-Ramsey graph on N vertices, where $k \leq 2^{(\log\log N)^2}$.*[9]

Applying the same proof as that of Theorem 15, but using the explicit construction of Ramsey graphs above, results with:

Theorem 17. *Fix a family of collision resistant hash functions $\mathcal{H} = \{h\colon \mathcal{B}^n \to \mathcal{B}^{2^{\sqrt{\log(n/2)}}}\}$. There exists a* deterministic *reduction from RAMSEY to breaking the collision resistance of $\mathcal{H}$.*

As a corollary of Theorem 17 we obtain that the class defined by the Ramsey problem (i.e., the class RAMSEY) includes the class $\mathsf{PWPP}^n_{2^{\sqrt{\log(n/2)}}}$.

[8] We note that any improvement on the best constructions of Ramsey graphs would imply an improvement in our underlying assumption regarding the hash function.

[9] Li [Li16] shows a stronger result, namely, that $k \leq 2^{(\log\log N)\cdot O(\log\log\log N)}$, but (for simplicity) we will not use this stronger version. Using better explicit constructions of Ramsey graphs (than the one we state in Proposition 10) will directly imply the result in Theorem 17 based on a weaker assumption on the hash function family.

Corollary 8. *$\mathrm{PWPP}^n_{2^{\sqrt{\log(n/2)}}} \subseteq$ RAMSEY.*

Proof of Theorem 17. Let $k = 2^{\sqrt{\log(n/2)}}$. Given a uniformly sampled hash function $h \leftarrow \mathcal{H}$, we show that given a solver for the Ramsey problem it is possible to find collisions in h. Consider the graph $G = (V, E)$ with $|V| = 2^k$ vertices given by Proposition 10 and execute the Ramsey problem solver on the graph-hash product graph $G \otimes h = (V', E')$. Notice that $|V'| = 2^n$ and thus the solver finds a clique or independent set of size $n/2 = 2^{(\log k)^2}$ in $G \otimes h$ with noticeable probability $\epsilon > 0$. Denote the solution by $v_1, \ldots, v_{n/2}$, and let $v'_i = h(v_i)$ for $i \in [n/2]$. We first observe that by Proposition 10 it must be that the v'_i's are not distinct. Indeed, if they are distinct then (by the definition of our graph-hash product) the sequence of vertices $(v'_1, \ldots, v'_{n/2})$ forms a clique or an independent set of size $n/2$ in G (which does not exist!). Now, given that the v'_i's are not distinct, then they must contain a collision relative to h. Thus, we obtained an algorithm that finds a collision for h with probability ϵ. □

Regarding the relation between the colorful ramsey problem and the class PWPP^n_{n-1} we have the following. Using Theorem 16 we obtain that the class defined by the colorful Ramsey problem, denoted by COLORFUL-RAMSEY, includes the class $\mathsf{PWPP}^n_{n/(6\log n)}$. Since PWPP^n_{n-1} is equivalent to $\mathsf{PWPP}^n_{n^\delta}$ for any positive constant δ [Jer16], we obtain the following result.

Corollary 9. *PWPP $\subseteq$ COLORFUL-RAMSEY.*

5.2.3 The Ramsey problem and Multi-CRH

In Theorem 15 we showed that under the assumption that CRH functions exist, the Ramsey problem is hard. Here we study the bipartite version of the Ramsey problem and point out a tight relationship to a cryptographic primitive we call *multi-collision resistant hash* (MCRH) functions.[10]

A bipartite graph on two sets of N vertices is a bipartite K-Ramsey graph if it has no $K \times K$ complete or empty bipartite subgraph. Ramsey's theorem for such graphs says that every bipartite graph on $2N$ vertices has a $\log N \times \log N$ complete or empty bipartite subgraph (see e.g., [Con08]).[11] The result of Erdös [Erd47] on the abundance of $(2\log N)$-Ramsey graphs (see Proposition 6 and Section 5.7.3) holds as is for bipartite graphs.

[10] Multiple collisions in hash functions were studied before in the context of *iterated* hash functions by Joux [Jou04]. He showed that for such functions, finding multi-collisions (a set of k messages that hash to the same value) is not much harder than finding ordinary collisions (pairs of messages that collide).

[11] Given a bipartite K-Ramsey graph G on $2N$ vertices, one can transform it into a non-bipartite $2K$-Ramsey graph H on N vertices. The graph H is defined by the upper triangle of the adjacency matrix of G.

Analogously to the Ramsey problem on graphs, the *bipartite Ramsey problem* is when the graphs are bipartite and the goal is to find a bi-clique or bi-independent set of a certain size. We focus on the task of finding a bi-clique or bi-independent set of size $n/4$. We say that the *bipartite* Ramsey problem is *hard* if there exists an efficiently samplable distribution $\mathcal{D} = \{C\colon \mathcal{B}^n \times \mathcal{B}^n \to \mathcal{B}\}$ over circuits of size polynomial in n that represent *bipartite* graphs on $2^n \times 2^n$ vertices, such that for every probabilistic polynomial-time algorithm A, there exists a negligible function $\mathsf{negl}(\cdot)$ such that

$$\Pr_{C \leftarrow \mathcal{D}}[u_1, \ldots, u_{n/4}, v_1, \ldots, v_{n/4} \leftarrow A(1^n, C)\ ;\ \exists b \in \mathcal{B}, \forall i, j \in [n/4]\colon C(u_i, v_j) = b] \leq \mathsf{negl}(n),$$

where the probability is over the uniform choice of $C \leftarrow \mathcal{D}$ and the randomness of A. All the efficiency requirements are polynomial in n.

Roughly, a family of *multi-collision resistant hash* functions is one such that it is hard to find multiple inputs that hash to the same output. More precisely, a sequence of families of functions $\mathcal{H}_n = \{h\colon \mathcal{B}^{\ell_1(n)} \to \mathcal{B}^{\ell_2(n)}\}$, where ℓ_1 and ℓ_2 are two functions such that $\ell_1(n) > \ell_2(n)$ for every $n \in \mathbb{N}$, is *k-multi-collision resistant* if for every probabilistic polynomial-time algorithms A, it holds that

$$\Pr_{h \leftarrow \mathcal{H}_n}[(x_1, \ldots, x_k) \leftarrow A(1^n, h);\ h(x_1) = \cdots = h(x_k)] \leq \mathsf{negl}(n).$$

By default, unless otherwise stated, we assume that a family of k-MCRH functions maps strings of length n to strings of length $n - \log k$. This assumption ensures that a k-multi-collision exists (but yet it is hard to find). k-MCRH functions are implied by standard CRH functions (but is seemingly a weaker primitive).

We show that MCRH functions are sufficient and necessary for bipartite Ramsey hardness.

Theorem 18. *If the bipartite Ramsey problem is hard, then there exists a family $\mathcal{H} = \{h\colon \mathcal{B}^n \to \mathcal{B}^{n/2}\}$ of $n/4$-MCRH functions.*

Furthermore, if there exists a family $\mathcal{H} = \{h\colon \mathcal{B}^n \to \mathcal{B}^{\sqrt{n}/8}\}$ of $\sqrt{n}$-MCRH functions, then the bipartite Ramsey problem is hard.

Proof. We show that the hardness of the bipartite Ramsey problem implies that $n/4$-MCRH functions exist. Let $\mathcal{D} = \{C\colon \mathcal{B}^n \times \mathcal{B}^n \to \mathcal{B}\}$ be a distribution over succinctly represented graphs on $2^n \times 2^n$ vertices such that it is hard to find a bi-clique or a bi-independent set of size $n/4$. Fix $v_1, \ldots, v_{n/2} \in \mathcal{B}^n$ to be arbitrary $n/2$ distinct vertices on the right side. We define the hash function $h_C\colon \mathcal{B}^n \to \mathcal{B}^{n/2}$ to be the concatenation of the bits $C(x, v_1), \ldots, C(x, v_{n/2})$.

$$h_C(x) = C(x, v_1) \circ \cdots \circ C(x, v_{n/2}).$$

We claim that it is hard to find an $(n/4)$-multi-collision in h_C by translating such a multi-collision to a bi-clique or a bi-independent set of size $n/4$ in C. Let $x_1, \ldots, x_{n/4}$ be a $(n/4)$-multi-collision in h_c and denote by $y = h_C(x_1) \in \mathcal{B}^{n/2}$. Without loss

of generality, assume that the string y has more 1's than 0's and denote by $I = \{i_1, \ldots, i_{n/4}\} \subseteq [n/2]$ a set of size $n/4$ of (distinct) indices for which $y_i = 1$. The collection of vertices $x_1, \ldots, x_{n/4}$ on the left side and the vertices $\{v_{i_1}, \ldots, v_{i_{n/4}}\}$ on the right form a bi-clique of size $n/4 \times n/4$.

For the other direction, namely that the bipartite Ramsey problem is hard *if* $\sqrt{n}$-MCRH functions exist (that map n bits to $\sqrt{n}/8$ bits), we adapt the proof of Theorem 15. First, following the proof of Proposition 6, a bipartite graph on $2 \cdot 2^{\sqrt{n}/8}$ vertices sampled via an $(n/16)$-wise independent family is a $(\sqrt{n}/4)$-Ramsey (bipartite) graph with probability at least $1 - 1/N$.

The hard distribution for bipartite Ramsey is defined similarly to the distribution given in Theorem 15. Specifically, a description of a graph is given via a $\sqrt{n}$-MCRH function h and a $(4 \log^2 N)$-wise independent function g and an edge (x, y) is in the graph iff $g(h(x), h(y)) = 1$.

Given a bi-clique or bi-independent set $u_1, \ldots, u_{n/4}, v_1, \ldots, v_{n/4}$, consider $u'_i = h(u_i)$ and $v'_i = h(v_i)$ for $i \in [n/4]$. Since h is a $\sqrt{n}$-MCRH function, then there are at least $\sqrt{n}/4$ distinct values of u''s and $\sqrt{n}/4$ distinct values of v''s (otherwise, we can break the security of h). Thus, we get a bi-clique or bi-independent set of size $\sqrt{n}/4 \times \sqrt{n}/4$ in the graph defined by g. This contradicts the fact that g contains no such graph (with very high probability). □

Subsequent work. Following this work, the notion of MCRH has been studied in depth showing a variety of applications such as statistically-hiding commitments with short communication and various types of efficient zero-knowledge protocols [KNY17, BKP17, BDRV17].

5.3 Hardness of Testing an Extractor

In this section we present a *graph property* which is hard to test in the white-box setting. Specifically, we present a property Π and a distribution over succinctly-represented graphs for which efficiently deciding whether an instance in the distribution has the property Π or is *far* from having the property Π is impossible (under appropriate cryptographic assumptions). We briefly recall the notions related to (graph) property testing and then describe our main result. A more elaborate introduction can be found in [Gol11] and references therein.

A property Π is simply a set of elements in a universe of interest. A property Π is a graph property, if it is a set of graphs closed under graph isomorphism. That is, if for every graph $G = (V, E)$ on N vertices and any permutation π on V it holds that $G \in \Pi$ if and only if $\pi(G) \in \Pi$, where $\pi(G) = (V, E')$ and $E' = \{(\pi(u), \pi(v)) \mid (u, v) \in E\}$. A graph $G = (V, E)$ on N vertices is said to be ϵ-far from a property Π if for every N-vertex graph $G' = (V', E')$ that has the property Π (i.e., $G' \in \Pi$), it holds that $|E \triangle E'| \geq \epsilon \cdot \binom{N}{2}$ (the operator $\triangle$ denotes symmetric difference).

Definition 38 (White-box property testing)**.** *An ϵ-tester for a graph property Π is a probabilistic machine that, on input a Boolean circuit $C\colon \mathcal{B}^n \times \mathcal{B}^n \to \mathcal{B}$ representing the adjacency matrix of a 2^n-vertex graph G, outputs a binary value that satisfies:*

1. *If G has the property Π, then the tester outputs 1 with probability at least $2/3$.*

2. *If G is ϵ-far from Π, then the tester outputs 1 with probability at most $1/3$.*

The above definition naturally generalized to bipartite graphs (and properties of bipartite graphs).

The property of being an extractor. The graph property Π we are interested in is being a *two-source extractor*: a bipartite graph $G = (U, V, E)$, where $|U| = |V| = 2^n$, is (k, δ)-balanced if for every set $U' \subseteq U$ and $V' \subseteq V$ of size $|U'| = |V'| = 2^k$, the induced subgraph $G_{U',V'}$ has $1/2 \pm \delta$ fraction of edges. The induced subgraph $G_{U',V'} = (U', V', E_{U',V'})$ is defined by $(u, v) \in E_{U',V'}$ if and only if $(u, v) \in E$, $u \in U'$ and $v \in V'$.

We present a distribution over succinctly represented (bipartite) graphs for which testing the above property is hard. The hardness reduces to breaking the security of a *collection of lossy functions* described in Section 2.2.2.

Theorem 19. *Assume the existence of a collection of $(n, 2n/3)$-lossy functions and consider the bipartite graph property Π of being $(0.52n, 2^{-n/2000})$-balanced. There exist a constant $\epsilon > 0$ and a distribution over succinctly represented bipartite graphs on 2^n vertices for which any ϵ-tester for Π must run in super-polynomial-time.*

Observe that the existence of a collection of lossy functions directly implies white-box hardness of testing whether a given function is injective or far from being such (i.e., lossy), but we will prove the hardness for a graph property.

Proof. Assume the existence of a collection of $(n, 2n/3)$-lossy functions defined by the pair of algorithms $(\mathsf{Gen}, \mathsf{Eval})$, where for every s in the output of Gen it holds that $\mathsf{Eval}(s, \cdot) = f_s(\cdot)$ maps strings of length n into strings of length $p(n) > n$ for some polynomial $p(\cdot)$. We construct a new collection of functions defined by a pair of algorithms $(\mathsf{Gen}', \mathsf{Eval}')$:

1. The algorithm Gen', on input 1^n and $b \in \mathcal{B}$, executes the algorithm $s \leftarrow \mathsf{Gen}(1^n, b)$ and samples a n-wise-independent hash-function $h\colon \mathcal{B}^{p(n)} \to \mathcal{B}^n$. It outputs $s' = (s, h)$.

2. The algorithm Eval', on input $s' = (s, h)$ and $x \in \mathcal{B}^n$, outputs $h(\mathsf{Eval}(s, x))$.

Claim 9. *The following holds:*

1. $\mathsf{Gen}'(1^n, 0) \approx_c \mathsf{Gen}'(1^n, 1)$.

2. *The algorithm* $\mathsf{Eval}'(s', \cdot)$*, where* $s' \leftarrow \mathsf{Gen}'(1^n, 0)$*, computes a function* $f_{s'}(\cdot)$ *over* $\mathcal{B}^n$ *such that with very high probability for any* $y \in \mathcal{B}^n$ *it holds that* $|\{ x : f_{s'}(x) = y \}| \leq n$*.*

3. *The algorithm* $\mathsf{Eval}'(s', \cdot)$*, where* $s' \leftarrow \mathsf{Gen}'(1^n, 1)$*, computes a function* $f_{s'}(\cdot)$ *over* $\mathcal{B}^n$ *whose image size is at most* $2^{n/3}$*.*

Proof. The first and last items follow immediately from the properties of $(\mathsf{Gen}, \mathsf{Eval})$. The second item follows by a balls and bins argument. For any n elements, the probability that they all hash to a specific value y is $(2^{-n})^n$. Thus, taking a union bound over all possible values $y \in \mathcal{B}^n$ and all possible n tuples we get that the total probability that there exists a bin with more than n elements is at most $2^n \cdot \binom{2^n}{n} \cdot 2^{-n^2} \leq \exp(-n)$. □

Fix $k = 0.51n$ and $\delta = 2^{-n/500}$ and observe that

$$2k = 2 \cdot 0.51n \geq n + 2\log(2^{n/500}) + 2 = n + \log(1/\delta) + 2.$$

By Proposition 3, there exists a polynomial-size Boolean circuit $B_{n,k,\delta}$ that acts on inputs from $\mathcal{B}^n \times \mathcal{B}^n$ succinctly represents a (k, δ)-balanced bipartite graph $G = (U, V, E)$. Recall that the graph G has $|U| = |V| = 2^n$ vertices on each side and has the property that every $2^k \times 2^k$ subgraph is balanced up to a fraction of δ edges.

Remark 3. *Better constructions of such explicit circuits (cf., Proposition 3) may be useful if we only have weaker lossy functions that do not compress much. They can also be used to get a hardness result for testing "weaker" properties (i.e., showing hardness for testing the* (k, δ)*-balanced property even for smaller values of* k *compared to* n*).*

We define our distribution over Boolean circuits that act on inputs from $\mathcal{B}^n \times \mathcal{B}^n$. Let $f'_{s'}(\cdot) = \mathsf{Eval}'(s', \cdot)$. First, we sample a description of a random function by $s' \leftarrow \mathsf{Gen}'(1^n, b)$ for $b \leftarrow \mathcal{B}$ chosen uniformly at random. Then, we output the following circuit $C_{s'}$ that represents a graph G': On input a pair of vertices $u \in U = \mathcal{B}^n, v \in V = \mathcal{B}^n$, output $B_{n,k,\delta}(f'_{s'}(u), f'_{s'}(v))$. That is, there is an edge between u and v iff there is an edge in $B_{n,k,\delta}$ between $f'_{s'}(u)$ and $f'_{s'}(v)$.

By item 1 in Claim 9 we know that the two distributions $s' \leftarrow \mathsf{Gen}'(1^n, b)$ for $b = 0$ and $b = 1$ are computationally indistinguishable. Next, we show that when $b = 0$ then the graph G' has the property Π, and when $b = 1$ the graph is $\epsilon = 1/4$ far from having the property Π, and conclude our theorem.

The injective case. When $b = 0$ we claim that G' is (k', δ)-balanced for $k' = 1.01k \leq 0.52n$. Consider any $2^{k'} \times 2^{k'}$ subgraph H of G'. Denote by L (resp. R) the set of edges on the left (resp. right) side of H. For $i \in [n]$, denote by A_i (resp. B_i) the values $y \in L$ (resp. $y \in R$) for which there are exactly i preimages under $f'_{s'}$, namely,

$$A_i = \{y \in L : |\{x : f'_{s'}(x) = y\}| = i\} \quad \text{and} \quad B_i = \{y \in R : |\{x : f'_{s'}(x) = y\}| = i\}.$$

By item 2 in Claim 9, (w.h.p.) for $i > n$ it holds that $|A_i| = 0$ so assume this is the case for the rest of the proof. Denote by I_L the set of indices i for which $|A_i| \geq 2^k \cdot i$ and similarly by I_R the set of indices j for which $|B_j| \geq 2^k \cdot j$. This implies that

$$\begin{aligned}\sum_{i\in I_L, j\in I_R} |A_i|\cdot|B_j| &= \sum_{i,j\in[n]} |A_i|\cdot|B_j| - \sum_{i\notin I_L \text{ or } j\notin I_R} |A_i|\cdot|B_j| \\ &\geq 2^{2k'} - n^2\cdot 2^{k'}\cdot 2^k \cdot n = 2^{2k'} - 2^{k'}\cdot 2^k \cdot n^3, \end{aligned} \tag{5.3.1}$$

where the inequality $\sum_{i\notin I_L \text{ or } j\notin I_R} |A_i|\cdot|B_j| \leq n^2\cdot 2^{k'}\cdot 2^k\cdot n$ follows since $|I_L|, |I_R| \leq n$ (giving the n^2 factor for the number of pairs $i \notin I_L$ or $j \notin I_R$) and either $|A_i|$ or $|B_j|$ is smaller than $2^k \cdot n$.

We will count the number of edges and non-edges between each A_i and each B_j for $i \in I_L$ and $j \in I_R$ (the sum of these will serve as a lower bound on the total number of edges and non-edges in H). Fix $i \in I_L$ and $j \in I_R$. Since $i, j \leq n$, in A_i (resp. B_j) there must exist at least $|A_i|/i \geq 2^k$ (resp $|B_j|/j \geq 2^k$) vertices whose values relative to $f'_{s'}(\cdot)$ are distinct. These vertices form a (k,δ)-balanced subgraph $\hat{H}$, namely, $\hat{H}$ contains at least

$$(|A_i|/i)\cdot(|B_j|/j)\cdot(1/2-\delta).$$

edges and non-edges (since they can be mapped to distinct vertices in $B_{n,k,\delta}$).

Counting the number of edges between A_i and B_j, any edge (and non-edge) in $\hat{H}$ will be counted exactly $i\cdot j$ times. Thus, the total number of edges and non-edges between A_i and B_j is at least

$$|A_i|\cdot|B_j|\cdot(1/2-\delta).$$

Thus, using Equation (5.3.1), the total number of edges in H is at least

$$\begin{aligned}\sum_{i\in I_L, j\in I_R} |A_i|\cdot|B_j|\cdot(1/2-\delta) &\geq (1/2-\delta)\cdot \sum_{i\in I_L, j\in I_R} |A_i|\cdot|B_j| \\ &\geq (1/2-\delta)\cdot 2^{2k'} - (1/2-\delta)\cdot 2^{k'}\cdot 2^k\cdot n^3 \\ &\geq (1/2-\delta)\cdot 2^{2k'} - 2^{k'+k+3\log n} \\ &= (1/2-\delta-2^{-k'+k+3\log n})\cdot 2^{2k'}.\end{aligned}$$

Since $k' = 1.01k$ and $k = 0.51n$, we have that $2^{-k'+k+3\log n} = 2^{-0.01k+3\log n} \leq 2^{-n/1000}$ for large enough n. Thus, letting $\delta' \triangleq \delta + 2^{-n/1000} \leq 2^{-n/2000}$, the number of edges in H is at least

$$\begin{aligned}\sum_{i\in I_L, j\in I_R} |A_i|\cdot|B_j|\cdot(1/2-\delta) &\geq (1/2-\delta-2^{-n/1000})\cdot 2^{2k'} \\ &= (1/2-\delta')\cdot 2^{2k'}.\end{aligned}$$

An analogous argument can be applied to show that the same lower bound holds on the number of non-edges which completes the proof.

The lossy case. When $b = 1$ we argue that the graph G' (represented by $C_{s'}$) is very far from satisfying the property Π. For any value y in the image of $f'_{s'}$ we define a non-empty set $A = f'^{-1}_{s'}(y)$. Since the image of $f'_{s'}$ is of size at most $2^{n/3}$, there are at most $2^{n/3}$ such disjoint sets of vertices $A_1, \ldots, A_{2^{n/3}} \subseteq U$. For any $i \in [2^{n/3}]$ we have that

$$\forall u_1, u_2 \in A_i\colon\ f'_{s'}(u_1) = f'_{s'}(u_2).$$

Similarly, there are at most $2^{n/3}$ disjoint sets of vertices $B_1, \ldots, B_{2^{n/3}} \subseteq V$ for which

$$\forall i \in [2^{n/3}], v_1, v_2 \in B_i\colon\ f'_{s'}(v_1) = f'_{s'}(v_2).$$

Let $I_L \subseteq [2^{n/3}]$ (respectively, $I_R \subseteq [2^{n/3}]$) be the subset of the A_i's (respectively, B_j's) which are of size at least $2^{k'}$. That is,

$$i \in I_L \iff |A_i| \geq 2^{k'} \quad \text{and} \quad j \in I_R \iff |B_j| \geq 2^{k'}$$

Each pair of sets A_i, B_j for $i \in I_L, j \in I_R$, gives us a set of vertices on the left and a set of vertices on the right which are either fully connected (if $B_{n,k,\delta}(f_s(u), f_s(v)) = 1$ for $u \in A_i$ and $v \in B_j$) or fully disconnected (if $B_{n,k,\delta}(f_s(u), f_s(v)) = 0$). Thus, if both A_i and B_j are of size at least $2^{k'}$, each such pair contributes $1/2 \cdot |A_i| \cdot |B_j|$ edges to G' that must be changed (added or removed) in order for the graph G' to posses the property Π. The number of nodes that are in a set A_i (and similarly for B_j) which is smaller than $2^{k'}$ is bounded by $2^{n/3} \cdot 2^{0.52n} \leq 2^{0.9n}$. Thus, we get that the number of edges that must be changed for the graph G' to posses the property Π is at least:

$$\begin{aligned}
\sum_{i \in I_L, j \in I_R} \frac{1}{2}|A_i| \cdot |B_j| &= \frac{1}{2} \sum_{i \in I_L, j \in I_R} |A_i| \cdot |B_j| + \frac{1}{2} \sum_{i \notin I_L, j \notin I_R} |A_i| \cdot |B_j| - \frac{1}{2} \sum_{i \notin I_L, j \notin I_R} |A_i| \cdot |B_j| \\
&= \frac{1}{2} \sum_{i \in U, j \in V} |A_i| \cdot |B_j| - \frac{1}{2} \sum_{i \notin I_L, j \notin I_R} |A_i| \cdot |B_j| \\
&\geq \frac{1}{2} \cdot 2^{2n} - \frac{1}{2} \cdot (2^{0.9n})^2 \geq \frac{1}{4} \cdot 2^{2n}.
\end{aligned}$$

Namely, at least $2^{2n}/4$ of the edges must be changed which is a constant fraction of the total number of edges in the graph. □

5.4 Impossibility of a General Transformation

In this section, we show (unconditionally) that there cannot be a general transformation from a black-box lower bound, to a white-box lower bound. That is, we show that there exists a problem that has exponentially high black-box complexity, however, is solvable in polynomial time given any white-box implementation of the search function.

We first give an informal overview of the problem we define. Consider the problem of finding a small circuit that is consistent with a large set of pairs (x_i, y_i). In particular, the set will be larger than the size of the circuit. In the black-box model, these points will be completely random and thus the task of finding a small circuit that is consistent is impossible (since one cannot compress random bits). On the other hand, in the white-box model, given any circuit that computes these points, the task becomes completely trivial: simply return the circuit in hand.

This approach raises two main difficulties. First, this problem does not always have a solution in the black-box model (which is not consistent with the definition of a search problem). Second, the solution has no a priori bound on its size.

The first problem is solved by taking any search problem with proven high black-box complexity (e.g., PPP or PWPP). Notice that this problem might have high white-box complexity as well. Then, we modify our search problem to be an OR of the two problems. That is, either find a small consistent circuit or solve the second search problem. In the black-box model, the complexity of the new problem remains high, and moreover, a solution always exists. In the white-box model, the problems remains solvable in polynomial time.

The second problem is solved as by instead of giving the circuit as a solution, giving a short *commitment* to the circuit and then proving that this commitment to a circuit is consistent on a random value. To achieve this, we use techniques such as Kilian's protocol combined with the Fiat-Shamir paradigm to remove interaction in the random oracle model (in the black-box model we have a random oracle!).

The search problem we define is the one considered by Goldwasser and Kalai [GK03] in the context of showing limitations for the Fiat-Shamir paradigm. Goldwasser and Kalai showed that there exists a 3-round public-coin identification scheme for which the Fiat-Shamir paradigm yields an insecure digital signature with any hash function in the standard model. (In contrast, Pointcheval and Stern [PS96] showed that in the random oracle model this paradigm always produces a signature scheme that is secure against chosen message attacks.)

The signature scheme of Goldwasser and Kalai naturally gives rise to a search problem: Given the public parameters of the scheme, find a valid signature for an arbitrary message. To make this problem in TFNP, we define the problem of either finding a valid signature as above or finding a collision in a compressing function. The latter has a guaranteed solution so this defines a valid search problem in TFNP.

The underlying relation. Underlying the construction of Goldwasser and Kalai [GK03] is a relation in which the instance is a function f and a witness C is a small circuit the approximates f: it agrees with f on a point the depends on the description of C. For a function $f \in \mathcal{F}_n$ where $f\colon \mathcal{B}^n \to \mathcal{B}^n$ the relation is:[12]

$$\mathcal{R}^f_{GK} = \{(f, C) \mid a = \mathsf{COM}^f(\tilde{C}) \ \wedge \ C(a) = f(a) \ \wedge \ |\tilde{C}| \leq 2^{n/10}\},$$

[12]The bound on the size of $\tilde{C}$ in [GK03] can be any super-polynomial function.

where $\tilde{C}$ is a specific error-correcting encoding of C and COM^f is the tree-commitment of Kilian [Kil92] which allows for a fixed polynomial-size commitment of any polynomial-size string and also allows for efficient decommitment for individual bits (here f is used as a hash function by ignoring half of the output bits).

Goldwasser and Kalai showed that when f is a random function (modelled as a oracle model), it is hard to devise a proof that one has a witness for membership in the relation with fewer than exponentially-many queries to f (with high probability). The high level idea is that a valid witness C will agree with f on the point $a = \mathsf{COM}^f(\tilde{C})$. Since f is random, the point a is also random, and hence C is a (small) circuit that approximates (the random oracle) f which does not exist. Therefore, finding such a witness is infeasible.

However, in the white-box model, given the code of f it is easy to find a proof (the code of f is used as a witness). There are two problems with using the above as a valid search problem in TFNP: (1) there is no guaranteed solution in the black-box setting, and (2) there is no guaranteed solution in the white-box setting if the circuit implementing f is of size larger than $2^{n/10}$. To solve this we allow to find solutions of a different kind: a pair of strings $a \neq b$ such that $f(a) = f(b)$ or $f(a) = 0$. This is exactly the complete problem for the class PPP. Namely, we OR with the relation:

$$\mathcal{R}^f_{\mathsf{PPP}} = \{(a,b) \in \mathcal{B}^n \times \mathcal{B}^n \mid a \neq b \ \wedge \ f(a) = f(b) \vee f(a) = 0\}.$$

This indeed solves both problems as now (1) there is always a guaranteed solution in the black-box setting since a compressing function must have a collision (the pigeonhole principle), and (2) such a solution can always be found with 2^n queries which is polynomial in $2^{n/10}$. To summarize, our final relation which is a search problem in TFNP is:

$$\mathcal{R}^f = \mathcal{R}^f_{GK} \cup \mathcal{R}^f_{\mathsf{PPP}}.$$

5.5 The Succinct Black-Box Model

We define and study a new model of computation which we call succinct black-box. In this model, as in the black-box model, the solver has only query access to the object and it is measured by the number of queries it performs in order to find a solution. However, in this model (as opposed to the black-box model), the number of queries is measured as a function of the size of the representation of the function. This is similar to the white-box model, where the running time is measured as a function of the size of the representation. In particular, if the function has a polynomial-sized representation, then an efficient algorithm in this model can perform only a polynomial number of queries (but the running-time may be arbitrary).

We show that for any problem in TFNP, there exists a deterministic procedure that performs only a *polynomial* number of queries (in the size of the representation of the function) and finds a valid solution.

Theorem 20. *For any search problem $\mathcal{S} \in$ TFNP it holds that* $\mathsf{sbbc}(\mathcal{S})$ *is polynomial. In particular, if the representation size is s and any solution is of size at most k, then the number of queries is $O(sk/\log k)$.*

The assumption that the search problem is in TFNP is essential for the theorem to hold. To see this, consider the case of *point functions* (functions that output 1 at a specific point and 0 everywhere else) where the goal is to find the hidden point. There exists a succinct representation (the point itself) but any algorithm that is only allowed to query the oracle must make exponentially many queries until it finds the hidden point.

Proof of Theorem 20. We construct an algorithm A in the succinct black-box model. Let $\mathcal{S}_n$ be a search problem where any solution is of size at most k (recall that k is polynomial in n). Suppose we are explicitly given the size $s = s(n)$ of the representation (we will get-rid of this assumption later). We begin by showing a simple version of the algorithm which performs $O(sk)$ queries:

Succinct black-box algorithm:

1. Initiate a list L of all possible representation of length less than s.
2. Repeat until a solution is found:
   (a) Define $f^*\colon \mathcal{B}^n \to \mathcal{B}^n$ such that $f^*(x) = \mathsf{MostFrequent}_{f\in L} f(x)$.
   (b) Find a solution $x_1, \ldots, x_k$ relative to f^*.
   (c) Query $x_1, \ldots, x_k$.
   (d) If all query results are consistent with f^*, then output $x_1, \ldots, x_k$.
   (e) Remove any f from L if it is not consistent with $x_1, \ldots, x_k$.

The algorithm always outputs a valid solution while performing at most a polynomial number of queries (in the size of the representation of f): First, for any f^* such a solution exists since the problem is in TFNP. In each round, the algorithm performs $k = \mathsf{poly}(n)$ queries. Since f^* is consistent with the most frequent value, if a solution is not found at any round then we get that at least half of the candidates are eliminated, and thus the list L is cut by half. The initial size of L is bound by 2^s. Thus, the total number of round is at most s. Overall, the total number of queries is at most ks.

Suppose now that we are not given the bound s on the size of the representation. Then, we do a variant of binary search: we run the algorithm with alleged upper bounds $s^* = 1, 2, 4, \ldots$ until it halts. When we run it with $s^* \geq s$ the algorithm will halt. The total cost is at most $k + 2k + 4k + \cdots + 2sk \leq 4ks = \mathsf{poly}(s)$ and thus $\mathsf{sbbc}(\mathcal{S}_n)$ is polynomial.

To get the bound $O(sk/\log k)$ we slightly fine-tune the above algorithm. We introduce a parameter α (which we set later) and add an extra step to the iteration (after step 2.1):

- While there exist an x such that $\Pr_{f \in L}[f(x) = f^*(x)] \leq 1 - \alpha$ query x and remove all inconsistent f from L.

In this case, we can average between single queries that eliminates α fraction of the candidates and k queries that together remove $1 - \alpha$ fraction of the candidates. To even these two cases, we set α such that $(1-\alpha)^k = \alpha$. Once α satisfies this equation, we get that after s/α iterations the number of remaining candidates is at most $2^s(1-\alpha)^{s/\alpha} \leq 2^s e^{-s} \leq 1$. Thus, the total number of queries is s/α. It is hard to get an exact solution to the above equation, however, a good enough approximation yields that $\frac{\log k}{2k} \leq \alpha \leq \frac{2\log k}{k}$. Plugging this in, we get that the number of queries is $s/\alpha = O(sk/\log k)$ as required. □

Goldberg and Roth [GR16, Theorem 3.3] investigated the number of queries needed to find an ϵ-well supported Nash equilibrium in multi-player games. They showed that if the game has a succinct representation, then there is an algorithm that performs a polynomial number of queries in the number of players n, the number of actions m, the description length of the target game, and finds such an equilibrium. One can view Theorem 20 as a generalization of that result.[13]

Efficient-succinct black-box model. We observe that our algorithm is inefficient, and thus is not applicable in the efficient-succinct black-box model (see Chapter 2 and Table 2.1). Moreover, there exist problems that are hard in the efficient-succinct black-box model, but are easy in the white-box model. This follows by adapting the proof from Section 5.4 while replacing the use of a random oracle with a pseudorandom function[14].

5.6 Further Research

The immediate direction this work raises is which other Ramsey-type problems are hard in the white-box model. Consider, for instance, Schur's Theorem that states that for every positive integer m, there exists a number $S(m)$ such that for every coloring of the numbers in the set $\{1, \ldots, S(m)\}$ with m colors, there must be x, y and z colored with the same color such that $x + y = z$ (see [GRS90], Chapter 3). This property naturally gives rise to the *m-Schur search problem*: Given an implicit representation of the coloring of the numbers $\{1, \ldots, S(m)\}$, find x, y and z colored with the same color and satisfy $x + y = z$. Can we argue that the m-Schur problem is hard?

What are the minimal assumptions needed to obtain the hardness results for Ramsey? Are one-way functions sufficient or is there an inherent reason why collision resistant hash functions are needed? For the *bipartite* Ramsey problem, we showed that a relaxation of CRH functions (MCRH functions) is necessary and sufficient.

[13] We thank Aviad Rubinstein for referring us to [GR16].

[14] A pseudorandom function is a (keyed) function that cannot be distinguished from a random function by any polynomially bounded adversary

Our results are "obfuscation-free", in the sense that we needed much weaker primitives for obtaining them than in the recent works of [BPR15, GPS16, KS17b]. Can we get similar results for showing the hardness of complexity classes such as PLS and PPAD?

We showed the general impossibility of transferring black-box results to white-box results. One direction which might be fruitful is to find conditions on the search problems that *do* allow for such general transformation from black-box to white-box. A natural candidate is when the search problem is defined over graphs, and we are looking for a graph property (i.e., the decision of $\mathcal{S}$ whether to accept or not depends solely on the presented subgraph and not on the names of the vertices). Can we prove a transformation in this case? Can we show an impossibility?

5.7 Chapter Appendix

5.7.1 Hardness of Finding a Sunflower

The famous Sunflower lemma, discovered by Erdös and Rado [ER60], asserts that in a sufficiently large family of sets of the same size a configuration called "sunflowers" must occur. In this section we show that even though the configuration is guaranteed to exist, efficiently finding it is hard assuming that collision resistant hash functions exist.

Definition 39 (A Sunflower)**.** *A* sunflower *with k petals and core Y is a collection of sets $S_1, \ldots, S_k$ such that*

1. *$S_i \cap S_j = Y$ for every distinct $i, j \in [k]$ and*
2. *$S_i \setminus Y \neq \emptyset$ for every $i \in [k]$.*

Lemma 7 (The Sunflower lemma)**.** *Let $\mathcal{F}$ be a collection of distinct sets each of cardinality s. If $|\mathcal{F}| > s!(k-1)^s$, then $\mathcal{F}$ contains a sunflower with k petals.*

We define the sunflower problem as a search problem in which one is given a collection of sets $\mathcal{F}$, each of size n, and the goal is to find a sunflower with $n+1$ petals. A circuit $C\colon \mathcal{B}^{2n\log n} \to (\mathcal{B}^n)^n$ represents the collection $\mathcal{F}$ of $2n\log n$ sets, each of size n. Namely, given such a circuit, the i^{th} set is given by evaluating $C(i) \in (\mathcal{B}^n)^n$. Therefore, the problem is formulated as follows.

Problem 1 (The Sunflower problem)**.** *Given a circuit $C\colon \mathcal{B}^{2n\log n} \to (\mathcal{B}^n)^n$ find $n+1$ indices such that $C(i_1), \ldots, C(i_{n+1})$ are a sunflower or two indices i_1, i_2 such that $C(i_1) = C(i_2)$.*

For any circuit C, either there exist two indices that encode the same set or the circuit C encodes $2^{2n\log n} = n^{2n} \geq 2n! \cdot n^n$ different sets (for $n > 1$). Then, according to Lemma 7, such a collection will contain a sunflower of size $n+1$ and therefore this is a valid search problem in TFNP

We say that the sunflower problem is hard if there exists an efficiently samplable distribution $\mathcal{D} = \{C\colon \mathcal{B}^{2n\log n} \to (\mathcal{B}^n)^n\}$ over circuits that succinctly represent a collection of sets as above, such that for every probabilistic polynomial-time algorithm A, there exists a negligible function $\mathsf{negl}(\cdot)$ such that

$$\Pr_{C\leftarrow\mathcal{D}}[i_1,\ldots,i_{n+1} \leftarrow A(1^n, C);\ C(i_1),\ldots,C(i_{n+1}) \text{ are a sunflower}] \leq \mathsf{negl}(n),$$

where the probability is over the uniform choice of $C \leftarrow D$ and the randomness of A.

Theorem 21. *The sunflower problem is hard assuming the existence of collision resistant hash functions.*

Proof. We first construct a large set $\mathcal{F}_{\mathsf{no}}$ in which there is no sunflower of size $n+1$ (our construction is taken from Exercise 6.2 in [Juk11]). Fix arbitrary n pairwise disjoint sets $T_1,\ldots,T_n \subseteq \mathcal{B}^n$ each of size n and consider the family $\mathcal{F}_{\mathsf{no}}$ of sets of size n such that every set has exactly one element from each T_i. That is,

$$\mathcal{F}_{\mathsf{no}} = \{S \in (\mathcal{B}^n)^n \mid \forall i \in [n]\colon |S \cap T_i| = 1\}.$$

First, we observe that the family $\mathcal{F}_{\mathsf{no}}$ is of size n^n. Second, the fact that the family $\mathcal{F}_{\mathsf{no}}$ has no sunflowers of size $n+1$ follows from the pigeonhole principle: for any $j \in [n]$ there must be two sets that contain the same element in T_j. Thus, the core Y must contain an element in each of the T_j's which leaves its petals empty. Lastly, we observe that $\mathcal{F}_{\mathsf{no}}$ has a succinct representation as a circuit: given a number $\ell \in [n^n]$, one can obtain the ℓ^{th} set of $\mathcal{F}_{\mathsf{no}}$ by first writing ℓ in base n as $\ell = (\ell_1,\ldots,\ell_n)$ and then outputting the set $\{T_j[\ell_j]\}_{i\in[n]}$ (where $T_j[\ell_j]$ is the ℓ_j-th element of T_j).

We proceed with the construction of the hard distribution for the sunflower problem. To sample a succinct representation of a collection of sets, we sample a collision resistant hash function $h\colon \mathcal{B}^{2n\log n} \to \mathcal{B}^{n\log n}$ and define the circuit $C \otimes h\colon \mathcal{B}^{2n\log n} \to (\mathcal{B}^n)^n$ as: Given an index $i \in \mathcal{B}^{2n\log n}$, we first hash it down to $h(i) \in \mathcal{B}^{n\log n}$ and then return the $h(i)$-th element of $\mathcal{F}_{\mathsf{no}}$. The description of this circuit is polynomial in n and it consists of an evaluation of a description of $\mathcal{F}_{\mathsf{no}}$ (which is polynomial in n) and a single evaluation of h.

Solving the sunflower problem for C can be reduced to finding collisions relative to h. Indeed, assume that there is an efficient adversary A that succeeds in solving the sunflower problem. First, suppose the adversary found two indices i_1, i_2 such that $C(i_1) = C(i_2)$. Since all the sets in $\mathcal{F}_{\mathsf{no}}$ are distinct it must be that $h(i_1) = h(i_2)$ and we have found a collision in h. Now suppose that A found a sunflower and denote the indices of the petals by $i_1,\ldots,i_{n+1}$. Also denote $i'_1 = h(i_1),\ldots,i'_{n+1} = h(i_{n+1})$. Either some two i'_j's are identical, or they are all different. In the latter case, it must be that $i'_1,\ldots,i'_{n+1}$ are indices of $n+1$ petals in $\mathcal{F}_{\mathsf{no}}$, but these do not exist so this cannot happen. The former, where $i'_j = i'_{j'}$ for some $j, j' \in [n+1]$, immediately gives a collision relative to h. □

5.7.2 An upper bound on colorful Ramsey

We restate and prove Propositions 8 and 9.

Proposition 11 (Restatement of Proposition 8)**.** *For every $k > 2$ and $m > 1$, it holds that $R(\underbrace{k,\ldots,k}_{m \text{ times}}) \leq m^{mk}$.*

Proposition 12 (Restatement of Proposition 9)**.** *Consider the full graph on N vertices. For every $k < \log N$, and every coloring $\psi\colon \binom{N}{2} \to [m]$, where $m = \frac{(\log N)/k}{\log\log N - \log k}$, there exists a monochromatic subgraph of size k.*

Proof of Proposition 8. Notice that the function $R(\cdot)$ is monotone, namely, if $k'_i \leq k_i$ for every $i \in [m]$, then

$$R(k'_1,\ldots,k'_m) \leq R(k_1,\ldots,k_m).$$

By Proposition 7, we know that

$$R(k_1,\ldots,k_m) \leq \sum_{i=1}^{m} R(k_1,\ldots,k_{i-1},k_i-1,k_{i+1},\ldots,k_m).$$

Thus, the expansion of $R(\underbrace{k,\ldots,k}_{m \text{ times}})$ using this formula, can be viewed as a tree whose root is labeled by $R(\underbrace{k,\ldots,k}_{m \text{ times}})$ and each internal node is labeled by $R(k_1,\ldots,k_m)$. The property of the tree is that if a node is labeled by $R(k_1,\ldots,k_m)$, then any of its children, labeled by $R(k'_1,\ldots,k'_m)$, satisfies that there exists a $j \in [n]$ such that $k'_j = k_j - 1$ and $k'_i = k_i$ for every $i \in [m] \setminus \{j\}$. The leaves of the tree are those nodes labeled by values upper bounded by $R(\underbrace{2,\ldots,2}_{m \text{ times}})$. Notice that $R(\underbrace{2,\ldots,2}_{m \text{ times}}) = 2$.

The value of $R(\underbrace{k,\ldots,k}_{m \text{ times}})$ is thus the sum of all the leaves of the tree. Since the value at each leaf of the tree is 2, the value of $R(\underbrace{k,\ldots,k}_{m \text{ times}})$ is bounded by the number of leaves times 2. By the description above, the tree is an m-ary tree with depth at most $mk-1$. Hence, the total number of leaves in the tree is at most m^{mk-1} which means that

$$R(\underbrace{k,\ldots,k}_{m \text{ times}}) \leq m^{mk}.$$

□

Proof of Proposition 9. It is enough to show that $R(\underbrace{k,\ldots,k}_{m \text{ times}}) \leq N$. Thus, by Proposition 8, it is enough to show that $m^{mk} \leq N$. Indeed, plugging in the value of m from the proposition, we get that

$$\begin{aligned}
\log(m^{mk}) &= mk \cdot \log m \\
&= k \cdot \frac{(\log N)/k}{\log\log N - \log k} \cdot \log\left(\frac{(\log N)/k}{\log\log N - \log k}\right) \\
&= \frac{\log N \cdot (\log\log N - \log k) - \log N \cdot \log(\log\log N - \log k)}{\log\log N - \log k} \\
&= \log N - \frac{\log N \cdot \log(\log\log N - \log k)}{\log\log N - \log k} \leq \log N.
\end{aligned}$$

Thus, $m^{mk} \leq N$, as required. □

5.7.3 A lower bound for Ramsey graphs

We show that a random graph on n vertices is $(2 \cdot \log N)$-Ramsey with high probability. Furthermore, instead of sampling the graph uniformly at random, one can sample it via a limited independence family.

Proposition 13 (Restatement of Proposition 6)**.** *A graph on N vertices sampled via a $(2 \cdot \log^2 N)$-wise independent distribution is a $(2 \cdot \log N)$-Ramsey graph with probability $1 - 1/N^{\Omega(\log\log N)}$.*

Proof. We review the classical proof showing the existence of a $(2 \cdot \log N)$-Ramsey graph via the probabilistic method. Sample a random graph $G = (V, E)$ on $|V| = N$ vertices, and fix any set V' of k vertices. The probability (over G) that V' forms an independent set or a clique in G is at most

$$2 \cdot 2^{-\binom{k}{2}}.$$

Applying a union bound over the set of all such sets V', we get that G has a clique or independent set of size k with probability at most

$$\binom{N}{k} \cdot 2 \cdot 2^{-\binom{k}{2}} \leq \left(\frac{Ne}{k}\right)^k \cdot 2 \cdot 2^{-\binom{k}{2}}.$$

Plugging in $N = 2^{k/2}$ (and then $k = 2\log N$), we get that the above probability is bounded by at most

$$\begin{aligned}
2^{k \cdot \log(2^{k/2} \cdot e/k) - \binom{k}{2} + 1} &= 2^{k(k/2 + \log e - \log k) - \binom{k}{2} + 1} \\
&\leq 2^{k(k/2 + \log e - \log k) - (k^2/2 - k) + 1} \\
&\leq 2^{-k\log k + k\log e + k + 1} \\
&\leq 1/N^{O(\log\log N)}
\end{aligned}$$

for large enough N.

Observing the above proof, one can see that it remains valid even if G is sampled from a $(2 \cdot \log^2 N)$-wise independent distribution. Thus, we get that sampling a graph G on n vertices from a $(2 \cdot \log^2 N)$-wise independent distribution, results with a $(2 \cdot \log N)$-Ramsey graph with probability with probability at least $1 - 1/N^{\Omega(\log \log N)}$. □

Chapter 6

Collision Resistant Hashing for Paranoids: Dealing with Multiple Collisions

6.1 Introduction

In any function that compresses its input, say from $2n$ bits to n bits, there are many collisions, that is, pairs of distinct inputs whose image is the same. But what is the complexity of finding such a collision? Families of functions where this collision finding task is hard are known as collision resistant hash (CRH) functions.[1] CRH functions have many appealing properties, such as preservation under composition and concatenation. The presumed hardness of finding collisions in such functions is the basis for increased efficiency of many useful cryptographic schemes, in particular signature schemes and succinct (zero-knowledge) arguments, i.e., methods for demonstrating the correctness of a statement that are much shorter than the proof or even the statement itself (see Kilian [Kil92] and Barak and Goldreich [BG08]). The latter is achieved via a hash tree commitment[2] scheme whose opening is local i.e., opening a bit does not require revealing the full string (known as a "Merkle tree"). Results of this sort enable the construction of efficient delegation of computation where the goal is to offload significant computation to some server but also to verify the computation.

Such a task ("breaking a collision resistant hash function") is indeed hard based on a variety of assumptions such as the hardness of factoring integers, finding discrete logs in finite groups or learning with errors (LWE). There are popular functions

[1] The function $h \in H$ can be sampled efficiently, it is easy to compute $h(x)$ given h and x, however, given h it is hard to find $x_1 \neq x_2$ s.t. $h(x_1) = h(x_2)$.

[2] A commitment scheme is a protocol where a sender commits to a string x in the "commit" phase and that later can be revealed at the opening phase. The two properties are binding and hiding: in what sense is the sender bound to the string x (computationally or information theoretically) and in what sense is x hidden from the receiver before the opening - statistically or computationally.

(standards) with presumed hardness of collision finding such as SHA-2 and SHA-3 (adopted by NIST[3] in 2015). These functions can be evaluated very quickly; however, their hardness is based on more ad hoc assumptions and some former standards have been shown to be insecure (such as MD4, MD5, SHA-1). On the other hand there is no known construction of CRHs based solely on the existence of one-way functions or even one-way permutations and, furthermore, they were shown to be separated in a black-box model (see Simon [Sim98]).

But a sufficiently compressing function also assures us that there are **multiple collisions**, i.e., k distinct values whose image under the function is equal. What about the problem of finding a k-collision? Assuming such hardness is a *weaker* computational assumption than hardness of finding a single pair of colliding inputs and the question is whether it yields a useful primitive.

In this paper we deal with *multiple collision resistant hash* (MCRH) functions and systematically investigate their properties and applications. We show that for some of the major applications of CRH functions it is possible to replace them by an MCRH, albeit at the price of adding some rounds of interaction: **a constant-round[4] commitment scheme with succinct communication that can be opened locally** (without revealing the full string) (see Theorem 22). This implies that it is possible to effectively verify the correctness of computation much more efficiently than repeating it. As an application we get universal arguments [BG08] (and public-coin zero-knowledge argument systems for NP [Bar01]) with an arbitrary super-constant number of rounds based on MCRH functions. We also provide a constant-round statistically-hiding scheme and thus we can get constant-round statistical zero-knowledge arguments [BCC88].[5]

On the other hand, we show various **black-box separation** results concerning MCRH. First, we separate them from one-way permutations. This follows from the lower bound of Haitner et al. [HHRS15] on the number of rounds needed to build a statistically-hiding commitment from one-way permutations. Furthermore, we show a black-box separation from standard CRH: there is no fully black-box construction of a k-MCRH from a $(k+1)$-MCRH for all k with polynomial security loss (see Theorem 24). These results yield an infinite hierarchy of natural cryptographic primitives, each two being separated by a fully black-box construction, between one-way function/permutations and collision-resistant hash function.[6]

One motivation for investigating MCRH functions is the progress in finding collisions in the hash function SHA-1 (that has long been considered insecure [WYY05]) and recently an actual meaningful collision has been found [SBK+17]. In general,

[3]NIST is the National Institute of Standards and Technology, a US agency.

[4]By "constant-round" we mean that for a $c \in \mathbb{N}$ to commit to a string of length n^c using a compression function from $2n$ to n bits the number of rounds is a constant that depends on c.

[5]For constant values of k, we give a 4-round computationally-binding commitment scheme with succinct communication (see Theorem 23).

[6]This hierarchy translates to an infinite hierarchy of natural subclasses in TFNP. See Section 6.8.2 for details.

finding a collision with arbitrary Initialization Vector (IV) allows finding multiple collisions in an iteration of the function (say in a Merkle-Damgård lopsided tree), as shown by Joux [Jou04] (see also Coppersmith and Girault et al. [Cop85, GCC88] for older attacks). However, for the compression function of SHA-1 (or other such functions) there is no non-trivial algorithm for finding multi-collisions.[7] Also, multi-collision resistance is sometimes useful for optimizing concrete parameters, as shown by Girault and Stern [GS94] for reducing the communication in identification schemes. So the question is what can we do if all we assume about a given hash function is that multi-collision are hard to find, rather than plain collisions.

Our interest in MCRH functions originated in the work of Komargodski et al. [KNY17], where MCRHs were first defined in the context of the bipartite Ramsey problem. They showed that finding cliques or independent sets in succinctly-represented bipartite graphs (whose existence is assured by Ramsey Theory) is equivalent to breaking an MCRH: a hard distribution for finding these objects implies the existence of an MCRH and vice-versa (with slightly different parameters).[8]

Families of CRHs compose very nicely, and hence *domain extension* is relatively simple, that is, once we have a CRH that compresses by a single bit we can get any polynomial compression (i.e., from $\mathsf{poly}(n)$ to n bits). In contrast, we do not know how to construct a k'-MCRH on very large domains from a fixed k-MCRH, where k' is not much larger than k. Nevertheless, in Section 6.5, we show how to get such a construction with $k' = k^{O(\log n)}$.

A well-known relaxation of CRHs are known as Universal One-way Hash Functions (UOWHF) or second pre-image resistance: first the target x is chosen (perhaps adversarially), then a function $h \in_R H$ is sampled and the challenge is to find $x' \neq x$ s.t. $h(X) = h(x')$. Such families are good enough for signatures (at least existentially) and can be used for the hash-and-sign paradigm, if one chooses h per message (see Naor and Yung [NY89] and Mironov [Mir06]). It is known how to get such family of functions from one-way functions, but the construction is rather involved and inefficient (and there are some inherent reasons for that, see [GGKT05]). We show how to go from *any* interactive commitment protocol where the communication is shorter than the string committed to (a succinct commitment) to a UOWHF (see Theorem 25). Together with our commitment schemes, this gives new constructions of UOWHFs on long inputs based on MCRHs with a shorter description than the ones known starting from a UOWHF on fixed input length.

The four worlds of hashing. Impagliazzo's five worlds [Imp95] are a way to characterize the strength of a cryptographic assumption. The worlds he defined are:

[7] Beyond the "birthday-like" algorithm that will take time $2^{n \cdot \frac{k-1}{k}}$ [Jou04], where 2^n is the size of the range. See [HSX17] for very recent work in the quantum case.

[8] In the bipartite Ramsey problem, the goal is to find a bi-clique or bi-independent set of size $n/4 \times n/4$ in a bipartite graph of size $2^n \times 2^n$. The work of [KNY17] showed that if this problem is hard, then there exists an $(n/4)$-MCRH from n bits to $n/2$ bits. Conversely, If a $\sqrt{n}$-MCRH mapping n bits to $\sqrt{n}/8$ bits exists, then this problem is hard.

Algorithmica (where $\mathsf{P} = \mathsf{NP}$), *Heuristica* (where NP is hard in the worst case but easy on average, i.e., one simply does not encounter hard problems in NP), *Pessiland* (where hard-on-the-average problems in NP exist, but one-way functions do not exist), *Minicrypt* (where one-way functions exist), and *Cryptomania* (where Oblivious Transfer exists). (Nowadays, it is possible to add a sixth world, *Obfustopia*, where indistinguishability obfuscation for all programs exists.)

In the spirit of Impagliazzo's five worlds of cryptographic assumptions, we define four worlds of hashing-related primitives:

Nocrypt: A world where there are no one-way functions. There are no cryptographic commitments of any kind in this world [IL89].

Unihash: A world where one-way functions exist (and so do UOWHFs), but there are no MCRH functions. Therefore, signatures exist and hashing applications such as the hash-and-sign paradigm [NY89]. Also, statistically-hiding commitments exist (albeit with a linear number of rounds) [NOVY98, HHK+09, HNO+09, HHRS15]. There are no known short commitment (where the communication is much shorter than the string committed to).

Minihash: A world where MCRH exists but there is no CRH: that is, for some polynomial $k(n)$ there exists a k-MCRH that compresses $2n$ to n bits. In this work we give a protocol for short and statistically-hiding commitments with a *constant* number of rounds. Furthermore the string can be opened locally, with little communication and computation, without requiring the full opening of the string.

Hashomania: A world where CRH exists. There is a short commitment protocol that requires only *two* rounds (i.e., two messages) with local opening. This is the famed Merkle-tree.

Note that our separation results imply that these four worlds have black-box separations. Unihash and Minihash are separated by the separation of MCRH from one-way permutations, and Minihash and Hashomania are separated by the separation of CRH from MCRH. Moreover, the separation in Section 6.6 actually implies that the world *Minihash* can be split further into sub-worlds parameterized by k, the number of collisions it is hard to find.

Multi-pair collision resistance. A different way to relax the standard notion of collision resistance is what we call *multi-pair-collision-resistance*, where the challenge is to find *arbitrary* k distinct pairs of inputs that collide (possibly to different values). One may wonder what is the difference between these two notions and why we focus on the hardness of finding a k-wise collision rather than hardness of finding k distinct colliding pairs. The answer is that the notion of k-pair-collision-resistance is *existentially equivalent* to the standard notion of collision resistance (see Section 6.8.1).

Concurrent work

In parallel to this work, MCRH functions were studied by two other groups that obtained various related results [BDRV17, BKP17] with different motivation and perspective.

Berman et al. [BDRV17] showed how to obtain MCRH functions from a specific assumption. Concretely, they construct an n^2-MCRH function compressing inputs of length n to outputs of length $n - \sqrt{n}$ from the average-case hardness of the min-max variant of the entropy approximation problem. This variant is a promise problem where the YES inputs are circuits whose output distribution has *min*-entropy at least κ, whereas NO inputs are circuits whose output distribution has *max*-entropy less than κ.[9] Berman et al. also show how to get a constant-round statistically-hiding (computationally-binding) commitment scheme from any k-MCRH that compresses n bits into $n - \log k$ bits (which implies a black-box separation of MCRH from one-way permutations). However, their commitment scheme is not short and does not support local opening[10].

Bitansky et al. [BKP17] replace the CRH assumption with a k-MCRH in several applications related to zero-knowledge and arguments of knowledge for NP with few rounds. They rely on either MCRH that compresses by a polynomial factor (say n^2 bits into n), or alternatively on an MCRH that compresses by a linear factor but lose a quasi-polynomial factor in security. Their main technical component is a two-round (i.e., two-message) short commitments with local opening but with *weak* computational-binding. The latter means that the sender may be able to open the commitment to more than one value, but not to too many values. Their construction of the commitment scheme is related to our construction presented in Theorem 23 but they design a specific code that allows them to get local opening.

Summary of results & paper organization

Our main results are:

1. Any k-MCRH can be used to get a constant round short commitment scheme which is computationally-binding, statistically-hiding and support local opening (à la Merkle commitments). This result in Section 6.4.

2. Any k-MCRH, where k is constant, can be used to get a 4 round short commitment scheme which is computationally-binding and statistically-hiding opening. This appears in Section 6.5.

3. We prove a fully black-box separation between standard collision resistant hash functions and multi-collision resistant ones. This appears in Section 6.6.

[9] The original *entropy approximation* problem is the one obtained by replacing the min- and max-entropy with the Shannon entropy. It is known to be complete for the class of languages that have non-interactive statistical zero-knowledge proofs (NISZK) [GSV99].

[10] Starting out with such a weak primitive our methods will not yield a succinct commitment either.

4. We present a generic and direct construction of UOWHFs from any short commitment schemes (and thereby from any multi-collision resistant functions). See Section 6.7.

In Section 6.2 we provide an overview of our main ideas and techniques. In Chapter 2 and section 6.3 we provide preliminary standard definitions used throughout the paper and the definition of MCRH functions, respectively.

In Sections 6.8.1 and 6.8.2 we discuss the related notion of multi-pair collision resistance, and the implication of our infinite hierarchy of assumptions to TFNP, respectively.

6.2 Our Techniques

In this section we present some of our main ideas and techniques used in the construction of the commitment scheme and in the black-box separation result.

6.2.1 The main commitment scheme

A commitment scheme is a two stage interactive protocol between a sender and a receiver such that after the first stage the sender is bound to at most one value (this is called "binding"). In the second stage the sender can open his committed value to the sender. There are a few security properties one can ask from such a protocol: have statistical/computational binding, and have the committed value of the sender be statistically/computationally hidden *given* the commitment (this is called "hiding"). Our commitment scheme satisfies computational binding and statistical hiding. In this overview we will mostly focus on obtaining computational binding and briefly discuss how we obtain hiding towards the end.

There is a trivial commitment protocol that is (perfectly) binding: let the sender send its value to the receiver. Perhaps the most natural non-trivial property one can ask is that the commitment is *shorter* than the committed string. There are additional useful properties one can require such as *local-opening* which allows the sender to open a small fraction of its input without sending the whole string (local opening is very important in applications of such commitment schemes, e.g., to proof systems and delegation protocols). In our protocol the commitment is short and it supports local-opening; we will focus here on the former and shortly discuss the latter towards the end.

Our goal now is to construct a commitment scheme which is computationally binding and the commitment is shorter than the value of the sender. If we had a standard collision resistant hash function mapping strings of length $2n$ to strings of length n, this task would be easy to achieve: the receiver will sample a hash function h and send it to the sender which will reply with $h(x^*)$, where $x^* \in \{0,1\}^{2n}$ is its

value. The commitment thus consists of $(h, h(x^*))$ and its size is n bits.[11] It is easy to verify that for a sender to cheat during the opening phase it actually has to break the collision resistance of h (i.e., come up with a value $x \neq x^*$ such that $h(x) = h(x^*)$).

When h is only a k-MCRH for $k > 2$, the above protocol is clearly insecure: the sender can potentially find two inputs that collide to the same value and cheat when asked to open its commitment. The first observation we make is that even though the sender is not bound to a single value after sending $h(x^*)$, it is bound to a set of values of size at most $k-1$. Otherwise, at least intuitively, he might be able to find k inputs that map to the same output relative to h, which contradicts the security of the k-MCRH. Our first idea is to take advantage of this fact by adding an additional round of communication whose goal is to "eliminate" all but one possible value for the sender. Specifically, after the receiver got $h(x^*)$, it samples a random universal hash[12] function, g, mapping strings of length $2n$ to strings of length m. and sends it to the sender. The sender then responds with $g(x)$. The commitment thus consists of $(h, h(x^*), g, g(x^*))$. To open the commitment the sender just sends x^* (just as before).

One can show that this protocol is binding: Out of the $k-1$ possible values the sender knows that are consistent with $h(x^*)$, with probability roughly $k^2 \cdot 2^{-m}$ there will be no two that agree on $g(x^*)$. Conditioning on this happening, the sender cannot submit $x \neq x^*$ that is consistent with both $h(x^*)$ and $g(x^*)$. To formally show that the protocol is binding we need to show how to find a k-wise collision using a malicious sender. We simulate the protocol between the malicious sender and the receiver *multiple times* (roughly k times) with the same h but with freshly sampled g, by *partially rewinding* the malicious sender. We show that with good probability, every iteration will result with a new collision.

The protocol consists now of 4 rounds, but is the commitment short? Well, it depends on m and on the description size of g. The description size of a universal function is proportional to the input size ($2n$ in our case), which totally ruins the shortness of the protocol. We fix this by sampling g from an *almost* universal family and apply the above protocol. We obtain a 4-round protocol in which the commitment size is of the order roughly $n + m + \log(1/\delta)$, where δ is related to the error probability of the almost universal function. Choosing m and δ appropriately we obtain a protocol with short ($< 2n$) commitments.

Handling longer inputs. How would we commit on a longer string, say of $10n$ bits or even n^{10} bits? (Recall that all we have is a hash function mapping $2n$ bits into n bits.) One well-known solution is based on a standard collision resistant hash function, and what is known as a Merkle tree (the tree structure will be useful later for a local opening). The input $x \in \{0,1\}^{2^d n}$ (for simplicity think of d as either a large constant or even $O(\log n)$) is partitioned into 2^d blocks each of size n. These blocks are

[11] For simplicity of presentation here, we ignore the cost of the description of h, so it will not significantly affect the size of the commitment.

[12] A universal hash function is a function of families $\mathcal{G} = \{g \colon \{0,1\}^n \to \{0,1\}^m\}$ such that for any $x, y \in \{0,1\}^n$ such that $x \neq y$ it holds that $\Pr_{g \leftarrow \mathcal{G}}[g(x) = g(y)] \leq 2^{-m}$.

partitioned into pairs and the hash function is applied to each pair resulting in 2^{d-1} blocks. Then, the remaining blocks are partitioned into pairs and the hash function is applied on each pair. This is repeated d times resulting in a binary tree of hash values of depth d. The value associated with the root of the tree is called the root-hash.

If h is a standard CRH sent by the receiver, then it is known that sending the root-hash by the sender is actually a commitment on the input x^* [Mer89a, Kil92]. Can we apply the same trick from before to make this a commitment protocol even when h is only a k-MCRH? That is, after sending the root-hash, let the sender sample a good-enough combinatorial hash function $g\colon \{0,1\}^{2^d n} \to \{0,1\}^m$ and send it to the sender that will reply with $g(x^*)$. Is this protocol binding? The answer is "no", even for large values of m. Observe that for every node in the Merkle tree, the sender can potentially provide $k-1$ valid inputs (that hash to the same value). Since the tree is of depth d, one can observe that by a mix-and-match method of different colliding values on different nodes of the tree the sender might be able to come up with as many as $(k-1)^{2^d}$ valid inputs x whose corresponding root-hash is $h(x^*)$. Thus, to satisfy that no two have the same value under g, we have to choose $m \approx 2^d \cdot \log(k-1)$, in which case the almost uniform hash function has a pretty long description. Nevertheless, it is less than $2^d n$ (the input length) so we might hope that some progress has been made. Is this protocol computationally-binding? Not quite. Using the proof technique from above (of partially rewinding and "collecting" collisions) would require running the malicious sender more than $(k-1)^{2^d}$ times until it has to present more than $k-1$ collisions for some value. This is, of course, way too expensive.

The bottom line of the above paragraph is that "mix-and-match" attacks are very powerful for a malicious sender in the context of tree hashing by allowing him to leverage the ability of finding few collisions into an ability to find exponentially many collisions. The reason why this happens is that we compose hash functions but apply the universal hash function on the whole input as a single string. Our next idea is to apply a "small" hash function $g\colon \{0,1\}^{2n} \to \{0,1\}^n$ per node in the Merkle tree. That is, after the sender sends the root-hash $h(x^*)$, the receiver samples g and sends it to the sender. The sender computes $g(\cdot,\cdot)$ for every pair of siblings along the Merkle tree, concatenates them all and sends this long string back to the receiver. This protocol is more promising since, in some sense, we have a small consistency check per node in the tree which should rule out simple "mix-and-match" attacks. This is our construction and the proof of security works by partially rewinding a malicious sender and "collecting" collisions until we get k collisions with respect to some internal node in the tree (we need to collect roughly $2^d k$ collisions overall so that such a node exists, by the pigeonhole principle). How efficient is the protocol? Details follow.

The protocol still consists of 4 rounds. A commitment consists of the hash function h, the root hash $h(x^*)$, a universal hash function $g\colon \{0,1\}^{2n} \to \{0,1\}^m$ and the value of g on *every* internal node of the tree. The overall size is thus of order $n + 2^d m$.

Notice that $n + 2^d m \ll 2^d n$ whenever d is not too small, so we have made progress! We reduced the size of the commitment by a factor of m/n. The final step is to really get down to a commitment of size roughly n. To achieve this, we apply our protocol *recursively*: Instead of sending the hashes (with respect to g) of all internal nodes, we run our *commit* protocol recursively on this string. Notice that this string is shorter ($2^d m$ compared to $2^d n$) so the recursion does progress. The base of the recursion is when the string length is roughly n bits, then the sender can simply send it to the receiver.

Choosing the parameters carefully, we get various trade-offs between the number of rounds, the commitment size, and the security of the resulting protocol. For example, setting $m = n^{0.99}$, results with a $O(1)$-round protocol in which the commitment size is $O(n)$ (here the big "O" hides constants that depend on $\log_n(|x^*|)$ which is constant for a polynomially long x^*), and whose security is worse than the security of the k-MCRH by an additive factor of $\exp(-n^{0.99})$.

Local opening. Due to the tree structure of our commitment protocol, it can be slightly modified to support local opening. Recall that the goal here is to allow the receiver to send an index i of a block to the sender, who can reply with the opening of the block, with communication proportional to n but not to the number of blocks 2^d. The idea here is, given an index i of a block, to open the hash values along the path corresponding to the i-block along with the tree sibling of every node in the path. Then, i' is defined to be the index of the block in the shorter string (the string committed to in the next step of the recursion) which containing all the $g(\cdot,\cdot)$ values of the nodes on the path (we make sure that such a block exists). Then, we add the hash values of the path for block i' and continue in a recursive manner.

Statistical hiding. We show how to transform any short commitment scheme that is computationally binding (but perhaps not hiding) to a new scheme that is short, computationally binding and *statistically hiding*. Moreover, if the original scheme admits a local-opening, then the new scheme admits a local-opening as well. Our transformation is information theoretic, adds no additional assumptions and preserves the security, the number of rounds and communication complexity of the original scheme (up to a small constant factor). The transformation is partially based on ideas originating in the work of Naor and Yung [NY89, Section 5.2] and the follow-up works of Damgård, Pedersen, and Pfitzmann [DPP97, DPP98] giving constructions of statistical-hiding commitments from (standard) collision resistant hash function.

The idea of our transformation is to leverage the fact that the basic commitment protocol is short: when committing to a long string x^*, the communication is very short. Thus, a large portion of x^* is not revealed to the receiver by the protocol so this part of x^* is statistically hidden. The task that remains is to make sure that all of x^* is hidden. Thus, instead of committing to x^* directly, we commit to a random string r that is independent of x^* and slightly longer. Then, we extract from r the remaining randomness r' *given the communication of the protocol* using a strong extractor. Finally,

we commit on the string $x^* \oplus r'$. It is not hard to show that if the original scheme was computationally binding, then the new one is as well. The fact that the scheme is statistically-hiding follows from the use of the strong extractor and the fact that the commitment is short.

One problem with the recipe above, is that the protocol (as describe) no longer admits a local-opening. This is because to open an index i, we need the i-th output bit of the extractor, but computing this bit might require reading a large portion of the input of r. Our solution is to break the input to sufficiently small parts such that each part is small enough to fit in a local-opening but is long enough to have enough entropy (given the communication of the protocol) so that we can apply the extractor on it.

6.2.2 Separating Multi-CRH from standard CRH

We show barriers of constructing a collision-resistant hash function from a 3-multi-collision-resistant hash function. We rule out fully black-box constructions (see Definition 14). Our proof technique is inspired by the works of Asharov and Segev [AS16] and Haitner et al. [HHRS15], that are based in turn on ideas originating in the works of Simon [Sim98], Gennaro et al. [GGKT05] and Wee [Wee07]. However, when trying to adapt their proof to the setting of multi collisions one encounters several obstacles and we explain how to overcome them.

The high-level overview of the proof is to show that there exists an oracle Γ such that relative to Γ there exists a 3-MCRH, however there exist no standard CRH. Our oracle will contain a truly random function f that maps $2n$ bits to n bits. Relative to this oracle, it is clear that 3-MCRH exists, however, also standard CRH exist. We add an oracle ColFinder that will be used to break any CRH construction. The main difficulty of the proof is to show that this oracle cannot be used to break the 3-MCRH.

The oracle ColFinder is essentially the same as in Simon [Sim98]. It gets as an input a circuit C, possibly with f gates and it outputs two random elements w, w' such that $C(w) = C(w')$. It is easy to see that no family of hash functions can be collision resistant in the presence of such an oracle. A single call to ColFinder with the query C (where $C(x) = f(x)$) will find a collision with high probability. The main question is whether this oracle be used to find *multiple* collisions?

Originally, Simon showed that this oracle cannot be used to *invert* a one-way function (or even a permutation). Let $\mathcal{A}$ be an adversary that uses ColFinder to invert f on a random challenge $y = f(x)$. Clearly, if $\mathcal{A}$ make no calls to ColFinder then his chances in inverting y are negligible. Assume, for simplicity, that $\mathcal{A}$ performs only a single query to ColFinder. In order for $\mathcal{A}$ to gain some advantage, it must make an "interesting" query to ColFinder. That is, a query which results in w, w' and the computation of either $C(w)$ or $C(w')$ makes a direct query to some $x \in f^{-1}(y)$. This event is called a hit. An important point is that for any circuit C the marginal distribution of w and of w' is uniform. Therefore, the probability of the event "hit" in the ColFinder query is at most twice that probability when evaluating $C(z)$ for a random

z. Thus, we can construct a simulator that replaces $\mathcal{A}$'s query to ColFinder with the evaluation of $C(z)$ and hits an inverse of y with roughly the same probability as $\mathcal{A}$ (while making no queries to ColFinder). The task of inverting y without ColFinder can be shown to be hard, ruling out the existence of such a simulator and in turn of such an adversary $\mathcal{A}$.

Our goal is to extend this proof and show that ColFinder cannot be used to find 3-wise collisions. The above approach above simply does not work: specifically, in our case the event "hit" corresponds to query C to ColFinder that results in w, w' and the computation of $C(w)$ and $C(w')$ *together* make direct queries three elements x_1, x_2, x_3 that collide under f (i.e., $f(x_1) = f(x_2) = f(x_3)$). It might be the case that these three elements are hit by $C(w)$ and $C(w')$ combined, but never by one of them alone. Thus, when simulating the $C(z)$ for a random z we will hit only part of the trio x_1, x_2, x_3 and might never hit all three.

Our main observation is that since $\mathcal{A}$ finds a 3-wise collision, but ColFinder finds only a 2-wise collision (namely w, w'), by the pigeonhole principle, either w or w' will hit two of the three elements of the 3-wise collision. Again, since the marginals of w and w' each are uniform, we can construct a simulator that runs $\mathcal{A}$ and on the query to ColFinder samples a uniform z and compute $C(z)$ and will get a colliding x_1 and x_2 without performing any queries to ColFinder. Then, one can show that such a simulator cannot exist.

Several problems arise with this approach. First, notice that this does not extend to an adversary $\mathcal{A}$ that makes more than one query. In such a case, the resulting simulator finds a 2-wise collision x_1, x_2 without the "hit" event occurring (i.e., finding a 3-wise collision) but while performing several ColFinder queries. Such a simulator (that finds a collision), of course, *trivially* exists, and we do not get the desired contradiction. Nevertheless, we show that the collision found by our simulator is somewhat special, and using ColFinder one can only find "non-special" collisions, ruling out the existence of $\mathcal{A}$ in this case. Second, the event "hit" itself might be spread out within several ColFinder queries, where, for example, one queries finds x_1 and then another query finds x_2, x_3. We tailor a simulator for each case, and show that for each case the resulting simulating cannot exist, completely ruling out the possibility of $\mathcal{A}$ to exist.

6.3 Multi-Collision-Resistant Function Families

A a multi-collision-resistant hash function is a relaxation of standard collision-resistant hash function in which it is hard to find *multiple* collisions on the same value.

Definition 40 (Multi-Collision-Resistant Hashing)**.** *Let $k = k(n)$ be a polynomial function. An efficient function family ensemble $\mathcal{H} = \{\mathcal{H}_n \colon \{0,1\}^{2n} \to \{0,1\}^n\}_{n\in\mathbb{N}}$ is a (t, ϵ)-secure k-*multi-collision-resistant hash *(MCRH) function family if for any probabilistic al-*

gorithm $\mathcal{A}$ that runs in time at most $t(n)$, for large enough $n \in \mathbb{N}$:

$$\Pr\left[\begin{array}{c|c} x_1,\ldots,x_k \text{ are distinct and} & h \leftarrow \mathcal{H}_n \\ h(x_1)=\cdots=h(x_k) & (x_1,\ldots,x_k) \leftarrow \mathcal{A}(h)\end{array}\right] \leq \epsilon(n).$$

*We call such $x_1,\ldots,x_k$ that map to the same value under h a k-*wise collision*. Lastly, we say that $\mathcal{H}$ is a secure k-MCRH if it is $(p, 1/p)$-secure for every polynomial $p(\cdot)$.*

The compression ratio. In the definition above we assume that the hash function compresses its input from $2n$ bits into n, where the choice of the constant 2 is somewhat arbitrary. Our choice of linear compression rate (in contrast to, say, a polynomial compression rate) models the basic building blocks in most standards of cryptographic hash functions, such as the ones published by NIST (e.g., most of the SHA-x family).

When considering k-MCRH functions, a compression that eliminates less than $\log k$ bits is not of interest, since such a function exists unconditionally, say by chopping (there simply will be no k-wise collision).

The factor two compression is somewhat arbitrary as any k-MCRH that compresses $(1+\epsilon)n$ bits into n bits can be translated into a $(k^{1/\epsilon})$-MCRH that compresses $2n$ bits into n (e.g. via the Merkle-Damgård iterated construction [Mer89b, Dam89]). It is possible to assume an even stronger hash function that compresses by a polynomial factor, say n^2 bits into n bits (and this is sometimes useful; see the paragraph in the end of Section 6.5 for an example), but this is a strong assumption that we prefer to avoid.

For standard collision resistant hash function (with $k = 2$), it is known that composition allows to translate hash functions that compress by one bit into hash functions that compress by any polynomial factor (from n bits into n^δ bits for any constant $\delta > 0$). Obtaining a similar result for k-MCRH functions (with $k > 2$) without significantly compromising on the value of k in the resulting family is an open problem. In Section 6.5 we give a transformation in which the resulting family is $(k^{O(\log n)})$-MCRH.

Public vs. private coins. Our definition above is of a private-coin MCRH, namely, the coins used by the key-generation procedure are not given to the collision finder, but are rather kept secret. One can define the stronger public-coin variant in which the aforementioned coins are given to the attacker. The weaker notion is enough for our applications. There are (other) cases where this distinction matters, see Hsiao and Reyzin [HR04].

6.4 Tree Commitments from Multi-CRH

We show how to build a commitment scheme which is computationally-binding, statistically-hiding, round-efficient, has short commitments, and supports local open-

ing. We refer to Section 2.2.3 for the definition of a commitment scheme, computational-binding, statistical-hiding, and the efficiency measures of commitments we consider below.

Theorem 22. *Assume that there exists a (t,ϵ)-secure k-MCRH $\mathcal{H}$ for a polynomial $k = k(n)$ in which every function can be described using $\ell = \ell(n)$ bits. For any parameters $d = d(n)$ and $1 < z \leq n/2d$, there is commitment protocol for strings of length $2^d \cdot n$ with the following properties:*

1. *(t',ϵ')-computationally-binding for $\epsilon' = O\left(2^{\frac{d}{\log(n/(zd))}} \cdot \left(\frac{k^2}{2^{z-d}} + \epsilon\right)\right)$ and $t' = O\left(\frac{\epsilon'^2 \cdot t}{nk2^d \cdot p(n)}\right)$, where $p(\cdot)$ is some fixed polynomial function.*
2. *2^{-n}-statistically-hiding.*
3. *takes $O\left(\frac{d}{\log(n/(zd))}\right)$ rounds.*
4. *the commitment has length $O(d\ell + dn)$.*
5. *supports local opening of size $O\left(d^2 n\right)$.*

There are various ways to instantiate z compared to n and d, offering various trade-offs between security and efficiency. We focus here on the case in which we wish to commit on a polynomially-long string, that is, $d = c \log n$ for a constant $c \in \mathbb{N}$. In the following the big "O" notation hides constants that depend on c. Setting $z = n^{1-\delta}$ for a small constant $\delta > 0$, the parameters of our commitment scheme are:

1. $\epsilon' = O\left(\frac{k^2}{2^{n^{1-\delta}}} + \epsilon\right)$ and $t' = \frac{\epsilon'^2 \cdot t}{\mathsf{poly}(n)}$.
2. 2^{-n}-statistically-hiding.
3. takes $O(1)$ rounds.
4. the commitment has length $O(\ell + n \log n)$.
5. supports local opening of size $O(n \log^2 n)$.

This setting is very efficient in terms of rounds (it is a constant that depends solely on c) but suffers in security loss (the resulting scheme is at most $2^{-n^{1-\delta}}$-secure). In the regime where the MCRH is $(p, 1/p)$-secure for every polynomial p, our resulting scheme is as secure (i.e., $(p, 1/p)$-computationally-binding for every polynomial p).

In case ϵ is very small to begin with (e.g., much smaller than $2^{-n^{1-\delta}}$), we can use set z to be $z = n/(2c \log n)$ for a small constant $\delta > 0$. The resulting commitment scheme satisfies:

1. $\epsilon' = O\left(n^{c-1} \cdot \left(\frac{k^2}{2^{n/\log n}} + \epsilon\right)\right)$ and $t' = \frac{t \cdot \epsilon'^2}{\mathsf{poly}(n)}$.
2. 2^{-n}-statistically-hiding.
3. takes $O(\log n)$ rounds.
4. the commitment has length $O(\ell \log n + n \log n)$.
5. supports local opening of size $O(n \log^2 n)$.

This setting has a logarithmic number of rounds, but the security loss is much smaller than before (only of order $2^{-n/\log n}$).

Roadmap. Our protocol is constructed in two main steps. In the first step (given in Section 6.4.1) we construct a protocol with the above properties (i.e., Theorem 22) except that it is *not* statistically hiding (but is computationally-binding, takes few rounds, has short commitment, and supports local opening). In the second step (given in Section 6.4.2), we show how to *generically* bootstrap our commitment scheme, into one that is also statistically-hiding. This reduction is both efficient and security preserving with respect to all parameters.

6.4.1 A computationally-binding scheme

The main ingredients in our first protocol are an MCRH (Definition 40) and a limited-independent family (Definition 2):

- A (t, ϵ)-secure k-MCRH for a polynomial $k = k(n)$:

  $$\mathcal{H} = \{h \colon \{0,1\}^{2n} \to \{0,1\}^n\}.$$

  We assume that every $h \in \mathcal{H}$ can be described using $\ell = \ell(n)$ bits.

- A family of pairwise-independent functions mapping strings of length $2n$ to strings of length z:

  $$\mathcal{G} = \{g \colon \{0,1\}^{2n} \to \{0,1\}^z\}.$$

  Recall that every $g \in \mathcal{G}$ can be described using $4n$ bits.

Description of the protocol. Our protocol relies on the notion of a Merkle hash tree. This is a method to hash a long string into a short one using a hash function with fixed input length. Let $x \in \{0,1\}^\ell$ be a string. A Merkle hash tree is a binary tree T, associated with a string $x \in \{0,1\}^\ell$ and a hash function $h \colon \{0,1\}^{2n} \to \{0,1\}^n$. Let $x = x_1, \ldots, x_{2^d}$ be the decomposition of x into 2^d blocks, each of length n. Every node v in the tree has a sibling denoted by $N(v)$ (we assume that the sibling of the

root is $\perp$). Every node v in the tree is labeled with a string $\pi_v \in \{0,1\}^n$. The tree has 2^d leaves $v_1, \ldots, v_{2^d}$ and the label of v_i are set to $\pi_{v_i} = x_i$. The labels of the rest of the nodes are computed iteratively from the leaves to the root. Given a node v whose both children u_1, u_2 are labeled with π_{u_1}, π_{v_2}, we set the label of v to be $\pi_v = h(\pi_{u_1}, \pi_{v_2})$. The node root has label y and we call it the root-hash.

Given a Merkle hash tree for a string $x = x_1 \ldots x_{2^d} \in \{0,1\}^{2^d \cdot n}$, let path_i be a set of nodes in the tree including the nodes on the path from x_i to the root of the tree and all their siblings along this path. We further let $P_i = \{\pi_v \mid v \in \mathsf{path}_i\}$ be the set of all the labels of the nodes in the set path_i (the labels of the nodes on the path from x_i to the root of the tree and the labels of their siblings). Each set path_i contains $2d$ nodes and thus the description size of each P_i is $2dn$ bits.

The commitment protocol $(\mathcal{S}, \mathcal{R}, \mathcal{V})$ specifies how to commit to a string of length $2^d n$. Our protocol uses a Merkle hash tree with a function h supplied by the receiver and a root hash replied by the sender. In the next round, the receiver chooses a limited-independence function g with z bits of output (z is a tunable parameter) and sends it to the sender. The sender then computes g on the *hash values* of every pair of sibling nodes in the Merkle hash tree (i.e., $g(\pi_v \circ \pi_{N(v)})$). Then it concatenates all of these values into one long string s'. Then, the protocol continues in a recursive manner on this string s'. The length of s' is roughly $z2^d$ bits (there are 2^d internal nodes in the tree and each requires z bits) which is still too long to send as is, however is smaller than the original string s. This allows us to apply the same ideas recursively with the base case, committing on a string of length n, being the trivial protocol of sending the string as is. Choosing parameters carefully, we balance between the efficiency and security of our resulting commitment.

The commitment protocol for strings of length $2^d n$ for $d \geq 1$ is described in Figure 6.1.

Rounds and communication analysis. We denote by $\mathsf{Size}(2^d n)$, $\mathsf{Rounds}(2^d n)$, and $\mathsf{Decom}(2^d n)$, the *total size* of the commitment, the *number of rounds* of the commit stage, and the *size of a local opening* on a string of length $2^d n$, respectively. The commitment consists of a description of a hash function $h \in \mathcal{H}$ (whose size is denoted by $\ell(n)$), the root-hash value of a Merkle hash-tree (which is of size n), a function $g \in \mathcal{G}$ (which is of size $4n$), and the recursive commitment. The opening for an index $i \in [2^d]$ consists of a block (of size n), the full i-th path (which consists of $2dn$ bits), and the recursive opening.

Recall that the protocol for committing on strings of length $2^d n$ uses (recursively) a protocol for committing on strings of length $2^{d'} n$, where

$$d' = d - (\log n - \log z - \log d) = d - \log(n/(zd)).$$

Moreover, the commitment protocol on strings of length n has communication complexity n, consists of a single round and the opening is n bits. Thus, the total number

of recursive call will be

$$\left\lceil \frac{d}{\log(n/(zd))} \right\rceil .$$

We get that the total number of communication rounds is bounded by

$$\mathsf{Rounds}(2^d n) \leq 3 + \mathsf{Rounds}(2^{d'} n) \leq \cdots \leq \left\lceil \frac{3d}{\log(n/(zd))} \right\rceil + O(1)$$

Actually, since each recursive call consists of three messages (except the base of the recursion), we can join the last round of every iteration with the first round in the next one. Therefore, we can get the improved bound

$$\mathsf{Rounds}(2^d n) \leq \frac{2d}{\log(n/(zd))} + O(1)$$

The size of a commitment is bounded by

$$\begin{aligned} \mathsf{Size}(2^d n) &\leq \ell(n) + 5n + \mathsf{Size}(2^{d'} n) \\ &\leq \left\lceil \frac{d \cdot (\ell(n) + 5n)}{\log(n/(zd))} \right\rceil \leq d \cdot (\ell(n) + 5n). \end{aligned}$$

The size of a local opening is bounded by

$$\begin{aligned} \mathsf{Decom}(2^d n) &\leq n + 2dn + \mathsf{Decom}(2^{d'} n) \\ &\leq \left\lceil \frac{3d^2 n}{\log(n/(zd))} \right\rceil \leq 3d^2 n. \end{aligned}$$

Computational-binding. We show that our protocol is (t_d, ϵ_d)-computationally-binding for strings of length $2^d \cdot n$. We assume that the k-MCRH is (t, ϵ)-secure and that the internal protocol for strings of length $2^{d'} n$ is $(t_{d'}, \epsilon_{d'})$-computationally-binding. We set ϵ_d to satisfy the following recursive relation:

$$\epsilon_d = \frac{4k^2}{2^{z-d}} + 4\epsilon + 4\epsilon_{d'}.$$

Plugging in the number of rounds our recursion takes, we get that

$$\epsilon_d \leq 2^{\frac{6d}{\log n - \log z - \log d}} \cdot \left(\frac{4k^2}{2^{z-d}} + 4\epsilon \right).$$

Let $\mathcal{S}^*$ be a (cheating) adversary that runs in time t_d and breaks the binding of the protocol, namely, with probability ϵ_d, $\mathcal{S}^*$ is able to (locally) open the commitment in two ways without getting caught. That is, after the commitment stage, there is an

index $i \in [2^d]$ for which the adversary $\mathcal{S}^*$ is able to open the corresponding block in two *different* ways with probability ϵ_d:

$$\Pr_{\mathcal{S}^*,\mathcal{R}}\left[\begin{array}{c}(i,\mathsf{decom}_i^0,\mathsf{decom}_i^1,\mathsf{com}) \leftarrow \langle \mathcal{S}^*(1^n),\mathcal{R}\rangle \text{ and} \\ \bot \neq \mathcal{V}(i,\mathsf{decom}_i^0,\mathsf{com}) \neq \mathcal{V}(i,\mathsf{decom}_i^1,\mathsf{com}) \neq \bot\end{array}\right] \geq \epsilon_d, \tag{6.4.1}$$

where the probability is taken over the random coins of both $\mathcal{S}^*$ and $\mathcal{R}$. The randomness of $\mathcal{R}$ consists of uniformly chosen functions $h \leftarrow \mathcal{H}$ and a function $g \leftarrow \mathcal{G}$, so we can rewrite Equation (6.4.1) as

$$\Pr_{\substack{\mathcal{S}^*,h\leftarrow\mathcal{H},\\ g\leftarrow\mathcal{G}}}\left[\begin{array}{c}(i,\mathsf{decom}_i^0,\mathsf{decom}_i^1,\mathsf{com}) \leftarrow \langle \mathcal{S}^*(1^n),(h,g)\rangle \\ \text{and } \bot \neq \mathcal{V}(i,\mathsf{decom}_i^0,\mathsf{com}) \neq \mathcal{V}(i,\mathsf{decom}_i^1,\mathsf{com}) \neq \bot\end{array}\right] \geq \epsilon_d. \tag{6.4.2}$$

We will show how to construct an adversary $\mathcal{A}$ that runs in time at most t and finds a k-wise collision relative to a randomly chosen hash function $h \leftarrow \mathcal{H}$. The procedure $\mathcal{A}$ will run the protocol $\langle \mathcal{S}^*(1^n),(h,g)\rangle$ with the given h and a function g chosen uniformly at random and get (with good probability) two valid and different openings for some index i. Then, $\mathcal{A}$ will *partially* rewind the adversary $\mathcal{S}^*$ to the stage after he received the hash function h and replied with the root hash y (of s), and execute it again but with a fresh function $g \leftarrow \mathcal{G}$. With noticeable probability, this will again result with two valid openings for some (possibly different) index i. Repeating this process enough times, $\mathcal{A}$ will translate a large number of different (yet valid) decommitments into a large number of collisions relative to h. The fact that these collisions are distinct (with good probability) will follow from the fact that the g's are sampled independently in every repetition.

We introduce some useful notation. Recall that $\mathcal{S}^*$ is the sender that is able to locally open its commitment in two different ways. We slightly abuse notation and also think of $\mathcal{S}^*$ as a distribution over senders (this is without loss of generality since we can assume it chooses all of its randomness ahead of time). For an index $i^* \in [R]$, string $y \in \{0,1\}^n$, we denote by $\mathcal{S}^*|_{h,y}$ a distribution over all senders $\mathcal{S}^*$ in which the first message received is h and the reply is the root-hash y. For a string y that is sampled by choosing $h \leftarrow \mathcal{H}$ and running $\mathcal{S}^*$ for one round to obtain y, the adversary $\mathcal{S}^*|_{h,y}$ is uniformly chosen from all possible continuations of the adversary $\mathcal{S}^*$ given these first two messages. Given this notation, we can write the adversary $\mathcal{S}^*$ as a pair of two distributions $(Y_h, \mathcal{S}^*|_{h,Y_h})$, where Y_h is the distribution of the first message $\mathcal{S}^*$ sends in response to h, and $\mathcal{S}^*|_{h,Y_h}$ is the distribution over the rest of the protocol.

In a high-level, our algorithm $\mathcal{A}$ maintains a binary tree of depth d and 2^d leaves, in which each node v is associated with a set of labels S_v. At the beginning, each such set S_v is initialized to be empty. The algorithm $\mathcal{A}$ uses $\mathcal{S}^*$ to get many valid pairs of openings decom_i^0 and decom_i^1 for an index i:

$$\mathsf{decom}_i^0 = (s_i^0, P_i^0, D_{i'}^0) \text{ and } \mathsf{decom}_i^1 = (s_i^1, P_i^1, D_{i'}^1)$$

that are consistent with a commitment:

$$\mathsf{com} = (h, y, g, C).$$

Since both openings are valid and $s_i \neq s'_i$, it must be that (1) s_i^0 (resp., s_i^1) appears in P_i^0 (resp., P_i^1) and (2) P_i^0 and P_i^1 are consistent with the root-hash y. Thus, it must be that $P_i^0 \neq P_i^1$ and there is a node v on the path path_i that is the first (going from the root to the leaves) non-trivial collision between P_i^0 and P_i^1. Now, by the induction hypothesis, the probability that $\mathcal{S}^*$ cheats in the internal decommitment is small, and since g is sampled uniformly at every iteration, we show that it must be a *new* collision that did not appear before. We will identify the location of the collision at a node v and add this collision to the set S_v. See Figure 6.2 for a precise description of $\mathcal{A}$.

The success probability of $\mathcal{A}$. We say that $h \in \mathcal{H}$ and $y \in \mathsf{supp}(Y_h)$ are "good" if it holds that

$$\Pr_{\mathcal{S}^*|_{h,y}, g \leftarrow \mathcal{G}} \left[\begin{array}{l} (i, \mathsf{decom}_i^0, \mathsf{decom}_i^1, \mathsf{com}) \leftarrow \langle \mathcal{S}^*|_{h,y}(1^n), g \rangle \text{ and} \\ \bot \neq \mathcal{V}(i, \mathsf{decom}_i, \mathsf{com}) \neq \mathcal{V}(i, \mathsf{decom}'_i, \mathsf{com}) \neq \bot \end{array} \right] \geq \frac{\epsilon_d}{2}. \tag{6.4.3}$$

By an averaging argument, Equation (6.4.2) implies that there are many such "good" (h, y) pairs, namely:

$$\Pr_{h \leftarrow \mathcal{H}, y \leftarrow Y_h} \left[\Pr_{\mathcal{S}^*|_{h,y}, g \leftarrow \mathcal{G}} \left[\begin{array}{l} (i, \mathsf{decom}_i^0, \mathsf{decom}_i^1, \mathsf{com}) \leftarrow \langle \mathcal{S}^*|_{h,y}(1^n), g \rangle \text{ and} \\ \bot \neq \mathcal{V}(i, \mathsf{decom}_i, \mathsf{com}) \neq \mathcal{V}(i, \mathsf{decom}'_i, \mathsf{com}) \neq \bot \end{array} \right] \geq \frac{\epsilon_d}{2} \right] \geq \frac{\epsilon_d}{2}. \tag{6.4.4}$$

For the rest of the proof we condition on the event that $\mathcal{A}$ sampled an h and got back a y such that (h, y) are good.

We say that iteration j of $\mathcal{A}$ is *successful* $\mathcal{A}$ passes step 3d and reaches step 3e: if $\mathcal{S}^*_{h,y}$ outputs (at Item 3b) a valid pair of openings $(i, \mathsf{decom}_i^0, \mathsf{decom}_i^1, \mathsf{com})$ that pass the tests. We denote this event by SUC_i. Recall that in every successful iteration, $\mathcal{A}$ adds X and Y to the set S_v for some node v. Notice that S_v is a set so it does not contain duplicate elements. Since we condition on (h, y) being good and cheating in the internal commitment is possible with probability at most $\epsilon_{d'}$, we have that

$$\Pr_{\mathcal{A}}[\mathsf{SUC}_j] \geq \frac{\epsilon_d}{2} - \epsilon_{d'}. \tag{6.4.5}$$

We say that iteration j is *effective* if for the chosen $g \in \mathcal{G}$ (at Item 3a) it holds that $\forall v, \forall X \neq Y \in S_v \colon g(X) \neq g(Y)$. We denote this event by EFF_i. Since g is pairwise independent, there are 2^d internal nodes in the tree, and at most k^2 pairs in each node (since $|S_v| \leq k$), it holds that

$$\Pr_{\mathcal{A}}[\mathsf{EFF}_i] = 1 - \Pr_{\mathcal{A}}[\exists v, \exists X \neq Y \in S_v \colon g(X) = g(Y)] \geq 1 - \frac{2^d \cdot k^2}{2^z}. \tag{6.4.6}$$

Combining Equations (6.4.5) and (6.4.6) together with the inequality that $\epsilon_d \geq 4k^2/2^{z-d} + 4\epsilon_{d'}$, we get that an iteration i is both successful and effective with probability at least

$$\Pr_{\mathcal{A}}[\mathsf{SUC}_i \text{ and } \mathsf{EFF}_i] \geq 1 - \Pr_{\mathcal{A}}[\neg\mathsf{SUC}_i] - \Pr_{\mathcal{A}}[\neg\mathsf{EFF}_i] \geq \frac{\epsilon_d}{2} - \epsilon_{d'} - \frac{k^2}{2^{z-d}} \geq \frac{\epsilon_d}{4}. \tag{6.4.7}$$

We observe that every iteration which is both successful and effective, increases the size of the set S_v for some node v. Indeed, let X^* and Y^* be the pair added to some set S_v in this iteration. Since the iteration is successful it holds that $X^* \neq Y^*$ and furthermore these values are consistent with the commitment, namely, $g(X^*) = g(Y^*)$ for the g that was chosen in this iteration. However, since the iteration is effective, there are no other two values $X, Y \in S_v$ that map to the same value relative to g. This means that either X^* or Y^* were not in the set S_v before.

To complete the proof we have to show that (with high probability) there are at least $k \cdot 2^d$ iterations that are both successful and effective. If this happens, there is a node v such that $|S_v| \geq k$ and this means that we found a k-wise collision. Denote by X_i the event that the i-th iteration is both successful and effective. Denote by $Y_n = \sum_{i=1}^{n}(X_i - \mathbf{E}[X_i \mid Y_0, \ldots, Y_{i-1}])$ and $Y_0 = 0$. We observe that the sequence $Y_0, Y_1, \ldots$ is a martingale with a bounded difference. By linearity of expectation and the law of total expectation:

$$\mathbf{E}[Y_n \mid Y_0, \ldots, Y_{n-1}] = \mathbf{E}[Y_{n-1} + X_n - \mathbf{E}[X_n \mid Y_0, \ldots, Y_{n-1}] \mid Y_0, \ldots, Y_{n-1}] = Y_{n-1}$$

and $|Y_i - Y_{i-1}| \leq 1$ for every $i \in [n]$. By Equation (6.4.7):

$$\sum_{i=1}^{T} \mathbf{E}[X_i \mid Y_0, \ldots, Y_{i-1}] \geq T \cdot \frac{\epsilon_d}{4}.$$

Since $T = 50nk2^d/\epsilon_d^2$, then both $T\epsilon_d^2/50 \geq n$ and $T\epsilon_d/n \geq k2^d$. Thus, by the Azuma bound (see e.g., [AS08, Theorem 7.2.1]):

$$\begin{aligned}
\Pr_{\mathcal{A}}\left[\sum_{i=1}^{T} X_i < k2^d\right] &= \Pr_{\mathcal{A}}\left[\sum_{i=1}^{T} X_i - \sum_{i=1}^{T} \mathbf{E}[X_i \mid Y_0, \ldots, Y_{i-1}] < k2^d - \sum_{i=1}^{T} \mathbf{E}[X_i \mid Y_0, \ldots, Y_{i-1}]\right] \\
&\leq \Pr_{\mathcal{A}}[Y_T < -(T\epsilon_d/4 - T\epsilon_d/n)] \\
&\leq \Pr_{\mathcal{A}}[Y_T < -T\epsilon_d/5] = \Pr_{\mathcal{A}}\left[Y_T < -(\sqrt{T}\epsilon_d/5) \cdot \sqrt{T}\right] \\
&\leq 2^{-T\epsilon_d^2/50} \leq 2^{-n}.
\end{aligned}$$

Summarizing the proof, since $\epsilon_d > 4\epsilon$, we get that $\mathcal{A}$ finds a collision in $h \leftarrow \mathcal{H}$

with probability

$$\begin{aligned}
&\Pr_{h,\mathcal{A}}\left[\mathcal{A}(1^n,h) \text{ outputs a } k\text{-wise collision}\right] = \Pr_{h,\mathcal{A}}\left[\exists v\colon |S_v| \geq k\right] \\
&\quad \geq \Pr_{h,\mathcal{A}}\Big[(h,y) \text{ are good } \wedge\ k\cdot 2^d \textit{ successful} \text{ and } \textit{effective} \text{ iterations}\Big] \\
&\quad \geq \frac{\epsilon_d}{2}\cdot\left(1-\frac{1}{2^n}\right) \geq \frac{\epsilon_d}{4} > \epsilon,
\end{aligned}$$

The running time of $\mathcal{A}$ is bounded by T executions of $\mathcal{S}^*$ and a fixed polynomial factor $p(n)$ (that bounds the time it takes to sample $g \in \mathcal{G}$ and the additional simple operations $\mathcal{A}$ makes including the evaluation of h inside $\mathcal{V}$). In total, since $\mathcal{S}^*$ runs in time t', then $\mathcal{A}$'s running time is at most

$$T\cdot t'\cdot p(n) = \frac{50nk2^d\cdot p(n)}{\epsilon_d^2}\cdot t' \leq t.$$

Remark 4 (Optimization I: recycling h)**.** *In the recursive step of our protocol, we can use the same hash function h as in the first step of the recursion. This saves sending its description (which is of size $\ell(n)$ at every iteration and the resulting size of a commitment is $(O(\ell(n)+d^2n)$.*

Remark 5 (Optimization II: almost-universal hashing)**.** *In our protocol we used a pair-wise independent hash function whose description size is proportional to their input size. This costs us in communication. We could save communication by using almost-universal hash functions whose description size is proportional to their output size.*

6.4.2 Getting statistical-hiding generically

We show how to transform any short commitment scheme Π that is computationally binding (but perhaps not hiding) to a new scheme Π' that is short, computationally binding and *statistically hiding*. Moreover, if Π admits a local-opening, then Π' admits a local-opening as well. Our transformation is information theoretic, adds no additional assumptions and preserves the security, the number of rounds and communication complexity of the original scheme (up to a small constant factor).

High-level idea. The underlying idea of our transformation is to leverage the fact that the commitment protocol Π is short. Specifically, when committing to a long string $s \in \{0,1\}^{n^c}$ for some large $c \in \mathbb{N}$, the communication is very short: $\Lambda(n)$ bits for a fixed polynomial function.[13] Thus, when we commit to s, a large portion of s is not revealed to the receiver by the protocol so this part of s is statistically hidden. The question is how to make sure that all of s is hidden.

[13]Our protocol from Section 6.4.1 has an additional linear dependence on c, but we will ignore this in this section to simplify notation.

Our solution takes advantage of the fact that some fraction of s remains hidden. We commit on a random string r that is independent of s and slightly longer. Then, we extract from r the remaining randomness r' *given the communication of the protocol* using a strong extractor (see Section 2.1.2). Finally, we commit on the string $s \oplus r'$. We need to show that this scheme is computationally-binding and statistically-hiding. The former follows from the computational-binding of the original scheme. The latter follows from the fact that r' is completely hidden to the receiver and it masks the value of s.

The commitment protocol $(\mathcal{S}, \mathcal{R}, \mathcal{V})$ that we describe uses the underlying commitment protocol $(\widehat{\mathcal{S}}, \widehat{\mathcal{R}}, \widehat{\mathcal{V}})$ (which is *not* statistically-hiding) and a family of pairwise-independent functions

$$\mathcal{G} = \{g \colon \{0,1\}^{8\Lambda(n)} \to \{0,1\}\}.$$

Recall that each function $g \in \mathcal{G}$ can be represented with $8\Lambda(n)$ bits.

In Section 6.4.1, we presented a commitment scheme that has also linear dependence on c. Specifically, the length of a commitment there is at most

$$(c-1)\log n \cdot (\ell(n) + 5n),$$

where $\ell(n)$ is the size of the descriptions of a hash function. For simplicity of presentation in this section we suppress this linear term in c and assume that $\Lambda(n)$ is independent of c. The precise description of the protocol is given in Figure 6.3.

Computational-binding. Assume that after the commitment stage, there is an index $i \in [N]$ for which a malicious sender $\mathcal{S}^*$ is able to open the corresponding bit in two *different* ways with probability $\epsilon = \epsilon(n)$:

$$\Pr_{\mathcal{S}^*,\mathcal{R}}\left[\begin{array}{c}(i, \mathsf{decom}_i, \mathsf{decom}'_i, \mathsf{com}) \leftarrow \langle \mathcal{S}^*(1^n), \mathcal{R}\rangle \text{ and} \\ \bot \neq \mathcal{V}(i, \mathsf{decom}_i, \mathsf{com}) \neq \mathcal{V}(i, \mathsf{decom}'_i, \mathsf{com}) \neq \bot\end{array}\right] \geq \epsilon,$$

where the probability is taken over the random coins of both $\mathcal{S}^*$ and $\mathcal{R}$. We sketch how to use this malicious sender to design a malicious sender $\widehat{\mathcal{S}}^*$ that breaks the computational binding of the basic protocol with $\widehat{\mathcal{R}}$.

Recall that $\mathsf{decom}_i = (\widehat{r_i^*}, \widehat{g_i^*}, \widehat{u_i^*})$ and $\mathsf{decom}'_i = (\widehat{r_i^*}', \widehat{g_i^*}', \widehat{u_i^*}')$ and the fact that $\mathcal{V}$ succeeds and outputs non-$\bot$ and different values, implies that

$$\widehat{g_i^*}(\widehat{r_i^*}) \oplus \widehat{u_i^*} \neq \widehat{g_i^*}'(\widehat{r_i^*}') \oplus \widehat{u_i^*}'.$$

Thus, either $\widehat{g_i^*}(\widehat{r_i^*}) \neq \widehat{g_i^*}'(\widehat{r_i^*}')$ or $\widehat{u_i^*} \neq \widehat{u_i^*}'$. This means that one of the following three must occur: (1) $\widehat{r_i^*} \neq \widehat{r_i^*}'$, (2) $\widehat{g_i^*} \neq \widehat{g_i^*}'$, or (3) $\widehat{u_i^*} \neq \widehat{u_i^*}'$. Let us assume that the first event occurs and the rest of the cases are handled analogously. In this case, our malicious sender $\widehat{\mathcal{S}}^*$ will simply simulate the first step of $\mathcal{S}^*$ (namely, the part in

which it commits on a random string r^1). The rest of the protocol will be executed with a simulation of $\mathcal{R}$, without real interaction. As a result, with probability ϵ, the output of $\langle \widehat{\mathcal{S}}^*(1^{n+N}), \widehat{\mathcal{R}} \rangle$ results with a single commitment and two openings which $\widehat{\mathcal{V}}$ will accept as valid. This is a contradiction to the computational binding of the basic protocol which implies that ϵ cannot be noticeable.

In conclusion, if the original protocol is (t, ϵ)-secure, our new protocol is $(t, \epsilon/3)$-secure.

Statistical-hiding. We will show that for every (deterministic) adversary $\mathcal{R}^*$ and every distinct $s_0, s_1 \in \{0,1\}^N$, the ensembles $\{\mathsf{view}_{\langle \mathcal{S}(s_0), \mathcal{R}^* \rangle}(n)\}$ and $\{\mathsf{view}_{\langle \mathcal{S}(s_1), \mathcal{R}^* \rangle}(n)\}$ have statistical distance at most $\rho(n) = 2^{-n}$ for all sufficiently large $n \in \mathbb{N}$.

Recall that the view of $\mathcal{R}^*$ consists of three parts where the first two parts are independent of the message and the last part depends on it. For a message s, denote the (distribution of) the view of $\mathcal{R}^*$ by a triple of random variables $(R^{\mathsf{view}}, G^{\mathsf{view}}, E_s^{\mathsf{view}})$. We further denote by (R, G, E_s) the distribution over (r^1, r^2, r^3). Using this notation, we have that

1. $R = R_1, \ldots, R_N$, where each R_i is the uniform distribution over $8\Lambda(n)$ strings.
2. $G = G_1, \ldots, G_N$, where each G_i is the uniform distribution over pairwise independent functions samples from $\mathcal{G}$. Each such G_i is supported on strings of length $8\Lambda(n)$.
3. $E_s = G_1(R_1) \oplus s, \ldots, G_N(R_N) \oplus s$.

Since each R_i is essentially the uniform distribution over strings of length $8\Lambda(n)$, it holds that $\mathsf{H}_\infty(R) = 8\Lambda(n)$. Moreover, since the whole communication of the protocol is bounded by $3\Lambda(n)$, by Proposition 2, we have that for every $i \in [N]$:

$$\begin{aligned} \mathsf{H}_\infty(R_i \mid (R^{\mathsf{view}}, G^{\mathsf{view}}, E_s^{\mathsf{view}})) &= 8\Lambda(n) - |(R^{\mathsf{view}}, G^{\mathsf{view}}, E_s^{\mathsf{view}})| \\ &\geq 8\Lambda(n) - 3\Lambda(n) = 5\Lambda(n). \end{aligned}$$

Thus, by the leftover hash lemma (see Proposition 1), for every $i \in [N]$:

$$\Delta((R^{\mathsf{view}}, G_i, G_i(R_i)), (R^{\mathsf{view}}, G_i, U)) \leq 2^{-2n},$$

where U is the uniform distribution on a single bit. Moreover, for every $s \in \{0,1\}^N$:

$$\Delta((R^{\mathsf{view}}, G_i, U), (R^{\mathsf{view}}, G_i, U \oplus s)) = 0.$$

Therefore, by a triangle inequality, for every two different messages $s_0, s_1 \in \{0,1\}^N$,

$$\Delta((R^{\mathsf{view}}, G_i, G_i(R_i) \oplus s_0), (R^{\mathsf{view}}, G_i, G_i(R_i) \oplus s_1)) \leq 2 \cdot 2^{-2n}.$$

Taking a union bound over the different i's, we get that

$$\begin{aligned}
&\Delta(\{\mathsf{view}_{\langle \mathcal{S}(s_0),\mathcal{R}^*\rangle}(n)\},\{\mathsf{view}_{\langle \mathcal{S}(s_1),\mathcal{R}^*\rangle}(n)\}) \\
&= \Delta((R^{\mathsf{view}},G^{\mathsf{view}},E_s^{\mathsf{view}}),(R^{\mathsf{view}},G^{\mathsf{view}},E_s^{\mathsf{view}})) \\
&\leq N\cdot\Delta((R^{\mathsf{view}},G_i,G_i(R_i)\oplus s_0),(R^{\mathsf{view}},G_i,G_i(R_i)\oplus s_1)) \\
&\leq 2N\cdot 2^{-2n}\leq 2^{-n}.
\end{aligned}$$

Efficiency. The (1) size of a commitment, (2) of a decommitment, or (3) or the number of rounds in $(\mathcal{S},\mathcal{R},\mathcal{V})$ are all the same as that of $(\widehat{\mathcal{S}},\widehat{\mathcal{R}},\widehat{\mathcal{V}})$ up to a multiplicative factor of 3, respectively.

6.5 Four-Round Short Commitments from Multi-CRH

We show how to construct a *4-round* short commitment protocol based on a family of k-MCRH functions. Compared to the protocol from Section 6.4, this protocol has *no* local opening and is secure only for *constant* values of k. However, it consists only of 4 rounds. Furthermore, using techniques similar to Section 6.4.2 the protocol can be made also statistically-hiding which suffices for some applications such as statistical zero-knowledge arguments [BCC88].

We discuss methods that allow us to prove security even for polynomial values of k towards the end of the section.

Theorem 23. *Assume that there exists a (t,ϵ)-secure k-MCRH $\mathcal{H}$ for a constant k in which every function can be described using $\ell=\ell(n)$ bits. For any $c\in\mathbb{N}$, there is a commitment protocol for strings of length n^c with the following properties:*

1. *(t',ϵ')-computationally-binding for $\epsilon'=4\epsilon+\frac{O(k^c\log n)}{2^n}$ and $t'=\frac{t\cdot\epsilon'^2}{O(k^c\log n)\cdot p(n)}$, where $p(\cdot)$ is some fixed polynomial function.*
2. *takes 4 rounds (i.e., 4 messages).*
3. *the commitment has length $\ell+O(n)$.*

Proof. We describe a commitment protocol for a string $s=s_1\ldots s_{2^d}$ of length $2^d\cdot n$ for $d=(c-1)\cdot\log n$. We show that our protocol is computationally-binding and getting statistical-hiding can be done using the generic transformation from Section 6.4.2. Our protocol uses a Merkle hash-tree as described in Section 6.4. The main observation made in this construction is that before using the Merkle hash-tree we can use a special type of encodings, called list-recoverable codes (see Definition 4) to get meaningful security. Let $C\colon\{0,1\}^{2^dn}\to(\{0,1\}^{2n})^{2^{d'}}$ be an (ℓ,L)-list-recoverable code for $\ell=k^d$, $L=O(k^d)$, and $d'=O(\log n)$ with some large enough hidden constants. Such a code exists by Theorem 1 (with efficient encoding and list-recovery). Let $\mathcal{G}$ be

a family of (2^{-n})-almost pairwise-independent functions mapping strings of length $2^d n$ to strings of length n (see Definition 3):

$$\mathcal{G} = \{g \colon \{0,1\}^{2^d n} \to \{0,1\}^n\}.$$

Recall that every $g \in \mathcal{G}$ can be described using at most $2n + \log(2^d n) + \log(2^n) \leq 4n$ bits.

The commitment protocol for a string $s = s_1 \dots s_{2^d}$ of length $2^d \cdot n$ for $d = (c-1) \cdot \log n$ works as follows. The receiver first sends a description of an $h \leftarrow \mathcal{H}$ which is a k-MCRH to the sender. The sender then computes the encoding $C(s)$ of s and computes the Merkle hash tree of $s' = C(s)$ (and not of s). The sender sends the root of the hash tree y to the receiver. The receiver replies with a hash function $g \leftarrow \mathcal{G}$ and finally the sender replies with $u = g(s)$. The opening is done is the natural way by letting the sender reveal s to the receiver who then simulates the computation and makes sure that the messages y and u are consistent. See Figure 6.4 for the precise description.

By the description of the protocol, one can see that the protocol consists of 4-rounds and has communication complexity of $\ell + n + 4n + n = \ell + 6n$ bits. In addition, for the honest sender the verification succeeds with probability 1. We proceed with the proof of security.

Assume that there is a malicious sender $\mathcal{S}^*$ that runs in time $t' = t \cdot \epsilon'^2/(L \cdot p(n))$ (for some polynomial $p(\cdot)$) and is able open a commitment in two *different* ways with probability $\epsilon' = \epsilon'(n) \geq 4\epsilon + 10L/2^n$:

$$\Pr_{\mathcal{S}^*,\mathcal{R}}\left[\begin{array}{c}(x, x', \mathsf{com}) \leftarrow \langle \mathcal{S}^*(1^n), \mathcal{R}\rangle \text{ and} \\ \bot \neq \mathcal{V}(x, \mathsf{com}) \neq \mathcal{V}(x', \mathsf{com}) \neq \bot\end{array}\right] \geq \epsilon',$$

where the probability is taken over the random coins of both $\mathcal{S}^*$ and $\mathcal{R}$. We sketch how to use this malicious sender to come up with more than k values that collide relative to the MCRH h. The randomness of $\mathcal{R}$ consists of uniformly chosen functions $h \leftarrow \mathcal{H}$ and a function $g \leftarrow \mathcal{G}$, so we can rewrite the above equation as

$$\Pr_{\substack{\mathcal{S}^*, h \leftarrow \mathcal{H}, \\ g \leftarrow \mathcal{G}}}\left[\begin{array}{c}(x, x', \mathsf{com}) \leftarrow \langle \mathcal{S}^*(1^n), (h, g)\rangle \\ \text{and } \bot \neq \mathcal{V}(x, \mathsf{com}) \neq \mathcal{V}(x', \mathsf{com}) \neq \bot\end{array}\right] \geq \epsilon'. \tag{6.5.1}$$

The high-level idea is that an adversary that succeeds in the task above must succeed in the task for a random h and a bunch of *different* g's. Simulating $\mathcal{S}^*$ on sufficiently many g's (but the same h) we obtain a list of different x's that map to the same value under $h(\cdot)$. Then, one of two must happen: (1) either the adversary is able to create many different consistent codewords using few options per coordinate, or (2) there is one entry with many different values. Using the properties of the list-recoverable code we rule-out option (1). We are then left with option (2) which means that there must be a location where the adversary uses many openings which translates to a k-wise collision relative to h for some node on the path between the location and the root.

We formalize this intuition next by defining an algorithm $\mathcal{A}$ that gets as input 1^n and $h \leftarrow \mathcal{H}$ and finds a k-wise collision. We say that $h \in \mathcal{H}$ is Good if

$$\Pr_{\mathcal{S}^*, g \leftarrow \mathcal{G}} \left[\begin{array}{c} (x, x', \mathsf{com}) \leftarrow \langle \mathcal{S}^*(1^n), (h, g) \rangle \\ \text{and } \bot \neq \mathcal{V}(x, \mathsf{com}) \neq \mathcal{V}(x', \mathsf{com}) \neq \bot \end{array} \right] \geq \frac{\epsilon'}{2}.$$

Thus, by averaging,

$$\Pr_{h \leftarrow \mathcal{H}}[h \text{ is Good}] \geq \frac{\epsilon'}{2}.$$

Our procedure $\mathcal{A}$ simulates the receiver in the protocol and then iteratively partially rewinds the sender with different choices of g.
The algorithm $\mathcal{A}(1^n, h)$:

1. Run $\mathcal{S}^*$'s first part with the hash function h to obtain a root hash y.
2. Initialize a set $\mathcal{X} = \emptyset$.
3. Do the following $T = 100n \cdot L/\epsilon'^2$ times:
   (a) Sample a fresh $g \leftarrow \mathcal{G}$.
   (b) Run $\mathcal{S}^*$'s second part (by rewinding) given the new hash function g to obtain a value u.
   (c) Open the commitment to get a values x and x' and add them to the set X.
   (d) If the set $\mathcal{X}$ contains more than L values, quit the loop.
4. Find a coordinate for which there are ℓ different values in the codewords of messages in $\mathcal{X}$. In this coordinate, traverse the path to the root of the hash tree and find a location that has more than k values that has to the same value. Output these values.

The running time of $\mathcal{A}$ is polynomial in n, $L = O(k^d)$, and $1/\epsilon'$. Altogether, since k is constant, $d = O(\log n)$, and $1/\epsilon'$ is a fixed polynomials in n, the running time of $\mathcal{A}$ is a fixed polynomial in n. We show that the algorithm $\mathcal{A}$ succeeds in finding a k-wise collision with noticeable probability. In the following claim we show that with noticeable probability the execution of the loop in the description of the adversary $\mathcal{A}$ terminates with a large set $\mathcal{X}$ (which will then translate into a large enough collision). The proof of this claim is similar to the proof in Theorem 22.

Claim 10. $\Pr_{\mathcal{A}}[|\mathcal{X}| > L \mid h \text{ is Good}] \geq 1 - 2^{-n}$.

Proof. Since h is Good, with probability $\epsilon'/2$, in any iteration we obtain x and x' for which verification succeeds yet outputs two different values (x and x' in our case).

We say that iteration i is *successful*, denoted by SUC_i if this iteration satisfies the above (it outputs two different x and x' that pass verification). Thus, it holds that

$$\Pr_{\mathcal{A}}[\mathsf{SUC}_i] \geq \frac{\epsilon'}{2}. \tag{6.5.2}$$

We further say that iteration j is *effective*, denoted by EFF_i, if for the chosen $g \in \mathcal{G}$, the outputted x and x', and every $x'' \in \mathcal{X}$, it holds that $g(x) \neq g(x'')$. Since g is (2^{-n})-almost pairwise independent and the set X is of size at most L, it holds that

$$\Pr_{\mathcal{A}}[\mathsf{EFF}_i] \geq 1 - \Pr_{\mathcal{A}}\left[\exists x'' \in \mathcal{X} \colon g(x) = g(x'')\right] \geq 1 - \frac{L}{2^n} - \frac{1}{2^n} \geq 1 - \frac{L+1}{2^n}. \tag{6.5.3}$$

Combining the inequalities in Equation (6.5.2) and in Eq. (6.5.3) with the assumption that $\epsilon' \geq 10L/2^n$, we get that

$$\Pr_{\mathcal{A}}[\mathsf{SUC}_i \text{ and } \mathsf{EFF}_i] \geq 1 - \Pr_{\mathcal{A}}[\neg\mathsf{SUC}_i] - \Pr_{\mathcal{A}}[\neg\mathsf{EFF}_i] \geq \frac{\epsilon'}{2} - \frac{L+1}{2^n} \geq \frac{\epsilon'}{4}. \tag{6.5.4}$$

Observe that every iteration which is both successful and effective increases the size of the set $\mathcal{X}$ by at least one. To see this, fix a pair x and x' that the malicious sender $\mathcal{S}^*$ outputs in a fixed iteration (which is both successful and effective). Since the iteration is successful it holds that $x \neq x'$ and furthermore these values are consistent with the g chosen in that iteration, namely, $g(x) = g(x')$. Since the iteration is effective, there is no other value $x'' \in \mathcal{X}$ that maps to the same value relative to g. This means that either x or x' were not in the set $\mathcal{X}$ before the current iteration.

To complete the proof we need to show that there are many successful and effective iterations. Denote by X_i the event that the i-th iteration is both successful and effective. Define $Y_n = \sum_{i=1}^{n}(X_i - \mathbf{E}[X_i \mid Y_0, \ldots, Y_{i-1}])$ and $Y_0 = 0$. The sequence $Y_0, Y_1, \ldots$ has bounded difference ($|Y_i - Y_{i-1}| \leq 1$ for all i) and is a martingale by the law of total expectation:

$$\mathbf{E}[Y_n \mid Y_0, \ldots, Y_{n-1}] = \mathbf{E}[Y_{n-1} + X_n - \mathbf{E}[X_n \mid Y_0, \ldots, Y_{i-1}] \mid Y_0, \ldots, Y_{n-1}] = Y_{n-1}.$$

Furthermore, by Equation (6.5.4)

$$\sum_{i=1}^{T} \mathbf{E}[X_i \mid Y_0, \ldots, Y_{i-1}] \geq T \cdot \frac{\epsilon'}{4}.$$

Thus, by the Azuma bound (see e.g., [AS08, Theorem 7.2.1]):

$$\begin{aligned}
\Pr_{\mathcal{A}}\left[|\mathcal{X}| < nk^d\right] &= \Pr_{\mathcal{A}}\left[\sum_{i=1}^{T} X_i < 4k^d\right] \\
&= \Pr_{\mathcal{A}}\left[\sum_{i=1}^{T} X_i - \sum_{i=1}^{T} \mathbf{E}[X_i \mid Y_0, \ldots, Y_{i-1}] < 4k^d - \sum_{i=1}^{T} \mathbf{E}[X_i \mid Y_0, \ldots, Y_{i-1}]\right] \\
&\leq \Pr_{\mathcal{A}}\left[Y_T < -(T\epsilon'/4 - T\epsilon'/n)\right] \\
&\leq \Pr_{\mathcal{A}}\left[Y_T < -T\epsilon'/5\right] = \Pr_{\mathcal{A}}\left[Y_T < -(\sqrt{T}\epsilon'/5)\cdot\sqrt{T}\right] \\
&\leq 2^{-T\epsilon'^2/50} \leq 2^{-n}.
\end{aligned}$$

This completes the proof of the claim. □

We now condition on an execution of $\mathcal{A}$ terminating with $|\mathcal{X}| > L$. This event implies that the malicious sender $\mathcal{S}^*$ came up with at least $|\mathcal{X}|$ different messages x all of which correspond to codewords of C that are consistent with the root hash. Each such x implies a different codeword $C(x)$ for our (ℓ, L)-list-recoverable code. By the definition of list-recoverability and since we have $|\mathcal{X}| > L$ different codewords, there must be a coordinate $i \in [2^{d'}]$ in the code that has more than ℓ different values. Since $\ell = k^d$ and the depth of the Merkle hash tree is d, by the pigeonhole principle, there is a coordinate (in which symbols are composed of $2n$ bits) in which there must be a location on the path from this coordinate to the root hash that has at least k different values that hash to the same value and thus give us a k-wise collision relative to h.

To conclude the proof, we observe that the algorithm $\mathcal{A}$ runs in time $\mathsf{poly}(n) \cdot t' \cdot L/\epsilon'^2 \leq t$ and finds a k-wise collision in $h \leftarrow \mathcal{H}$ with probability

$$\begin{aligned}
\Pr_{h,\mathcal{A}}\left[\mathcal{A}(1^n, h) \text{ outputs a } k\text{-wise collision}\right] &= \Pr_{h,\mathcal{A}}\left[|\mathcal{X}| > L\right] \\
&\geq \Pr_{h,\mathcal{A}}\left[|\mathcal{X}| > L \mid h \text{ is Good}\right] \cdot \Pr_{h \leftarrow \mathcal{H}}\left[h \text{ is Good}\right] \\
&\geq \left(1 - \frac{1}{2^n}\right) \cdot \frac{\epsilon'}{2} > \epsilon,
\end{aligned}$$

which is a contradiction to the security of the MCRH $\mathcal{H}$.

□

Supporting arbitrary larger k. The reason why we could prove security only for constant values of k stems from the fact that our adversary $\mathcal{A}$ for the MCRH runs in time proportional to k^d. The source of this term in the running time is that our Merkle hash tree in the construction is of depth $d = O(\log n)$ and when counting the number of possible openings per coordinate in a leaf, we get k^d possibilities. Our adversary $\mathcal{A}$ basically "collects" this number of different openings for some coordinate and

thereby finds a k-wise collision. If k is super-constant the running time of $\mathcal{A}$ becomes super-polynomial.

There are two paths we can take to bypass this. One is to assume super-polynomial security of the MCRH and allow our adversary to run in super-polynomial time. The second assumption is to assume that we start with a stronger MCRH that compresses by a polynomial factor (i.e., from $n^{1+\Omega(1)}$ to n) rather than by a factor of 2. This will cause the Merkle hash tree to be of constant depth. Under either of the assumptions, we can support polynomial values of k (in n). However, both assumptions are rather strong on the underlying MCRH and we thus prefer to avoid them; see Section 6.3 for a discussion.

Domain extension of MCRH functions. The construction we gave in the proof of Theorem 23 can be viewed as a domain extension method for MCRH functions. Specifically, given a k-MCRH f that maps $2n$ bits into n bits, we constructed a function g that maps $m = m(n)$ bits into n bits for any polynomial $m(\cdot)$ such that g is a $((k-1)^{\log m}+1)$-MCRH.

Other uses of list-recoverable codes for domain-extension. List-recoverable codes have been proven useful in various applications in cryptography. The work of Maurer and Tessaro [MT07] considers the problem of a extending the domain of a public random function. They show how to use a length-preserving random function f on n input bits, to get a variable-input-length function g that maps arbitrary polynomial-size inputs into arbitrary polynomial-size outputs such that g is indistinguishable from a random function for adversaries making up to $2^{(1-\epsilon)n}$ queries to g. One of their main components in the construction is a list-recoverable code (there referred to as an *input-restricting function family*).

Building on ideas and techniques of [MT07], Dodis and Steinberger [DS11] showed that list-recoverable codes are also useful for security preserving domain extension of message-authentication codes (with "beyond-birthday" security).

More recently, Haitner et al. [HIOS15] studied the possibility of a *fully parallel* (i.e., non-adaptive) domain-extension scheme that realizes a collision-resistant hash function. Starting with a *random* function f that maps n bits into n bits, they construct a function g that maps $m(n)$ bits into n bits for any polynomial $m(\cdot)$, makes only parallel calls to f, and requiring an attacker to make at least $2^{n/2}$ queries to g to find a collision with high probability. Their construction uses list-recoverable codes and they show that such codes are actually necessary for this task.

6.6 Separating Multi-CRH from CRH

In this section we rule out fully black-box constructions (see Definition 14) of CRH functions from k-MCRH functions for $k > 2$. Our proof technique is inspired by the works of Asharov and Segev [AS16] and Haitner et al. [HHRS15], that are based in

turn on ideas originating in the works of Simon [Sim98] Gennaro et al. [GGKT05]) and Wee [Wee07].

We present an oracle Γ relative to which there exists a 3-multi-collision-resistant hash function, but any collision-resistant hash function can be easily broken. The theorem below is stated and proved only for the case of standard CRH and 3-MCRH. The ideas in this proof naturally extend to a separation of k-MCRH from $(k+1)$-MCRH for all fixed values of k.

Theorem 24. *There is no fully black-box construction of a collision-resistant hash function family from a 3-multi-collision-resistant hash function family mapping $2n$ bits to n bits.*

We describe the oracle Γ next. It is in fact a distribution over pairs of oracles $(f, \mathsf{ColFinder}^f)$ in which f is a uniformly chosen function and ColFinder is an oracle that finds pair-wise collisions.

The oracle Γ. The oracle Γ consists of a pair of oracles $(f, \mathsf{ColFinder}^f)$:

- **The function $f = \{f_n\}_{n\in\mathbb{N}}$:** For every n, the function f_n is a uniformly chosen function from $2n$ bits to n bits.
- **The function $\mathsf{ColFinder}^f$:** This function consists of an infinite collection of permutations, where for every $n \in \mathbb{N}$ and every circuit $C^f\colon \{0,1\}^{2n} \to \{0,1\}^n$, there are two uniformly and independently chosen permutations π^1_C, π^2_C over $\{0,1\}^{2n}$. When $\mathsf{ColFinder}^f$ is given C^f as input, it sets $x_1 = \pi^1_C(0^{2n})$ and then computes $x_2 = \pi^2_C(t)$ for the lexicographically smallest $t \in \{0,1\}^{2n}$ such that $C(x_1) = C(x_2)$ for every. Finally, $\mathsf{ColFinder}^f$ outputs x_1, x_2.

Using $\mathsf{ColFinder}^f$ it is easy to break any collision-resistant hash function.

Lemma 8. *There exists a probabilistic polynomial-time algorithm $\mathcal{A}$ such that for any function f, any $n \in \mathbb{N}$ and any oracle-aided circuit C^f mapping strings of length $2n$ to strings of length n, it holds that*

$$\Pr_{\mathsf{ColFinder}}\left[\begin{array}{l} x_1, x_2 \text{ are distinct and} \\ C^f(\sigma, x_1) = C^f(\sigma, x_2) \end{array} \middle| (x_1, x_2) \leftarrow \mathcal{A}^{f,\mathsf{ColFinder}}(1^n, C)\right] \geq \frac{3}{4}.$$

Proof. Fix n and f. The algorithm $\mathcal{A}$, on input C, sends it to the oracle $\mathsf{ColFinder}^f$, and receives back a pair (x_1, x_2). By the definition of $\mathsf{ColFinder}^f$, it holds that $C(x_1) = C(x_2)$.

Since C maps strings of length $2n$ to strings of length n, there are at most $n \cdot 2^n$ values of $x \in \{0,1\}^{2n}$ for which $|f^{-1}(f(x))| \leq n$. Therefore, for at least $2^{2n} - n \cdot 2^n$ values of x it holds that $|f^{-1}(f(x))| > n$. Namely,

$$\Pr_{x\leftarrow\{0,1\}^n}\left[|f^{-1}(f(x))| > 1\right] \geq 1 - \frac{n}{2^n}.$$

So, with probability $1 - \frac{n}{2^n}$ the first x_1 chosen by $\mathsf{ColFinder}^f$ is one such that $|f^{-1}(f(x))| > n$. Then, x_2 is chosen uniformly at random from the set of x's in $f^{-1}(f(x))$. The probability of choosing $x_2 \neq x_1$ is at least $(n-1)/n$. Altogether, the probability of choosing two distinct (colliding) values x_1, x_2 is at least $(1 - n/2^n) \cdot (n-1)/n \geq 3/4$. □

Next, we show that f is a 3-MCRH hash function family even in the presence of $\mathsf{ColFinder}$.

Lemma 9. *For every probabilistic oracle-aided algorithm $\mathcal{A}$ that makes polynomially-many queries, there exists a negligible function $\mathsf{negl}(\cdot)$ such that*

$$\Pr_{f,\mathsf{ColFinder}}\left[\begin{array}{c} x_1, x_2, x_3 \text{ are distinct and} \\ f(x_1) = f(x_2) = f(x_3) \end{array} \middle| (x_1, x_2, x_3) \leftarrow \mathcal{A}^{\Gamma}(1^n) \right] \leq \mathsf{negl}(n).$$

We show how Theorem 24 follows from Lemmas 8 and 9 and then in Section 6.6.1 prove Lemma 9.

Proof of Theorem 24. Assume there exists a black-box construction (see Definition 14) of a 3-MCRH function from a CRH function $\mathcal{H}'$ specified by $(\mathcal{H}.\mathsf{G}, \mathcal{H}.\mathsf{E}, M)$.

By Lemma 8, there exists an oracle-aided algorithm $\mathcal{A}$ and a polynomial $p(\cdot)$, such that for every f, it holds that

$$\Pr\left[\begin{array}{c} x_1, x_2 \textit{ are distinct and} \\ \mathcal{H}'.\mathsf{E}^f(\sigma, x_1) = \mathcal{H}'.\mathsf{E}^f(\sigma, x_2) \end{array} \middle| \begin{array}{c} \sigma \leftarrow \mathcal{H}'.\mathsf{G}^f(1^n) \\ (x_1, x_2) \leftarrow \mathcal{A}^{f,\mathsf{ColFinder}}(1^n, \sigma) \end{array} \right] \geq \frac{1}{p(n)}.$$

Definition 14 says that we can use $\mathcal{A}$ to get an algorithms that finds a 3-wise collision relative to f. Specifically, there exists a probabilistic polynomial-time algorithm M that has oracle access to $\mathcal{A}$, and a polynomial $p'(\cdot)$ such that

$$\Pr_{\mathsf{ColFinder}}\left[\begin{array}{c} x_1, x_2 x_3 \textit{ are distinct and} \\ f(x_1) = f(x_2) = f(x_3) \end{array} \middle| (x_1, x_2, x_3) \leftarrow M^{\mathcal{A},f}(1^n) \right] \geq \frac{1}{p'(n)}.$$

Viewing $M^{\mathcal{A}}$ as a single procedure, its total running time is bounded by some fixed polynomial. This is because every operation of M can be an oracle query to $\mathcal{A}$ with a security parameter polynomial in n and the running time of $\mathcal{A}$ is polynomial. This is a contradiction to Lemma 9. □

6.6.1 Proof of Lemma 9

We prove a slightly stronger lemma, namely, we show that for every algorithm $\mathcal{A}$ that performs polynomially many queries, and every fixing of the oracle f for all inputs except n, it holds that

$$\Pr_{f_n,\mathsf{ColFinder}}\left[\begin{matrix}x_1,x_2,x_3 \text{ are distinct and}\\ f(x_1)=f(x_2)=f(x_3)\end{matrix}\middle|(x_1,x_2,x_3)\leftarrow\mathcal{A}^{f,\mathsf{ColFinder}}(1^n)\right]\leq \mathsf{negl}(n).$$

Denote by y the value of $f(x_1)$ (which is equal to the value of $f(x_2)$ and $f(x_3)$). Consider an algorithm $\mathcal{A}$ that has oracle access to f and ColFinder. It can gain information about x's that map to y under f_n either through direct queries to f_n or via indirect queries made by ColFinder to f_n.

We formalize the latter by defining an event 3-multi-hit that occurs when one of the f_n gates during the computation of ColFinder hits a value y that closes a 3-wise collision under f_n.

Consider a query C^f to ColFinder with output (w_1, w_2). Without loss of generality, we assume that A directly queries f_n on all elements in the computation of $C^f(w_1)$ and $C^f(w_2)$. Moreover, we assume that $\mathcal{A}$ never performs the same query twice.

Definition 41 (3-multi-hit)**.** *Let C^f be a query to* ColFinder*, and let $x_1,\ldots,x_q$ be all the previous f_n queries performed by $\mathcal{A}$ with responses $y_1,\ldots,y_q$. The query of C^f to* ColFinder *outputs (w_1,w_2). Let $x_{q+1},\ldots,x_{q'}$ be the queries performed by the computation of $C^f(w_1)$ and $C^f(w_2)$ and let $y_{q+1},\ldots,y_{q'}$ be the responses.*

We say that the query C produced a 3-multi-hit *if there exist $1\leq i_1<i_2<i_3\leq q'$ and $q<i_k$ such that $y_{i_1}=y_{i_2}=y_{i_3}$.*

Moreover, we define the event $\mathcal{A}$-3-wins that occurs when the adversary $\mathcal{A}$ is successful in finding a 3-wise collision relative to f_n.

Definition 42 ($\mathcal{A}$-3-wins)**.** *We say that the event $\mathcal{A}$-3-wins occurred if x_1,x_2,x_3 are distinct and $f_n(x_1)=f_x(x_2)=f_n(x_3)$ where $(x_1,x_2,x_3)\leftarrow\mathcal{A}^{f,\mathsf{ColFinder}}(1^n)$.*

Using the above definitions, we can write

$$\Pr_{f_n,\mathsf{ColFinder}}[\mathcal{A}\text{-3-wins}]=\Pr_{f_n,\mathsf{ColFinder}}[\mathcal{A}\text{-3-wins}\wedge\text{3-multi-hit}]+\Pr_{f_n,\mathsf{ColFinder}}\left[\mathcal{A}\text{-3-wins}\wedge\overline{\text{3-multi-hit}}\right]$$

The proof proceeds by bounding both of these terms. Bounding the second term is done by a compression argument. We show that if 3-multi-hit does not occur and nevertheless $\mathcal{A}$ successfully finds a 3-wise collision (with noticeable probability), then f_n can be succinctly represented given $\mathcal{A}$. This is a contradiction to the fact that f_n is a random function that does not admit such short representation. To bound the first term, we reduce it to the second one by showing that any occurrence of a 3-multi-hit can be simulated. That is, we show that if $\mathcal{A}$ succeeds in finding a 3-wise collision with probability $\epsilon(n)$, then there exists a machine M that succeeds in finding a 3-wise collision (with about the same probability as $\mathcal{A}$ and with proportional query complexity) but without producing any 3-multi-hit. This is summarized in the following two claims:

Claim 11. *For every q-query algorithm $\mathcal{A}^{f,\mathsf{ColFinder}}$, where $q(n)$ is a polynomial, it holds that*

$$\Pr_{f_n,\mathsf{ColFinder}}\left[\mathcal{A}\text{-3-wins} \wedge \overline{\text{3-multi-hit}}\right] \leq \mathsf{negl}(n).$$

Claim 12. *Fix the oracle f and a poly-query algorithm $\mathcal{A}^{f,\mathsf{ColFinder}}$, There exists a poly-query oracle-aided machine M procedure such that if for a polynomial $q(\cdot)$ such that*

$$\Pr_{f_n,\mathsf{ColFinder}}\left[\mathcal{A}\text{-3-wins} \wedge \text{3-multi-hit}\right] \geq \frac{1}{q(n)}$$

for infinitely many n's, then for some polynomial $q'(\cdot)$

$$\Pr_{f_n,\mathsf{ColFinder}}\left[M\text{-3-wins} \wedge \overline{\text{3-multi-hit}}\right] \geq \frac{1}{q'(n)}$$

for infinitely many n's.

From Claims 11 and 12 we can complete the proof of Theorem 24. Assume that there exists a poly-query algorithm $\mathcal{A}$, and a fixing of the oracle f for all inputs except n, such that for a polynomial $q(\cdot)$ it holds that

$$\Pr_{f_n,\mathsf{ColFinder}}\left[\mathcal{A}\text{-3-wins}\right] > 1/q(n)$$

for infinitely many n's. This means that

$$1/q(n) < \Pr_{f_n,\mathsf{ColFinder}}\left[\mathcal{A}\text{-3-wins} \wedge \text{3-multi-hit}\right] + \Pr_{f_n,\mathsf{ColFinder}}\left[\mathcal{A}\text{-3-wins} \wedge \overline{\text{3-multi-hit}}\right].$$

By Claim 11, the second term is bounded by some negligible term $\mathsf{negl}(n)$. This implies that

$$\frac{1}{q(n)} - \mathsf{negl}(n) < \Pr_{f_n,\mathsf{ColFinder}}\left[\mathcal{A}\text{-3-wins} \wedge \text{3-multi-hit}\right].$$

By averaging,

$$\Pr_{f_n}\left[\Pr_{\mathsf{ColFinder}}\left[\mathcal{A}\text{-3-wins} \wedge \text{3-multi-hit}\right] \geq \frac{1}{8q(n)}\right] \geq \frac{1}{8q(n)}.$$

Let us call f_n *good* if the inner event happens. By Claim 12, we get a poly-query oracle-aided machine M such that for every good f_n it holds that M finds a 3-wise collision without making 3-multi-hit occurring. Thus, we get that for some polynomial $q'(\cdot)$,

$$\begin{aligned}
&\Pr_{f_n,\mathsf{ColFinder}}\left[\mathcal{A}\text{-3-wins} \wedge \overline{\text{3-multi-hit}}\right] \\
&\geq \Pr_{f_n}[f_n \text{ is good}] \cdot \Pr_{\mathsf{ColFinder}}\left[M\text{-3-wins} \wedge \overline{\text{3-multi-hit}} \mid f_n \text{ is good}\right] \\
&\geq \frac{1}{8q(n)} \cdot \frac{1}{q'(n)}
\end{aligned}$$

which is a contradiction to Claim 11 since this is not a negligible function.

The proofs of Claims 11 and 12 appear in Sections 6.6.2 and 6.6.3, respectively.

6.6.2 Proof of Claim 11

Fix ColFinder, an inverse-polynomial ϵ and recall that f is fixed at all input lengths except n. For any prefix s of size $n/2$ let $f_n|_s(x) = f(s \circ x)$. We show that for every f_n if

$$\Pr_{s \leftarrow \{0,1\}^{n/2}}[\mathcal{A}^{f_n|_s}\text{-3-wins} \wedge \overline{\text{3-multi-hit}}] \geq \epsilon,$$

then f_n has a succinct description.[14]

Succinct description of f_n. The description of f_n consists of a truth-table Z a set of prefixes S, a set of images Y. We will be running $\mathcal{A}$ on $f_n|_s$ for many different prefixes s. Notice that if $f_n|_s(x) = f_n|_s(x')$, then $f_n(s \circ x) = f_n(s \circ x')$. Denote by S the set of prefixes s for which $\mathcal{A}$ successfully finds a collision for $f_n|_s$.

For every $s \in S$, we follow the computation of $\mathcal{A}^{f_n|_s,\text{ColFinder}}$ and define $F_s = \{x_1, \ldots, x_\ell\}$ to be the set of all inputs of $f_n|_s$-gates during this computation. Finally, $\mathcal{A}^{f_n|_s,\text{ColFinder}}$ outputs a 3-wise collision $\{x_1, x_2, x_3\}$. Assume without loss of generality, that $\mathcal{A}$ directly queried x_1, x_2 and x_3, and never queries the same input twice. Let $I_s = \{i_1, i_2, i_3\}$ be the indices of the queries performed by $\mathcal{A}$ to x_1, x_2, x_3, respectively. Assume that $i_1 < i_2 < i_3$. Let $X_s = \{s \circ x_3\}$. Continue to the next iteration which is defined analogously.

Let $X = \cup_{s \in S} X_s$. Notice that $|X| = |S|$, and that since ℓ is polynomial in n we can write each of the indexes using $c \log n$ bits for some constant $c \in \mathbb{N}$. Moreover, by the assumption we have that $|S| \geq \epsilon 2^{n/2}$.

We write f_n as follows: first we write the set S, then for each $s \in S$ we write I_s, and finally we write the truth-table of $Z = \{0,1\}^{2n} \setminus X$. The total size (in bits) is at most

$$\begin{aligned}
|f_n| &\leq \log \binom{2^{n/2}}{|S|} + |S| 3c \log n + (2^{2n} - |X|)n \\
&\leq (n/2)|S| + 2|S| - |S| \log |S| + |S| 3c \log n + n 2^{2n} - n|S| \\
&= n 2^{2n} - |S|(\log |S| - 3c \log n + n/2 - 2) \\
&\leq n 2^{2n} - n.
\end{aligned}$$

We next show that this representation is sufficient to answer f_n and ColFinder queries. Given x as input, either we know it explicitly (via the table Z), or we reconstructed it before, or we "hit" it by one of the indices i_1, i_2, i_3. For each $s \in S$, taken in

[14] More accurately, what happens is that we succinctly describe f_n given that $\mathcal{A}$ successfully find collisions in $f_{n/2}$. Since $\mathcal{A}$ works for infinitely many n's this is without loss of generality as the $n/2$ can be replaced with any other fraction. Alternatively, we could succinctly represent f_{2n} given that $\mathcal{A}$ finds collisions in f_n.

lexicographical increasing order, the simulator reads I_s and runs $\mathcal{A}^{f_n|_s,\mathsf{ColFinder}}$. The latter requires answering $f_n|_s$-queries and ColFinder queries. Finally, $\mathcal{A}$ outputs x_1, x_2, x_3, where x_1 has its value in the truth-table Z, and thus we can reconstruct the values for x_2 and x_3.

Answering $f_n|_s$ queries. We show how to simulate a direct $f_n|_s$ query made by $\mathcal{A}$. Assume that $\mathcal{A}$ makes such a query on input x. If it is already known (either because $s \circ x$ is in Z or because we already reconstructed it), then the simulator can return it. If it is i_3-th query, then the simulator takes the answer of the i_1-th query and answers with the same value.

Answering ColFinder-queries. On input oracle-aided circuit $C = C_i^{f_n|_s}$, the simulator computes $w_i = \pi_1^C(0)$ and begins with evaluating $C(w_i)$. Then, it enumerates over all $t \in \{0,1\}^{2n}$ in lexicographically increasing order, and stops with the first $w_i' = \pi_2^C(t)$ for which $C(w_i)$ can be computed (*i.e.*, it is able to answer all $f_n|_s$-queries as discussed above) and that $C(w_i) = C(w_i')$. It then answers to $\mathcal{A}$ with the pair (w_i, w_i').

We now show that the simulator can evaluate $C(w_i), C(w_i')$ and answer the queries to $f_n|_s$. On a query x to $f_n|_s$-gate, we have:

1. If $s \circ x \notin X$ then the value $f_n|_s(x)$ is given by Z.
2. If $s \circ x \in X \setminus X_s$, then this cannot happen as $\mathcal{A}$ works on a different oracle $f_n|_s$ for different s's.
3. If $s \circ x \in X_s$, then this is impossible, as we condition on the event that 3-multi-hit does not occur.

Concluding the proof. Assume that

$$\Pr_{f_n,\mathsf{ColFinder}}\left[\mathcal{A}\text{-3-wins} \wedge \overline{\text{3-multi-hit}}\right] \geq 2\epsilon,$$

for some inverse-polynomial ϵ. Then, by averaging we get that

$$\Pr_{f_n}[\Pr_{s\leftarrow\{0,1\}^{n/2}}[\mathcal{A}^{f_n|_s}\text{-3-wins} \wedge \overline{\text{3-multi-hit}}] \geq \epsilon] \geq \epsilon.$$

We have shown that using $\mathcal{A}$ we can represent many of the possible $2^{n\cdot 2^{2n}}$ functions. Thus, the fraction of functions f_n for which

$$\Pr_{s\leftarrow\{0,1\}^{n/2}}[\mathcal{A}^{f_n|_s}\text{-3-wins} \wedge \overline{\text{3-multi-hit}}] \geq \epsilon,$$

holds is at most

$$\frac{2^{n\cdot 2^{2n}-n}}{2^{n\cdot 2^{2n}}} = 2^{-n}.$$

This implies that any algorithm $\mathcal{A}$ that makes polynomially many queries to f and ColFinder, where its oracle queries to ColFinder are bounded to circuits of polynomial size, it holds that

$$\Pr_{f_n,\mathsf{ColFinder}}[\mathcal{A}\text{-3-wins} \wedge \overline{\text{3-multi-hit}}] \leq \mathsf{negl}(n) + 2 \cdot 2^{-n} \leq \mathsf{negl}(n).$$

6.6.3 Proof of Claim 12

The algorithm M simulates the execution of $\mathcal{A}$ except for one change: whenever $\mathcal{A}$ makes a ColFinder-query with some circuit C^f, the algorithm M first tries to simulate the query and make a 3-multi-hit by itself as follows: It evaluates $C^f(z)$ and $C^f(z')$ on random z and z' and without submitting the queries to ColFinder. If during the computation there three inputs x_1, x_2, x_3 (from the computation of $C^f(z)$ and $C^f(z')$ together with queries already performed in the past) that have the same output y, then M halts and outputs this collision. Otherwise, if a 3-multi-hit does not occur, M passes the circuit C^f to ColFinder and outputs whatever it returns.

The algorithm M makes at most as many calls to ColFinder as $\mathcal{A}$. Moreover, M may perform at most $\mathsf{poly}(n)$ queries to f as direct queries of $\mathcal{A}$ to f, and additional $\mathsf{poly}(n)$ many queries as a result of the simulation of ColFinder.

We split the proof into several cases related to how $\mathcal{A}$-3-wins happen. Recall that $\mathcal{A}$ makes queries to ColFinder and direct queries to f and finally outputs (with good probability) a triple of items x_1, x_2, x_3 that map under f to the same value. This collision happens due to some query to ColFinder which completes a 3-wise collision.

1. The first case is that the three values x_1, x_2, x_3 all happened during a *single* ColFinder query. Namely, $\mathcal{A}$ submitted a circuit C to ColFinder and during the computation of ColFinder these three queries were made.

   We show that this event has very low probability of happening so we can ignore it.

2. The second case is that the two values x_2, x_3 were queried during a *single* ColFinder query and joined (in the sense that they all have the same $f(\cdot)$ value) with a single x_1 that was queried sometime before.

   We show that if this event happens, then we can simulate the execution and make sure that we find the same triple but without making the last ColFinder query.

3. The second case is that one value x_3 was queried during a ColFinder query and joined with a pair x_1, x_2 that was queried sometime before.

   As in the second case, we show that if this event happens, then we can simulate the execution and make sure that we find the same triple but without making the last ColFinder query.

We write

$$\Pr_{f_n,\mathsf{ColFinder}}[\mathcal{A}\text{-3-wins} \wedge \text{3-multi-hit}] = \sum_{i=1}^{3} \Pr_{f_n,\mathsf{ColFinder}}[\mathcal{A}\text{-3-wins} \wedge \text{3-multi-hit} \wedge \text{case } i \text{ happens}]$$

and prove the claim for each i separately. For $i = 1$, we obtain an algorithm M that is able to output 2 colliding values such that non of them were ever in the output of any f or ColFinder query. Following similar lines to our compression lemma (see Section 6.6.2), one can show that such an adversary exists with only negligible probability. So, we ignore this event for the rest of the proof and assume that one of two happens: case 2 or case 3. This means that in the triple that $\mathcal{A}$ outputs (that collide), at least one input was queried on before the last query. The last query completes this triple by querying on either a colliding pair (case 2) or a single element (case 3).

Fix f, and denote by $C_1, \ldots, C_q$, where $q = q(n)$ is some polynomial, the random variables corresponding to the circuits with which $\mathcal{A}$ queries ColFinder. Denote by $(x_1^1, x_2^1), \ldots, (x_1^q, x_2^q)$ the corresponding replies of ColFinder. For every $i \in [q]$, denote by α_i the probability that x_1^i produces a 3-multi-hit. That is

$$\alpha_i = \Pr[x_1^i \text{ produces a 3-multi-hit}].$$

Recall that for every $i \in [q]$ the marginal distributions of x_1^i and x_2^i are uniform. Thus,

$$\forall i \in [q]\colon \Pr[x_1^i \text{ produces a 3-multi-hit}] = \Pr[x_2^i \text{ produces a 3-multi-hit}].$$

For every $i \in [q]$, denote by JUMP_i the event that $\alpha_i > 1/(20q^2)$ and let $\mathsf{JUMP} = \bigcup_{i=1}^{q} \mathsf{JUMP}_i$. Using the above notation, we have that

$$\begin{aligned}\Pr_{\mathsf{ColFinder}}[\mathsf{JUMP}] &\geq \Pr_{\mathsf{ColFinder}}[\text{3-multi-hit}] - \Pr_{\mathsf{ColFinder}}[\text{3-multi-hit} \mid \overline{\mathsf{JUMP}}] \\ &\geq \frac{1}{q} - \frac{1}{10q} \geq \frac{1}{2q},\end{aligned}$$

where the second inequality follows from the assumption and from the following claim.

Claim 13. $\Pr_{\mathsf{ColFinder}}[\text{3-multi-hit} \mid \overline{\mathsf{JUMP}}] \leq \frac{1}{10q}$.

Proof. We divide into several cases depending on how many collisions we obtained in the last query.

$$\begin{aligned}\Pr_{\mathsf{ColFinder}}[\text{3-multi-hit} \mid \overline{\mathsf{JUMP}}] = &\Pr_{\mathsf{ColFinder}}[\text{3-multi-hit} \wedge \text{ case 2} \mid \overline{\mathsf{JUMP}}] + \\ &\Pr_{\mathsf{ColFinder}}[\text{3-multi-hit} \wedge \text{ case 3} \mid \overline{\mathsf{JUMP}}].\end{aligned}$$

We analyze each term separately conditioned on $\overline{\mathsf{JUMP}}$. Observe that when 3-multi-hit $\wedge$ case 3 occurs, then it must be that our simulator will be able to simulate this query and obtain the collision by itself:

$$\begin{aligned}\Pr_{\mathsf{ColFinder}}[\text{3-multi-hit} \wedge \text{ case 3}] \leq &\sum_{i=1}^{q} \Pr[\text{One of } x_1^i, x_2^i \text{ produces a 3-multi-hit}] \\ \leq &2 \cdot q \cdot \frac{1}{30q^2} = \frac{1}{20q}.\end{aligned}$$

The case 3-multi-hit $\wedge$ case 3 is similar and we have that

$$\begin{aligned}\Pr_{\mathsf{ColFinder}}[\text{3-multi-hit} \wedge \text{ case 2}] \leq &\sum_{i=1}^{q} \Pr[\text{One of } x_1^i, x_2^i \text{ produces a 3-multi-hit}] \\ \leq &2 \cdot q \cdot \frac{1}{30q^2} = \frac{1}{20q}.\end{aligned}$$

Summing up, we get that

$$\Pr_{\mathsf{ColFinder}}[\text{3-multi-hit} \mid \overline{\mathsf{JUMP}}] \leq \frac{1}{10q}.$$

□

Assume for now that the event JUMP occurs and denote by i^* the minimal $i \in [q]$ for which JUMP_i occurs. Consider the following two events:

1. None of the $C_1, \ldots, C_{i^*-1}$ produced a 3-multi-hit. Recall that the events $\mathsf{JUMP}_1, \ldots, \mathsf{JUMP}_{i^*-1}$ did not happen since i^* is minimal. The probability of this event happening is at least $1 - 1/(10q)$ (similarly to the computation in Claim 13).

2. Given C_{i^*}, the algorithm M produces a 3-multi-hit (without submitting the query to ColFinder). This event happens with probability at least $(\alpha_{i^*})^2/2 > (1/(20q^2))^2/2$. In case 2, this happens with probability $\alpha_{i^*} > 1/(20q^2)$, but in case 3, we need to succeed in both simulation steps of M and to output different values.

Notice that these two events are independent since ColFinder uses an independent permutation per circuit. Thus we get that

$$\begin{aligned}&\Pr_{\mathsf{ColFinder}}[M^{f,\mathsf{ColFinder}}(1^n) \text{ finds a 3-wise collision} \wedge \overline{\text{3-multi-hit}}] \\ &\geq \Pr_{\mathsf{ColFinder}}[M^{f,\mathsf{ColFinder}}(1^n) \text{ finds a 3-wise collision} \wedge \overline{\text{3-multi-hit}} \mid \mathsf{JUMP}] \cdot \Pr_{\mathsf{ColFinder}}[\mathsf{JUMP}] \\ &\geq \left(1 - \frac{1}{10q}\right) \cdot \frac{1}{2(20q^2)^2} \cdot \frac{1}{2q} \geq \frac{1}{q^7}.\end{aligned}$$

6.7 UOWHFs, MCRHs and Short Commitments

We explore the relationship between short commitments, MCRHs and universal one-way hash functions (see Definition 10). Our main message is that short commitment protocols (with some requirements listed below) directly imply UOWHFs. The transformation is efficient in the sense that a description of a hash function corresponds to the messages sent by the receiver and evaluation of the function is done by executing the protocol. In some cases this gives a way to construct a UOWHF which is more efficient than the direct construction based on one-way functions or permutations [NY89, Rom90] (see comparison below). In Remark 7 we sketch how to get an efficient construction of a UOWHF that operates on inputs of fixed length directly from an MCRH.

Theorem 25. *Any short commitment protocol in which the receiver is public-coin yields a universal one-way hash function with the following properties:*

1. *The key size is the total number of (public) coins sent by the receiver.*
2. *The length of inputs that the UOWHF supports is the length of messages the commitment protocol commits to and the length of the output is the amount of bits sent from the sender to the receiver.*
3. *The evaluation of a hash function amounts to a single execution of the commitment protocol.*

Plugging in our construction of short commitments from MCRH functions from Theorem 22, we obtain a new construction of a UOWHF for messages of length $n2^d$ starting with a k-MCRH for a polynomial $k = k(n)$. The key size in the resulting family is proportional to the number of bits sent from the receiver to the sender: $\ell + O(d \cdot n / \log n)$ bits, where ℓ is the size of an MCRH key.[15]

Using our construction of short commitments from MCRH functions from Theorem 23, we get a new construction of a UOWHF for messages of length $n2^d$ starting from a k-MCRH for any constant k. The key size in the resulting family is $\ell + O(n)$ bits. Notice that this term is independent of d and the hidden constant in the big "O" is pretty small.[16]

Comparison with previous constructions. Starting with a standard collision resistant hash function on short inputs, it is known how a collision resistant hash function on long inputs (based on the so called Merkle-Damgård iterated construction) [Mer89b, Dam89]. This directly implies a UOWHF. The key size in the resulting construction is optimal: it is just a key for a single collision resistant hash. However, when starting with weaker building blocks the key size grows.

[15] The overhead in the key size can be improved if the pairwise hash function is replaced by an almost uniform hash function as described in Remark 5.

[16] The concrete constant in our scheme is roughly 6 but we did not try to optimize it further.

Naor and Yung [NY89] suggested a solution based on a tree hash (similar to a construction of Wegman and Carter for universal hash functions [WC81]). In their scheme, hashing a message of length $n2^d$ is done by a balanced tree such that in the i-th level $n2^{d-i}$ bits are hashed into $n2^{d-i-1}$ bits by applying the same basic compression function 2^{d-i-1} times. Each level in the tree requires its own basic compression function which results with a total of d keys for the basic UOWHF. Thus, the total size of a key in the resulting function is $d\ell$ bits, where ℓ is the bit size of a key in the basic UOWHF.

In case that the keys of the basic scheme (ℓ above) are rather long, Shoup [Sho00], following on the XOR tree hash of Bellare and Rogaway [BR97], offered the following elegant scheme to transform a fixed-input UOWHF into a UOWHF that supports arbitrary long inputs. Given an input of 2^d blocks each of size n, for $i = 1, \ldots, 2^d$ we compute $c_i = h((c_{i-1} \oplus s_{\kappa(i)}) \circ m_i)$, where $s_0, \ldots, s_d$ are uniformly random "chaining variables", and $\kappa(i)$ chooses one of the elements $s_0, \ldots, s_d$.[17] Thus, the total size of a key in the resulting function is $\ell + (d+1)n$ bits.

Remark 6 (Two-round short commitments)**.** *If the short commitment protocol consists of two rounds (a message from the receiver to the sender and then a message back), then we actually get a collision resistant hash. The description of the hash is the message sent by the receiver and evaluation is done by simulating the (single) message sent by the sender.*

Proof of Theorem 25. Assume the existence of a short commitment protocol $(\mathcal{S}, \mathcal{R}, \mathcal{V})$ in which the receiver is public-coin and the sender is deterministic. The latter is without loss of generality due to the perfect completeness of the protocol. Assume that the protocol allows to commit on strings of length $\ell_{\mathsf{in}} = \ell_{\mathsf{in}}(n)$. Denote the length of the randomness used by the receiver by $\ell_{\mathsf{r}} = \ell_{\mathsf{r}}(n)$. Denote the length of the transcript by ℓ_{trns} (this is an upper bound on the maximum length of the transcript over all possible inputs and messages from the receiver). Assume that the opening of the sender opening is just sending the committed input x. The verifier then decides whether the opening is valid or not based on the input of the sender and on the (public) randomness of the verifier.

We construct a universal one-way hash function family $\mathcal{H} = \{\mathcal{H}_n\}_{n \in \mathbb{N}}$, where each function h in the family $\mathcal{H}_n$ gets as input strings of length ℓ_{in} and outputs a string of length ℓ_{trns}. This results with a compressing function since the length of a commitment is shorter than the length of the input (i.e., $\ell_{\mathsf{in}} < \ell_{\mathsf{trns}}$). The description of a hash function $h \in \mathcal{H}_n$ is a string of length ℓ_{r} that contains the randomness used by the receiver. Evaluation of h on input $x \in \{0,1\}^{\ell_{\mathsf{in}}}$ is the transcript of an execution of the commitment protocol between the sender and the receiver (where the receiver uses randomness coming from the description of the hash function):

$$h(x) = \langle \mathcal{S}(1^n, x), \mathcal{R} \rangle.$$

[17] The function $\kappa(i)$ counts the number of times 2 divides i, that is, for $i \geq 1$, $\kappa(i)$ is the largest integer κ such that 2^κ divides i.

We prove that this function family is a universal one-way hash function. In the latter, an adversary first submits a challenge x^*, then gets to see a description of a random hash function h from the family and finally needs to output (with noticeable probability) an $x \neq x^*$ such $h(x) = h(x^*)$. Let us assume that such an adversary $\mathcal{A}$ exists and it succeeds in finding such an x with noticeable probability $\epsilon > 0$. We design a sender $\mathcal{S}^*$ that breaks the computational-binding of the short commitment protocol.

The sender chooses x^* by simulating the choice of the latter by $\mathcal{A}$. Then, $\mathcal{S}^*$ simulates the execution of $\mathcal{S}$ when interacting with $\mathcal{R}$ and committing on the string x^*. Now, given the randomness of the receiver (here we use the fact that the receiver is public coin), we use it as a description of a hash function in $\mathcal{H}$ and simulate the execution of $\mathcal{A}$ with this function and output the pair of openings (x, x^*). With probability ϵ, the adversary $\mathcal{A}$ will output an $x \neq x^*$ for which $h(x) = h(x^*)$. In other words, $\mathcal{A}$ will output (with probability ϵ) an x with which the transcript of the commitment protocol would have been *the same*. Due to perfect completeness, the verifier of the commitment protocol must accept this value as a valid opening, as well. This is a contradiction to the computational-binding of the protocol.

□

Remark 7 (UOWHF on fixed-length inputs)**.** *We can get a very efficient construction of a UOWHF family that operates on inputs of fixed length, say $2n$ and outputs $< 2n$ bits. For this we can use the construction of short commitments described in Section 6.2 which allows to commit to a string of length $2n$ using $< 2n$ bits. The construction is fairly efficient as it requires merely an evaluation of an almost-universal hash function (see Definition 3) in addition to the MCRH.*

Let $\mathcal{H} = \{\mathcal{H}_n \colon \{0,1\}^{2n} \to \{0,1\}^n\}_{n\in\mathbb{N}}$ be a k-MCRH for a polynomial $k = k(n)$. Assume that every family member of $\mathcal{H}$ is of length ℓ bits. Let $\mathcal{G} = \{g \colon \{0,1\}^{2n} \to \{0,1\}^{0.3n}\}$ be a δ-almost universal function family, where $\delta = 2^{-0.3n}$ and each member is described by $O(\log n)$ bits. We define a function family ensemble $\mathcal{F} = \{\mathcal{F}_n \colon \{0,1\}^{2n} \to \{0,1\}^{1.6n}\}_{n\in\mathbb{N}}$ as follows:

1. *Each member $f \in \mathcal{F}_n$ consists of a pair: an $h \in \mathcal{H}_n$ and a $g \in \mathcal{G}$.*

2. *For a member $f = (h, g) \in \mathcal{F}_n$, and an input $x \in \{0,1\}^{2n}$, we define $f(x) = h(x) \circ g(x)$.*

Using our technique of partial rewinding of the adversary to collect multiple collisions, we can show that if $\mathcal{H}$ is a k-MCRH for a polynomial $k = k(n)$, then $\mathcal{F}$ is a UOWHF, where each member of $\mathcal{F}_n$ can be described by a member h of $\mathcal{H}_n$ and additional $O(n)$ bits, and evaluation of a member of $\mathcal{F}_n$ consists of a single evaluation of a member in $\mathcal{H}$.

6.8 Chapter Appendix

6.8.1 Multi-Pair Collision Resistance

We have considered so far a weakening of the security definition of a standard CRH called *multi*-collision-resistance, where the challenge is to find a k-wise collision. In this section, we consider a *different* relaxation of collision resistance we call *multi-pair-collision-resistance*, where the challenge is to find *arbitrary* k distinct pairs of inputs that collide. More precisely, an efficient function family ensembles $\mathcal{H} = \{\mathcal{H}_n \colon \{0,1\}^{3n} \to \{0,1\}^n\}_{n\in\mathbb{N}}$ is a secure *k-multi-pair-collision-resistant hash* (MPCRH) if it is computationally hard to find $k = k(n)$ distinct pairs $(x_1, y_1), \ldots, (x_k, y_k)$ such that $h(x_1) = h(y_1), \ldots, h(x_k) = h(y_k)$.

It is immediate that a standard CRH is also an MPCRH, and we prove the other direction next. Let $\mathcal{H}$ be a secure k-MPCRH for a polynomial $k = k(n)$. Define the family

$$\mathcal{H}' = \{h' = (h, s)\}_{h\in\mathcal{H}, s\in\{0,1\}^{n/2}}$$

to be a family of function mapping $2n$ bits to n bits, where for $h' = (h, s)$ we define $h'(x) = h(s \circ x)$.

Lemma 10. *$\mathcal{H}'$ is a secure CRH.*

Proof. Assume that there exist an adversary $\mathcal{A}$ that can break the security of $\mathcal{H}'$. That is, there exits a polynomial $p(\cdot)$ such that

$$\Pr\left[\begin{array}{c|c} (x_1, y_1), \ldots, (x_k, y_k) \text{ are distinct and} & h \leftarrow \mathcal{H}_n \\ h(x_1) = h(y_1), \ldots, h(x_k) = h(y_k) & ((x_1, y_1), \ldots, (x_k, y_k)) \leftarrow \mathcal{A}(h) \end{array}\right] \geq 1/p(n).$$

We construct $\mathcal{A}'$ that breaks the security of the MPCRH. The algorithm $\mathcal{A}'$ acts as follows:
$\mathcal{A}'(h)$:

1. Let $L = \emptyset$ be an empty list.
2. Repeat $T = (nkp(n))^2$ times:
   (a) Sample $s \leftarrow \{0,1\}^n$ at random.
   (b) Run $\mathcal{A}(h')$, where $h' = (s, h)$ to get a pair (x, y).
   (c) If $h(s \circ x) = h(s \circ y)$, then add (x, y) to L.
3. Output L.

Since $\mathcal{A}$ succeeds with probability $\epsilon = 1/p(n)$, a standard Chernoff bound (see e.g., [AS08, §A.1]) gives that $\mathcal{A}'$ will collect k colliding pairs from $\mathcal{A}$ with all but exponentially small probability. Notice that if $s \neq s'$ then the collisions that $\mathcal{A}$ outputs on (h, s) must be distinct from the collisions that it outputs on (h, s'). Thus, $\mathcal{A}'$ will find k distinct pairs of collisions with high probability. □

6.8.2 On TFNP and Multi-CRH

The class TFNP, defined by Megiddo and Papadimitriou [MP91], is the class of all search problems for which a solution is guaranteed to exist for every instance and verifying a solution can be done efficiently. TFNP has attracted extensive attention due to its important syntactic subclasses. The common way of defining such subclasses is via various non-constructive arguments used for proving totality of search problems. Some of the most known subclasses are PPAD (for Polynomial Parity Argument on Directed graphs [Pap94]), PPP (for Polynomial Pigeonhole Principle [Pap94]) and PLS (for Polynomial Local Search [JPY88]).

For many of the subclasses of TFNP we either can show containment of one class in the other, or an oracle separation ruling out the possibility of a black-box reduction from one to the other. Using our separation results from Section 6.6 we get an infinite hierarchy of subclass of TFNP, each one is contained in the next, but separated (relative to an oracle) from the previous one[18]. We define this hierarchy below.

The formal definition of TFNP is given by:

Definition 43 (TFNP)**.** *A* total NP search problem *is a relation $\mathcal{S}(x,y)$ such that it is (i) computable in polynomial (in $|x|$ and $|y|$) time (ii)* total*, i.e. there is a polynomial q such that for every x there exists a y such that $\mathcal{S}(x,y)$ and $|y| \leq q(|x|)$. The set of all total NP search problems is denoted by TFNP.*

The weak pigeon class (PWPP) is a subclass in TFNP defined by the collision finding problem for circuits that compress their input.

Definition 44 (PWPP^n_{n-1} complete problem)**.** *Given a circuit $C\colon \{0,1\}^n \to \{0,1\}^{n-1}$, find $x \neq y$ such that $C(x) = C(y)$. Moreover, we define PWPP $=$ PWPP^n_{n-1}.*

Any hard distribution for PWPP^n_k naturally gives rise for a collision resistant hash function, and vice-versa. Using our notion of MCRH, we define a new class of total problems that we call Multi Pigeonhole Principle with a parameter k, or k-PMPP for short.

Definition 45 (k-PMPP complete problem)**.** *Given a circuit $C\colon \{0,1\}^n \to \{0,1\}^{n/2}$, find k distinct elements $x_1, \ldots, x_k$ such that $C(x_1) = \cdots = C(x_k)$.*

The first thing to notice is that k-PMPP is a subclass of TFNP for any constant $k \in N$. Suppose that $C\colon \{0,1\}^n \to \{0,1\}^{n/2}$. If there is no k-wise collision then each output has at most $k-1$ inputs. Thus, the total number of element in the domain is at most $2^{n/2} \cdot (k-1) < 2^n$. This is of course a contradiction since the number of elements in the domain is exactly 2^n.

Notice that 2-PMPP $\subset$ 3-PMPP $\subset$ 4-PMPP $\ldots$. Moreover, using the separation results from Section 6.6 we get that (under black-box reductions) all of these classes are separated.[19] That is, 3-PMPP $\not\subseteq$ 2-PMPP, 4-PMPP $\not\subseteq$ 3-PMPP, 5-PMPP $\not\subseteq$ 4-PMPP and so on. This concludes the hierarchy.

[18] We thank Bobby Kleinberg for asking us whether such an infinite hierarchy exists inside TFNP.

[19] The results in Section 7 are stated for $k = 3$ but naturally generalize to any k.

The commit protocol between $\mathcal{S}$ and $\mathcal{R}$

The sender $\mathcal{S}$ has string $s = s_1 \dots s_{2^d}$ where $s_i \in \{0,1\}^n$ for all $i \in [2^d]$.

1. $\mathcal{R} \Rightarrow \mathcal{S}$: Sample $h \leftarrow \mathcal{H}$ and send h.
2. $\mathcal{S} \Rightarrow \mathcal{R}$: Compute a Merkle hash-tree T of s using h and send the root-hash y. Let π_v be the hash value in the tree for node $v \in T$, and let $P_i = \left\{\pi_v \circ \pi_{N(v)}\right\}_{v \in \mathsf{path}_i}$ for all $i \in [d]$.
3. $\mathcal{R} \Rightarrow \mathcal{S}$: Sample $g \leftarrow \mathcal{G}$ and send g.
4. $\mathcal{S} \Leftrightarrow \mathcal{R}$: Recursively interact to commit on the string $s' = u_1 \circ \dots \circ u_{2^d}$, where $u_i = \left\{g(\pi_v \circ \pi_{N(v)})\right\}_{v \in \mathsf{path}_i}$. Notice that $|s'| = 2^d \cdot dz = 2^{d'} n$, where $d' = d - (\log n - \log z - \log d)$. Denote the outputs of the sender and receiver by
$$((D_1, \dots, D_{2^{d'} n}), C) \leftarrow \langle \mathcal{S}(1^n, s'), \mathcal{R} \rangle.$$

The output of each party

- $\mathcal{R}$'s output: The receiver $\mathcal{R}$ outputs $\mathsf{com} = (h, y, g, C)$.
- $\mathcal{S}$'s output: The sender $\mathcal{S}$ outputs $(\mathsf{decom}_1, \dots, \mathsf{decom}_{2^d})$, where decom_i is defined as follows: Let i' be the index in s' of the block containing u_i. Set $\mathsf{decom}_i = (s_i, P_i, D_{i'})$.[a]

The local-opening procedure $\mathcal{V}$

The verifier $\mathcal{V}$ has an index $i \in [2^d]$, a decommitment decom_i, and a commitment com.

1. Verify that s_i appears in P_i in the right location.
2. Verify that the values in P_i are consistent with respect to h and y.
3. Compute $u_i = \left\{g(\pi_v \circ \pi_{N(v)})\right\}_{v \in \mathsf{path}_i}$, where $P_i = \left\{\pi_v \circ \pi_{N(v)}\right\}_{v \in \mathsf{path}_i}$. Recursively compute $u'_i \leftarrow \mathcal{V}(i', D_{i'}, C)$ and verify that $u'_i = u_i$.
4. If all tests pass, output s_i. Otherwise, output $\perp$.

[a] We assume without loss of generality that each u_i is contained in a single block (otherwise, we pad the string).

Figure 6.1: Our commitment protocol for strings of length $2^d \cdot n$.

The adversary $\mathcal{A}(1^n, h)$:

1. For all nodes v, set $S_v = \emptyset$ to be the empty set.
2. Send to $\mathcal{S}^*$ the function h and receive a root-hash $y \in \{0,1\}^n$.
3. Do the following $T = 50nk2^d/\epsilon_d^2$ times:
   (a) Sample $g \leftarrow \mathcal{G}$.
   (b) Obtain $(i, \mathsf{decom}_i^0, \mathsf{decom}_i^1, \mathsf{com}) \leftarrow \langle \mathcal{S}^*|_{h,y}(1^n), g\rangle$.
   (c) Parse $\mathsf{decom}_i^0 = (s_i^0, P_i^0, D_{i'}^0)$ and $\mathsf{decom}_i^1 = (s_i^1, P_i^1, D_{i'}^1)$. Parse $\mathsf{com} = (h, y, g, C)$.
   (d) If any of the following occurs, continue to the next iteration:
      i. $\perp \neq \mathcal{V}(i, \mathsf{decom}_i^0, \mathsf{com}) \neq \mathcal{V}(i, \mathsf{decom}_i^1, \mathsf{com}) \neq \perp$.
      ii. $\perp \neq \mathcal{V}(i', D_{i'}^0, C) \neq \mathcal{V}(i', D_{i'}^1, C) \neq \perp$.
   (e) Let v the node of the first (from the root to the leaves) non-trivial collision between P_i^0 and P_i^1. Let $X = \pi_v^0, \pi_{N(v)}^0$ and $Y = \pi_v^1, \pi_{N(v)}^1$ be the values of the collision for P_i^0 and P_i^1, respectively. Add X and Y to S_v.
   (f) If there exists a node v for which $|S_v| \geq k$, then output S_v and halt.
4. Output $\perp$.

Figure 6.2: The algorithm $\mathcal{A}$ to find a k-wise collision in $\mathcal{H}$.

The statistically-hiding commit protocol between $\mathcal{S}$ and $\mathcal{R}$

The sender $\mathcal{S}$ has string $s \in \{0,1\}^N$.

1. $\mathcal{S} \Leftrightarrow \mathcal{R}$: Interact according to $(\widehat{\mathcal{S}}, \widehat{\mathcal{R}})$ to obtain a commitment on a random string $r^1 = r_1 \circ \ldots \circ r_N \leftarrow \{0,1\}^{8\Lambda(n)N}$. Denote the outputs of the sender and receiver by

$$((\mathsf{decom}_1^1, \ldots, \mathsf{decom}_N^1), \mathsf{com}^1) \leftarrow \langle \mathcal{S}(1^n, r^1), \mathcal{R} \rangle,$$

   where decom_i^1 corresponds to the opening of the whole string r_i.

2. $\mathcal{S} \Leftrightarrow \mathcal{R}$: Interact according to $(\widehat{\mathcal{S}}, \widehat{\mathcal{R}})$ to obtain a commitment on a string $r^2 = g_1 \circ \ldots \circ g_N \in \{0,1\}^{8\Lambda(n)N}$, where $g_i \leftarrow \mathcal{G}$ is sampled at random. Denote the outputs of the sender and receiver by

$$((\mathsf{decom}_1^2, \ldots, \mathsf{decom}_N^2), \mathsf{com}^2) \leftarrow \langle \mathcal{S}(1^n, r^2), \mathcal{R} \rangle,$$

   where decom_i^2 corresponds to the opening of the whole string g_i.

3. $\mathcal{S} \Leftrightarrow \mathcal{R}$: Interact according to $(\widehat{\mathcal{S}}, \widehat{\mathcal{R}})$ to obtain a commitment on the string $r^3 = s \oplus (g_1(r_1) \circ \ldots \circ g_N(r_N))$. Denote the outputs of the sender and receiver by

$$((\mathsf{decom}_1^3, \ldots, \mathsf{decom}_N^3), \mathsf{com}^3) \leftarrow \langle \mathcal{S}(1^n, r^3), \mathcal{R} \rangle,$$

   where decom_i^3 corresponds to the opening of the bit $s_i \oplus g_i(r_i)$.

The output of each party

- $\mathcal{S}$: The sender $\mathcal{S}$ outputs $(\mathsf{decom}_1, \ldots, \mathsf{decom}_N)$, where

$$\mathsf{decom}_i = \left(\mathsf{decom}_i^1, \mathsf{decom}_i^2, \mathsf{decom}_i^3\right)$$

- $\mathcal{R}$: The receiver $\mathcal{R}$ outputs $\mathsf{com} = \left(\mathsf{com}^1, \mathsf{com}^2, \mathsf{com}^3\right)$.

The local-opening procedure $\mathcal{V}$

The verifier $\mathcal{V}$ has an index $i \in [N]$, a decommitment decom_i, and a commitment com.

1. Execute $\widehat{r_i^*} \leftarrow \widehat{\mathcal{V}}(i, \mathsf{decom}_i^1, \mathsf{com}^1)$.
2. Execute $\widehat{g_i^*} \leftarrow \widehat{\mathcal{V}}(i, \mathsf{decom}_i^2, \mathsf{com}^2)$.
3. Execute $\widehat{u_i^*} \leftarrow \widehat{\mathcal{V}}(i, \mathsf{decom}_i^3, \mathsf{com}^3)$.
4. If all executions succeed (output non-$\perp$), output $\widehat{g_i^*}(\widehat{r_i^*}) \oplus \widehat{u_i^*}$, where $\widehat{g_i^*}$ is treated as a description of a hash function in $\mathcal{G}$. Otherwise, output $\perp$.

The commit protocol between $\mathcal{S}$ and $\mathcal{R}$

The sender $\mathcal{S}$ has string $s = s_1 \dots s_{2^d}$ where $s_i \in \{0,1\}^n$ for all $i \in [2^d]$.

1. $\mathcal{R} \Rightarrow \mathcal{S}$: Samples $h \leftarrow \mathcal{H}$ and sends h.
2. $\mathcal{S} \Rightarrow \mathcal{R}$: Compute $s' = C(s)$ and a Merkle hash-tree T of s' using h and send the root-hash y.
3. $\mathcal{R} \Rightarrow \mathcal{S}$: Sample $g \leftarrow \mathcal{G}$ and send g.
4. $\mathcal{S} \Leftrightarrow \mathcal{R}$: Send $u = g(s)$ to the receiver

The output of the receive is $\mathsf{com} = (h, y, g, u)$**.**

The verifier $\mathcal{V}$ gets the input s simulates the sender to verify y and u.

Figure 6.4: Our four-round commitment protocol for strings of length $2^d \cdot n$.

Bibliography

[Aar06] Scott Aaronson. Lower bounds for local search by quantum arguments. *SIAM J. Comput.*, 35(4):804–824, 2006.

[AASY16] Benny Applebaum, Sergei Artemenko, Ronen Shaltiel, and Guang Yang. Incompressible functions, relative-error extractors, and the power of nondeterministic reductions. *Computational Complexity*, 25(2):349–418, 2016.

[AB09] Sanjeev Arora and Boaz Barak. *Computational Complexity - A Modern Approach*. Cambridge University Press, 2009.

[AB15] Benny Applebaum and Zvika Brakerski. Obfuscating circuits via composite-order graded encoding. In *Theory of Cryptography - 12th Theory of Cryptography Conference, TCC 2015, Warsaw, Poland, March 23-25, 2015, Proceedings, Part II*, pages 528–556, 2015.

[AGHP92] Noga Alon, Oded Goldreich, Johan Håstad, and René Peralta. Simple construction of almost k-wise independent random variables. *Random Struct. Algorithms*, 3(3):289–304, 1992.

[AIKS16] Sergei Artemenko, Russell Impagliazzo, Valentine Kabanets, and Ronen Shaltiel. Pseudorandomness when the odds are against you. In *31st Conference on Computational Complexity, CCC 2016, May 29 to June 1, 2016, Tokyo, Japan*, pages 9:1–9:35, 2016.

[AJN+16] Prabhanjan Ananth, Aayush Jain, Moni Naor, Amit Sahai, and Eylon Yogev. Universal constructions and robust combiners for indistinguishability obfuscation and witness encryption. In *CRYPTO*, pages 491–520, 2016.

[AKV04] Tim Abbott, Daniel Kane, and Paul Valiant. On algorithms for Nash equilibria. http://web.mit.edu/tabbott/Public/final.pdf, 2004. unpublished manuscript.

[Ald83] David Aldous. Minimization algorithms and random walk on the d-cube. *The Annals of Probability*, 11(2):403–413, 1983.

[AS08] Noga Alon and Joel Spencer. *The Probabilistic Method*. John Wiley, third edition, 2008.

[AS15] Gilad Asharov and Gil Segev. Limits on the power of indistinguishability obfuscation and functional encryption. In *IEEE 56th Annual Symposium on Foundations of Computer Science, FOCS 2015, Berkeley, CA, USA, 17-20 October, 2015*, pages 191–209, 2015.

[AS16] Gilad Asharov and Gil Segev. Limits on the power of indistinguishability obfuscation and functional encryption. *SIAM J. Comput.*, 45(6):2117–2176, 2016.

[Bab14] Yakov Babichenko. Query complexity of approximate Nash equilibria. In *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014*, pages 535–544, 2014.

[Bar01] Boaz Barak. How to go beyond the black-box simulation barrier. In *42nd Annual Symposium on Foundations of Computer Science, FOCS*, pages 106–115. IEEE Computer Society, 2001.

[BCC88] Gilles Brassard, David Chaum, and Claude Crépeau. Minimum disclosure proofs of knowledge. *J. Comput. Syst. Sci.*, 37(2):156–189, 1988.

[BCE+98] Paul Beame, Stephen A. Cook, Jeff Edmonds, Russell Impagliazzo, and Toniann Pitassi. The relative complexity of NP search problems. *J. Comput. Syst. Sci.*, 57(1):3–19, 1998.

[BCGL92] Shai Ben-David, Benny Chor, Oded Goldreich, and Michael Luby. On the theory of average case complexity. *J. Comput. Syst. Sci.*, 44(2):193–219, 1992.

[BDRV17] Itay Berman, Akshay Degwekar, Ron D. Rothblum, and Prashant Nalini Vasudevan. Multi collision resistant hash functions and their applications. *IACR Cryptology ePrint Archive*, 2017:489, 2017.

[BDT16] Avraham Ben-Aroya, Dean Doron, and Amnon Ta-Shma. Explicit two-source extractors for near-logarithmic min-entropy. *Electronic Colloquium on Computational Complexity (ECCC)*, 23:88, 2016.

[BDV16] Nir Bitansky, Akshay Degwekar, and Vinod Vaikuntanathan. Structure vs hardness through the obfuscation lens. *Electronic Colloquium on Computational Complexity (ECCC)*, 23:91, 2016.

[Ben89] Charles H. Bennett. Time/space trade-offs for reversible computation. *SIAM J. Comput.*, 18(4):766–776, 1989.

[BFK+10] Harry Buhrman, Lance Fortnow, Michal Koucký, John D. Rogers, and Nikolai K. Vereshchagin. Does the polynomial hierarchy collapse if onto functions are invertible? *Theory Comput. Syst.*, 46(1):143–156, 2010.

[BG94] Mihir Bellare and Shafi Goldwasser. The complexity of decision versus search. *SIAM J. Comput.*, 23(1):97–119, 1994.

[BG08] Boaz Barak and Oded Goldreich. Universal arguments and their applications. *SIAM J. Comput.*, 38(5):1661–1694, 2008.

[BGI+01] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. In *CRYPTO*, volume 2139 of *Lecture Notes in Computer Science*, pages 1–18. Springer, 2001.

[BGI+12a] 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, 2012. Preliminary version [BGI+01].

[BGI+12b] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. *J. ACM*, 59(2):6, 2012.

[BHKN13] Itay Berman, Iftach Haitner, Ilan Komargodski, and Moni Naor. Hardness preserving reductions via cuckoo hashing. In *Theory of Cryptography - 10th Theory of Cryptography Conference, TCC 2013, Tokyo, Japan, March 3-6, 2013. Proceedings*, pages 40–59, 2013.

[BK95] Manuel Blum and Sampath Kannan. Designing programs that check their work. *J. ACM*, 42(1):269–291, 1995.

[BKP17] Nir Bitansky, Yael Tauman Kalai, and Omer Paneth. Multi-collision resistance: A paradigm for keyless hash functions. *IACR Cryptology ePrint Archive*, 2017:488, 2017.

[Blo70] Burton H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Commun. ACM*, 13(7):422–426, 1970.

[BLV06] Boaz Barak, Yehuda Lindell, and Salil P. Vadhan. Lower bounds for non-black-box zero knowledge. *J. Comput. Syst. Sci.*, 72(2):321–391, 2006.

[BM04] Josh Buresh-Oppenheim and Tsuyoshi Morioka. Relativized NP search problems and propositional proof systems. In *19th Annual IEEE Conference on Computational Complexity (CCC 2004), 21-24 June 2004, Amherst, MA, USA*, pages 54–67, 2004.

[BOV07] Boaz Barak, Shien Jin Ong, and Salil P. Vadhan. Derandomization in cryptography. *SIAM J. Comput.*, 37(2):380–400, 2007.

[BPR15] Nir Bitansky, Omer Paneth, and Alon Rosen. On the cryptographic hardness of finding a nash equilibrium. In *FOCS*, pages 1480–1498. IEEE Computer Society, 2015.

[BR97] Mihir Bellare and Phillip Rogaway. Collision-resistant hashing: Towards making uowhfs practical. In *Advances in Cryptology - CRYPTO*, volume 1294, pages 470–484, 1997.

[Bra83] Gilles Brassard. Relativized cryptography. *IEEE Trans. Information Theory*, 29(6):877–893, 1983.

[Bus09] Sam Buss. Introduction to NP functions and local search, 2009. Slides: http://www.math.ucsd.edu/~sbuss/ResearchWeb/Prague2009/talkslides1.pdf. Accessed: 2017-04-01.

[BV15] Nir Bitansky and Vinod Vaikuntanathan. A note on perfect correctness by derandomization. *Electronic Colloquium on Computational Complexity (ECCC)*, 22:187, 2015.

[BVWW16] Zvika Brakerski, Vinod Vaikuntanathan, Hoeteck Wee, and Daniel Wichs. Obfuscating conjunctions under entropic ring LWE. In *Proceedings of the 2016 ACM Conference ITCS, Cambridge, MA, USA, January 14-16, 2016*, pages 147–156, 2016.

[CD08] Xi Chen and Xiaotie Deng. Matching algorithmic bounds for finding a Brouwer fixed point. *J. ACM*, 55(3), 2008.

[CD09] Xi Chen and Xiaotie Deng. On the complexity of 2D discrete fixed point problem. *Theor. Comput. Sci.*, 410(44):4448–4456, 2009.

[CDG01] Fan Chung, Persi Diaconis, and Ronald Graham. Combinatorics for the east model. *Advances in Applied Mathematics*, 27(1):192–206, 2001.

[CDT09] Xi Chen, Xiaotie Deng, and Shang-Hua Teng. Settling the complexity of computing two-player Nash equilibria. *J. ACM*, 56(3), 2009.

[CG88] Benny Chor and Oded Goldreich. Unbiased bits from sources of weak randomness and probabilistic communication complexity. *SIAM J. Comput.*, 17(2):230–261, 1988.

[CGH04] Ran Canetti, Oded Goldreich, and Shai Halevi. The random oracle methodology, revisited. *J. ACM*, 51(4):557–594, 2004.

[Coh16a] Gil Cohen. Two-source dispersers for polylogarithmic entropy and improved ramsey graphs. In Daniel Wichs and Yishay Mansour, editors, *STOC*, pages 278–284. ACM, 2016.

[Coh16b] Gil Cohen. Two-source extractors for quasi-logarithmic min-entropy and improved privacy amplification protocols. *Electronic Colloquium on Computational Complexity (ECCC)*, 23:114, 2016.

[Con92] Anne Condon. The complexity of stochastic games. *Inf. Comput.*, 96(2):203–224, 1992.

[Con08] David Conlon. A new upper bound for the bipartite ramsey problem. *Journal of Graph Theory*, 58(4):351–356, 2008.

[Cop85] Don Coppersmith. Another birthday attack. In *Advances in Cryptology - CRYPTO*, pages 14–17, 1985.

[CT07] Xi Chen and Shang-Hua Teng. Paths beyond local search: A tight bound for randomized fixed-point computation. In *48th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2007), October 20-23, 2007, Providence, RI, USA, Proceedings*, pages 124–134, 2007.

[CZ16] Eshan Chattopadhyay and David Zuckerman. Explicit two-source extractors and resilient functions. In *STOC*, pages 670–683. ACM, 2016.

[Dam89] Ivan Damgård. A design principle for hash functions. In *Advances in Cryptology - CRYPTO*, volume 435, pages 416–427, 1989.

[DGP09] Constantinos Daskalakis, Paul W. Goldberg, and Christos H. Papadimitriou. The complexity of computing a Nash equilibrium. *SIAM J. Comput.*, 39(1):195–259, 2009.

[DN07] Cynthia Dwork and Moni Naor. Zaps and their applications. *SIAM J. Comput.*, 36(6):1513–1543, 2007.

[DNR04] Cynthia Dwork, Moni Naor, and Omer Reingold. Immunizing encryption schemes from decryption errors. In *Advances in Cryptology - EUROCRYPT 2004, International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004, Proceedings*, pages 342–360, 2004.

[DP11] Constantinos Daskalakis and Christos H. Papadimitriou. Continuous local search. In *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 790–804, 2011.

[DPP97] Ivan Damgård, Torben P. Pedersen, and Birgit Pfitzmann. On the existence of statistically hiding bit commitment schemes and fail-stop signatures. *J. Cryptology*, 10(3):163–194, 1997.

[DPP98] Ivan Damgård, Torben P. Pedersen, and Birgit Pfitzmann. Statistical secrecy and multibit commitments. *IEEE Trans. Information Theory*, 44(3):1143–1151, 1998.

[DS11] Yevgeniy Dodis and John P. Steinberger. Domain extension for macs beyond the birthday barrier. In Kenneth G. Paterson, editor, *Advances in Cryptology - EUROCRYPT*, volume 6632 of *Lecture Notes in Computer Science*, pages 323–342. Springer, 2011.

[ER60] Paul Erdös and Richard Rado. Intersection theorems for systems of sets. *Journal of London Mathematical Society*, 35:85–90, 1960.

[Erd47] Paul Erdös. Some remarks on the theory of graphs. *Bull. Amer. Math. Soc.*, 53(4):292–294, 04 1947.

[ESY84] Shimon Even, Alan L. Selman, and Yacov Yacobi. The complexity of promise problems with applications to public-key cryptography. *Information and Control*, 61(2):159–173, 1984.

[FGK+13] David Mandell Freeman, Oded Goldreich, Eike Kiltz, Alon Rosen, and Gil Segev. More constructions of lossy and correlation-secure trapdoor functions. *J. Cryptology*, 26(1):39–74, 2013.

[FISV09] Katalin Friedl, Gábor Ivanyos, Miklos Santha, and Yves F. Verhoeven. On the black-box complexity of Sperner's lemma. *Theory Comput. Syst.*, 45(3):629–646, 2009.

[FPT04] Alex Fabrikant, Christos H. Papadimitriou, and Kunal Talwar. The complexity of pure Nash equilibria. In *Proceedings of the 36th Annual ACM Symposium on Theory of Computing, Chicago, IL, USA, June 13-16, 2004*, pages 604–612, 2004.

[FS90] Uriel Feige and Adi Shamir. Witness indistinguishable and witness hiding protocols. In *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing, May 13-17, 1990, Baltimore, Maryland, USA*, pages 416–426, 1990.

[GCC88] Marc Girault, Robert Cohen, and Mireille Campana. A generalized birthday attack. In *Advances in Cryptology - EUROCRYPT*, pages 129–156, 1988.

[GGH+13a] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *FOCS*, pages 40–49, 2013.

[GGH+13b] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *FOCS*, 2013.

[GGKT05] Rosario Gennaro, Yael Gertner, Jonathan Katz, and Luca Trevisan. Bounds on the efficiency of generic cryptographic constructions. *SIAM J. Comput.*, 35(1):217–246, 2005.

[GI02] Venkatesan Guruswami and Piotr Indyk. Near-optimal linear-time codes for unique decoding and new list-decodable codes over smaller alphabets. In *Proceedings on 34th Annual ACM Symposium on Theory of Computing*, pages 812–821. ACM, 2002.

[GI03] Venkatesan Guruswami and Piotr Indyk. Linear time encodable and list decodable codes. In *Proceedings of the 35th Annual ACM Symposium on Theory of Computing*, pages 126–135. ACM, 2003.

[GI04] Venkatesan Guruswami and Piotr Indyk. Linear-time list decoding in error-free settings: (extended abstract). In *Automata, Languages and Programming: 31st International Colloquium, ICALP*, volume 3142 of *Lecture Notes in Computer Science*, pages 695–707, 2004.

[GK03] Shafi Goldwasser and Yael Tauman Kalai. On the (in)security of the fiat-shamir paradigm. In *FOCS*, pages 102–113. IEEE Computer Society, 2003.

[GLSW14] Craig Gentry, Allison B. Lewko, Amit Sahai, and Brent Waters. Indistinguishability obfuscation from the multilinear subgroup elimination assumption. *IACR Cryptology ePrint Archive*, 2014:309, 2014.

[Gol08] Oded Goldreich. *Computational complexity - a conceptual perspective*. Cambridge University Press, 2008.

[Gol11] Oded Goldreich. Introduction to testing graph properties. In *Studies in Complexity and Cryptography*, volume 6650, pages 470–506. 2011.

[GP17] Paul W. Goldberg and Christos H. Papadimitriou. Towards a unified complexity theory of total functions. *Electronic Colloquium on Computational Complexity (ECCC)*, 24:56, 2017.

[GPS16] Sanjam Garg, Omkant Pandey, and Akshayaram Srinivasan. Revisiting the cryptographic hardness of finding a nash equilibrium. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO*, volume 9815, pages 579–604, 2016.

[GPW15] Mika Göös, Toniann Pitassi, and Thomas Watson. Deterministic communication vs. partition number. In *FOCS*, pages 1077–1088. IEEE Computer Society, 2015.

[GR16] Paul W. Goldberg and Aaron Roth. Bounds for the query complexity of approximate equilibria. *ACM Trans. Economics and Comput.*, 4(4):24:1–24:25, 2016.

[GRS90] Ronald L. Graham, Bruce L. Rothschild, and Joel H. Spencer. *Ramsey Theory*. John Wiley, second edition, 1990.

[GS94] Marc Girault and Jacques Stern. On the length of cryptographic hash-values used in identification schemes. In *Advances in Cryptology - CRYPTO*, pages 202–215, 1994.

[GS99] Venkatesan Guruswami and Madhu Sudan. Improved decoding of Reed-Solomon and algebraic-geometry codes. *IEEE Trans. Information Theory*, 45(6):1757–1767, 1999.

[GSV99] Oded Goldreich, Amit Sahai, and Salil P. Vadhan. Can statistical zero knowledge be made non-interactive? or on the relationship of SZK and NISZK. In Michael J. Wiener, editor, *Advances in Cryptology - CRYPTO*, volume 1666 of *Lecture Notes in Computer Science*, pages 467–484. Springer, 1999.

[Gue09] Shay Gueron. *Intel's New AES Instructions for Enhanced Performance and Security*, pages 51–66. Springer Berlin Heidelberg, 2009.

[GUV09] Venkatesan Guruswami, Christopher Umans, and Salil P. Vadhan. Unbalanced expanders and randomness extractors from parvaresh-vardy codes. *J. ACM*, 56(4):20:1–20:34, 2009.

[Had00] Satoshi Hada. Zero-knowledge and code obfuscation. In *ASIACRYPT*, volume 1976, pages 443–457, 2000.

[HHK+09] Iftach Haitner, Omer Horvitz, Jonathan Katz, Chiu-Yuen Koo, Ruggero Morselli, and Ronen Shaltiel. Reducing complexity assumptions for statistically-hiding commitment. *J. Cryptology*, 22(3):283–310, 2009.

[HHR+10] Iftach Haitner, Thomas Holenstein, Omer Reingold, Salil P. Vadhan, and Hoeteck Wee. Universal one-way hash functions via inaccessible entropy. *IACR Cryptology ePrint Archive*, 2010:120, 2010.

[HHRS15] Iftach Haitner, Jonathan J. Hoch, Omer Reingold, and Gil Segev. Finding collisions in interactive protocols - tight lower bounds on the round and communication complexities of statistically hiding commitments. *SIAM J. Comput.*, 44(1):193–242, 2015.

[HILL99] Johan Håstad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. A pseudorandom generator from any one-way function. *SIAM J. Comput.*, 1999.

[HIOS15] Iftach Haitner, Yuval Ishai, Eran Omri, and Ronen Shaltiel. Parallel hashing via list recoverability. In *Advances in Cryptology - CRYPTO*, volume 9216 of *Lecture Notes in Computer Science*, pages 173–190. Springer, 2015.

[HNO+09] Iftach Haitner, Minh-Huyen Nguyen, Shien Jin Ong, Omer Reingold, and Salil P. Vadhan. Statistically hiding commitments and statistical zero-knowledge arguments from any one-way function. *SIAM J. Comput.*, 39(3):1153–1218, 2009.

[HNY17] Pavel Hubácek, Moni Naor, and Eylon Yogev. The journey from NP to TFNP hardness. In *8th Innovations in Theoretical Computer Science Conference, ITCS 2017, January 9-11, 2017, Berkeley, CA, USA*, pages 60:1–60:21, 2017.

[HO12] Brett Hemenway and Rafail Ostrovsky. Extended-ddh and lossy trapdoor functions. In *PKC*, volume 7293, pages 627–643. Springer, 2012.

[HPV89] Michael D. Hirsch, Christos H. Papadimitriou, and Stephen A. Vavasis. Exponential lower bounds for finding Brouwer fixed points. *J. Complexity*, 5(4):379–416, 1989.

[HR04] Chun-Yuan Hsiao and Leonid Reyzin. Finding collisions on a public road, or do secure hash functions need secret coins? In *Advances in Cryptology - CRYPTO*, pages 92–105, 2004.

[HSX17] Akinori Hosoyamada, Yu Sasaki, and Keita Xagawa. Quantum multicollision-finding algorithm. *IACR Cryptology ePrint Archive*, 2017:864, 2017.

[HW15] Brett Hemenway and Mary Wootters. Linear-time list recovery of high-rate expander codes. In *Automata, Languages, and Programming - 42nd International Colloquium, ICALP*, volume 9134 of *Lecture Notes in Computer Science*, pages 701–712. Springer, 2015.

[HY17] Pavel Hubácek and Eylon Yogev. Hardness of continuous local search: Query complexity and cryptographic lower bounds. In *SODA*, pages 1352–1371. SIAM, 2017.

[IL89] Russell Impagliazzo and Michael Luby. One-way functions are essential for complexity based cryptography (extended abstract). In *30th Annual Symposium on Foundations of Computer Science, FOCS*, pages 230–235. IEEE Computer Society, 1989.

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

[ILL89] Russell Impagliazzo, Leonid A. Levin, and Michael Luby. Pseudo-random generation from one-way functions (extended abstracts). In *Proceedings of the 21st Annual ACM Symposium on Theory of Computing*, pages 12–24. ACM, 1989.

[Imp95] Russell Impagliazzo. A personal view of average-case complexity. In *Proceedings of the Tenth Annual Structure in Complexity Theory Conference*, pages 134–147. IEEE Computer Society, 1995.

[IN88] Russell Impagliazzo and Moni Naor. Decision trees and downward closures. In *3rd Annual Structure in Complexity Theory Conference*, pages 29–38. IEEE Computer Society, 1988.

[IW97] Russell Impagliazzo and Avi Wigderson. $P = BPP$ if E requires exponential circuits: Derandomizing the XOR lemma. In *Proceedings of the Twenty-Ninth Annual ACM Symposium on the Theory of Computing, El Paso, Texas, USA, May 4-6, 1997*, pages 220–229, 1997.

[Jer09] Emil Jerábek. Approximate counting by hashing in bounded arithmetic. *J. Symb. Log.*, 74(3):829–860, 2009.

[Jer16] Emil Jerábek. Integer factoring and modular square roots. *J. Comput. Syst. Sci.*, 82(2):380–394, 2016.

[Jou04] Antoine Joux. Multicollisions in iterated hash functions. application to cascaded constructions. In *CRYPTO*, volume 3152, pages 306–316, 2004.

[JPY88] David S. Johnson, Christos H. Papadimitriou, and Mihalis Yannakakis. How easy is local search? *J. Comput. Syst. Sci.*, 37(1):79–100, 1988.

[Juk11] Stasys Jukna. *Extremal Combinatorics - With Applications in Computer Science*. Texts in Theoretical Computer Science. An EATCS Series. Springer, second edition, 2011.

[Kil92] Joe Kilian. A note on efficient zero-knowledge proofs and arguments (extended abstract). In *STOC*, pages 723–732. ACM, 1992.

[KK05] Jonathan Katz and Chiu-Yuen Koo. On constructing universal one-way hash functions from arbitrary one-way functions. *IACR Cryptology ePrint Archive*, 2005:328, 2005.

[KMN+14] Ilan Komargodski, Tal Moran, Moni Naor, Rafael Pass, Alon Rosen, and Eylon Yogev. One-way functions and (im)perfect obfuscation. In *55th IEEE Annual Symposium on Foundations of Computer Science, FOCS*, pages 374–383, 2014.

[KNY17] Ilan Komargodski, Moni Naor, and Eylon Yogev. White-box vs. black-box complexity of search problems: Ramsey and graph property testing. In *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS*, pages 622–632, 2017.

[KNY18] Ilan Komargodski, Moni Naor, and Eylon Yogev. Collision resistant hashing for paranoids: Dealing with multiple collisions. In *Advances in Cryptology - EUROCRYPT 2018 - 37th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tel Aviv, Israel, April 29 - May 3, 2018 Proceedings, Part II*, pages 162–194, 2018.

[KOS10] Eike Kiltz, Adam O'Neill, and Adam D. Smith. Instantiability of RSA-OAEP under chosen-plaintext attack. In *CRYPTO*, volume 6223, pages 295–313. Springer, 2010.

[Kra01] Jan Krajícek. On the weak pigeonhole principle. *Fundamenta Mathematicae*, 170:123–140, 2001.

[Kra05] Jan Krajícek. Structured pigeonhole principle, search problems and hard tautologies. *J. Symb. Log.*, 70(2):619–630, 2005.

[KS17a] Ilan Komargodski and Gil Segev. From minicrypt to obfustopia via private-key functional encryption. In *Advances in Cryptology - EUROCRYPT 2017 - 36th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Paris, France, April 30 - May 4, 2017, Proceedings, Part I*, pages 122–151, 2017.

[KS17b] Ilan Komargodski and Gil Segev. From minicrypt to obfustopia via private-key functional encryption. To appear in EUROCRYPT, 2017.

[KSS11] Jeff Kahn, Michael E. Saks, and Clifford D. Smyth. The dual BKR inequality and Rudich's conjecture. *Combinatorics, Probability & Computing*, 20(2):257–266, 2011.

[Lau83] Clemens Lautemann. BPP and the polynomial hierarchy. *Inf. Process. Lett.*, 17(4):215–217, 1983.

[Li16] Xin Li. Improved non-malleable extractors, non-malleable codes and independent source extractors. *Electronic Colloquium on Computational Complexity (ECCC)*, 23:115, 2016.

[LNNW95] László Lovász, Moni Naor, Ilan Newman, and Avi Wigderson. Search problems in the decision tree model. *SIAM J. Discrete Math.*, 8(1):119–132, 1995.

[LTT89] Donna Crystal Llewellyn, Craig Tovey, and Michael Trick. Local optimization on graphs. *Discrete Applied Mathematics*, 23(2):157–178, 1989.

[Mer89a] Ralph C. Merkle. A certified digital signature. In *CRYPTO*, volume 435, pages 218–238. Springer, 1989.

[Mer89b] Ralph C. Merkle. One way hash functions and DES. In Gilles Brassard, editor, *Advances in Cryptology - CRYPTO*, volume 435, pages 428–446, 1989.

[Mir06] Ilya Mironov. Collision-resistant no more: Hash-and-sign paradigm revisited. In *Public Key Cryptography - PKC*, pages 140–156, 2006.

[Mor01] Tsuyoshi Morioka. Classification of search problems and their definability in bounded arithmetic. *Electronic Colloquium on Computational Complexity (ECCC)*, (082), 2001.

[MP91] Nimrod Megiddo and Christos H. Papadimitriou. On total functions, existence theorems and computational complexity. *Theor. Comput. Sci.*, 81(2):317–324, 1991.

[MT07] Ueli M. Maurer and Stefano Tessaro. Domain extension of public random functions: Beyond the birthday barrier. In *Advances in Cryptology - CRYPTO*, volume 4622, pages 187–204. Springer, 2007.

[MX10] Mohammad Mahmoody and David Xiao. On the power of randomized reductions and the checkability of SAT. In *Proceedings of the 25th Annual IEEE Conference on Computational Complexity, CCC*, pages 64–75, 2010.

[Nao91] Moni Naor. Bit commitment using pseudorandomness. *Journal of Cryptology*, 1991.

[Nao92] Moni Naor. Constructing ramsey graphs from small probability spaces. *IBM Research Report RJ 8810, 1992*, 1992.

[Nas50] John F. Nash. Equilibrium points in *n*-person games. *Proceedings of the National Academy of Sciences of the United States of America*, 36:48–49, 1950.

[NN93] Joseph Naor and Moni Naor. Small-bias probability spaces: Efficient constructions and applications. *SIAM J. Comput.*, 22(4):838–856, 1993.

[NOVY98] Moni Naor, Rafail Ostrovsky, Ramarathnam Venkatesan, and Moti Yung. Perfect zero-knowledge arguments for *NP* using any one-way permutation. *J. Cryptology*, 11(2):87–108, 1998.

[NPR12] Hung Q. Ngo, Ely Porat, and Atri Rudra. Efficiently decodable compressed sensing by list-recoverable codes and recursion. In *29th International Symposium on Theoretical Aspects of Computer Science, STACS*, volume 14 of *LIPIcs*, pages 230–241. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2012.

[NSW08] Moni Naor, Gil Segev, and Udi Wieder. History-independent cuckoo hashing. In *Automata, Languages and Programming, 35th International Colloquium, ICALP 2008, Reykjavik, Iceland, July 7-11, 2008, Proceedings, Part II - Track B: Logic, Semantics, and Theory of Programming & Track C: Security and Cryptography Foundations*, pages 631–642, 2008.

[NW94] Noam Nisan and Avi Wigderson. Hardness vs randomness. *J. Comput. Syst. Sci.*, 49(2):149–167, 1994.

[NY89] Moni Naor and Moti Yung. Universal one-way hash functions and their cryptographic applications. In *Proceedings of the 21st Annual ACM Symposium on Theory of Computing*, pages 33–43. ACM, 1989.

[NY15a] Moni Naor and Eylon Yogev. Bloom filters in adversarial environments. In *Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part II*, pages 565–584, 2015.

[NY15b] Moni Naor and Eylon Yogev. Tight bounds for sliding bloom filters. *Algorithmica*, 73(4):652–672, 2015.

[Pap94] Christos H. Papadimitriou. On the complexity of the parity argument and other inefficient proofs of existence. *J. Comput. Syst. Sci.*, 48(3):498–532, 1994.

[PS96] David Pointcheval and Jacques Stern. Security proofs for signature schemes. In *EUROCRYPT*, volume 1070, pages 387–398. Springer, 1996.

[PST14] Rafael Pass, Karn Seth, and Sidharth Telang. Indistinguishability obfuscation from semantically-secure multilinear encodings. In *CRYPTO (1)*, volume 8616 of *Lecture Notes in Computer Science*, pages 500–517. Springer, 2014.

[Pud90] Pavel Pudlák. Ramsey's theorem in bounded arithmetic. In *Computer Science Logic, 4th Workshop, CSL*, pages 308–317, 1990.

[PW11] Chris Peikert and Brent Waters. Lossy trapdoor functions and their applications. *SIAM J. Comput.*, 40(6):1803–1844, 2011.

[RM99] Ran Raz and Pierre McKenzie. Separation of the monotone NC hierarchy. *Combinatorica*, 19(3):403–435, 1999.

[Rom90] John Rompel. One-way functions are necessary and sufficient for secure signatures. In *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing*, pages 387–394. ACM, 1990.

[Ros73] Robert W. Rosenthal. A class of games possessing pure-strategy Nash equilibria. *International Journal of Game Theory*, 2(1):65–67, 1973.

[RSS16] Alon Rosen, Gil Segev, and Ido Shahaf. Can PPAD hardness be based on standard cryptographic assumptions? *Electronic Colloquium on Computational Complexity (ECCC)*, 23:59, 2016.

[Rub15] Aviad Rubinstein. Inapproximability of Nash equilibrium. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015, Portland, OR, USA, June 14-17, 2015*, pages 409–418, 2015.

[Rud89] Steven Rudich. *Limits on the Provable Consequences of One-way Functions*. PhD thesis, University of California at Berkeley, 1989.

[SBK+17] Marc Stevens, Elie Bursztein, Pierre Karpman, Ange Albertini, and Yarik Markov. The first collision for full SHA-1. In *Advances in Cryptology - CRYPTO*, volume 10401, pages 570–596, 2017.

[Sha53] Lloyd S Shapley. Stochastic games. *Proceedings of the National Academy of Sciences*, 39(10):1095–1100, 1953.

[She11] Alexander A. Sherstov. The pattern matrix method. *SIAM J. Comput.*, 40(6):1969–2000, 2011.

[Sho00] Victor Shoup. A composition theorem for universal one-way hash functions. In *Advances in Cryptology - EUROCRYPT*, volume 1807, pages 445–452, 2000.

[Sim98] Daniel R. Simon. Finding collisions on a one-way street: Can secure hash functions be based on general assumptions? In *EUROCRYPT*, volume 1403, pages 334–345. Springer, 1998.

[SS06] Rahul Savani and Bernhard Stengel. Hard-to-solve bimatrix games. *Econometrica*, 74(2):397–429, 2006.

[SS09] Miklos Santha and Mario Szegedy. Quantum and classical query complexities of local search are polynomially related. *Algorithmica*, 55(3):557–575, 2009.

[SW14a] Amit Sahai and Brent Waters. How to use indistinguishability obfuscation: deniable encryption, and more. In *STOC*, pages 475–484. ACM, 2014.

[SW14b] Amit Sahai and Brent Waters. How to use indistinguishability obfuscation: deniable encryption, and more. In *STOC*, 2014.

[SY91] Alejandro A. Schäffer and Mihalis Yannakakis. Simple local search problems that are hard to solve. *SIAM J. Comput.*, 20(1):56–87, 1991.

[SY09] Xiaoming Sun and Andrew Chi-Chih Yao. On the quantum query complexity of local search in two and three dimensions. *Algorithmica*, 55(3):576–600, 2009.

[Ta-17] Amnon Ta-Shma. Explicit, almost optimal, epsilon-balanced codes. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC*, pages 238–251, 2017.

[WC81] Mark N. Wegman and Larry Carter. New hash functions and their use in authentication and set equality. *J. Comput. Syst. Sci.*, 22(3):265–279, 1981.

[Wee07] Hoeteck Wee. One-way permutations, interactive hashing and statistically hiding commitments. In *Theory of Cryptography, 4th Theory of Cryptography Conference, TCC*, pages 419–433, 2007.

[WS11] David P. Williamson and David B. Shmoys. *The design of approximation algorithms*. Cambridge university press, 2011.

[WYY05] Xiaoyun Wang, Yiqun Lisa Yin, and Hongbo Yu. Finding collisions in the full SHA-1. In *Advances in Cryptology - CRYPTO*, pages 17–36, 2005.

[Zha09] Shengyu Zhang. Tight bounds for randomized and quantum local search. *SIAM J. Comput.*, 39(3):948–977, 2009.