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

Dedicated to Oded Goldreich on the occasion of his sixtieth birthday

Ilan Komargodski Moni Naor Eylon Yogev

February 4, 2017

Abstract

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 with a guaranteed solution.

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.

[*]Department of Computer Science and Applied Mathematics, Weizmann Institute of Science Israel, Rehovot 76100, Israel. Emails: {ilan.komargodski,moni.naor,eylon.yogev}@weizmann.ac.il. Supported in part by grants from the Israel Science Foundation. Ilan Komargodski is supported in part by a Levzion Fellowship. Moni Naor is the incumbent of the Judith Kleeman Professorial Chair.

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 has no clique or independent set of size $4n$ with high probability (see Section 2.2).

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.

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 2.2 for the precise statement). The question is: given a circuit that represents such a colored graph, what is the computational complexity of

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

[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.

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.

Finally, we consider the bipartite version of the Ramsey problem and prove similar hardness results in the white-box setting. Specifically, we show a hardness result for finding a bi-clique or bi-independent set in a bipartite graph based on the assumption that multi-collision resistant hash functions exist. These are hash functions for which it is hard to find *multiple* inputs that hash to the same output.[4] To complement this result, we show the other direction: the hardness of the bipartite Ramsey problem *implies* the existence of multi-collision resistant hash functions.

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 4). In this model, the function is represented succinctly but the algorithm is unbounded and has only black-box access to the function.

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 1 we give a short summary of the above results.

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

[5] Due to this, 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 (for related problems).

[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.

Model	$\mathcal{R}$ Problem	**Ramsey-like**	**TFNP**
Black-Box (BB)	Hard (Sec. 5)	Hard [IN88]	Hard [HPV89]
White-Box	Easy (Sec. 5)	Hard (Sec. 3)	Hard [Pap94, HNY16]
Succinct Black-Box	Easy (Sec. 6)	Easy (Sec. 6)	Easy (Sec. 6)

Table 1: A summary of our results on the complexity of search problems. $\mathcal{R}$ is the problem defined in Section 5. The term "Ramsey-like" problems refers to the problems we consider in Section 3, 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.

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 8). 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 Appendix A 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.[7]

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 [HNY16] 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+12], so we have to work with weaker primitives such as indistinguishability obfuscation [BGI+12, GGH+13, SW14]. 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, KS17], 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.

2 Preliminaries

Unless stated otherwise, the logarithms in this paper are base 2. For a distribution $\mathcal{D}$ we denote by $x \leftarrow \mathcal{D}$ an element chosen from $\mathcal{D}$ uniformly at random. For an integer $n \in \mathbb{N}$ we denote by $[n]$ the set $\{1, \ldots, n\}$.

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 $n > N_c$. Two sequences of random variables $X = \{X_n\}_{n\in\mathbb{N}}$ and $Y = \{Y_n\}_{n\in\mathbb{N}}$ are *computationally indistinguishable* if for any probabilistic polynomial-time algorithm A, there exists a negligible function $\mathsf{negl}(\cdot)$ such that $|\Pr[A(1^n, X_n) = 1] - \Pr[A(1^n, Y_n) = 1]| \leq \mathsf{negl}(n)$ for all $n \in \mathbb{N}$.

[7] We thank Pavel Hubácek for telling us about [Kra01].

2.1 Search problems in the black-box and white-box models

Let $\mathcal{F}_n = \{f : \{0,1\}^n \to \{0,1\}^n\}$ be the class of all circuits f mapping n bits into n bits. We give a definition of a search problem for the family $\mathcal{F}_n$.[8]

Definition 1. *A* search problem $\mathcal{S}$ *is a relation on* $2q(n)$ *tuples. More precisely,* $\mathcal{S} = \cup_{n=1}^{\infty} \mathcal{S}_n$*, where* $\mathcal{S}_n \subseteq (\{0,1\}^n)^{q(n)} \times (\{0,1\}^n)^{q(n)}$ *for a polynomial* $q(\cdot)$*, such that: (i) for all* $f \in \mathcal{F}_n$*, there exist* $x_1, \dots, x_{q(n)} \in \{0,1\}^n$ *for which* $\big(x_1, \dots, x_{q(n)}, f(x_1), \dots, f(x_{q(n)})\big) \in \mathcal{S}$*, and (ii)* $\mathcal{S}$ *is computable in polynomial time in* n*. The class of all such search problems is denoted* TFNP*.*

The tuple $(x_1, \dots, x_{q(n)})$ is called the witness (i.e., the solution). In general, a witness is not necessarily given as a sequence of points in the domain $\{0,1\}^n$ 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 $\mathcal{S} \subseteq (\{0,1\}^n)^q \times (\{0,1\}^n)^q$.

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 2 (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* $x_1, \dots, x_q$ *such that* $(x_1, \dots, x_q, f(x_1), \dots, f(x_q)) \in \mathcal{S}$*.*

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 3 (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* $x_1, \dots, x_q$ *such that* $(x_1, \dots, x_q, f(x_1), \dots, f(x_q)) \in \mathcal{S}$*.*

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).

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

Definition 4 (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 below we summarize the differences between the models.

Model	**Function Access**	**Solver Running Time**	**Representation 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: A summary of the different models defined.

2.2 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 5 (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 1. *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,*

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

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

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

where the inequality follows by the inequality $\binom{2k}{k} \le 2^{2k}$. As a corollary of Proposition 1, we get:

Proposition 2. *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[9] or a small-bias probability space [NN93]). For completeness, the proof of the next statement is given in Appendix B.2. No explicit construction of graphs matching these parameters is known, but see Section 3.2 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 3. *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 $\{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ 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 6 (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 4. *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,*

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

Based on Proposition 4, 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 5. *For every $k > 2$ and $m > 1$, it holds that $R(\underbrace{k, \ldots, k}_{m \text{ times}}) \le m^{mk}$.*

As a corollary of Proposition 5, we obtain a bound on the number of colors that ensure the existence of a monochromatic subgraph of size k.

[9] 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 $y_1, y_2, \ldots, y_k \in \mathcal{R}$.

Proposition 6. *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 5 and 6 appear in Appendix B.1.

2.3 Randomness extractors

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

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

We say that a function $\mathsf{Ext}\colon \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ 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].[10]

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 7. *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.

2.4 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 $\mathcal{H}_n = \{h\colon \{0,1\}^{\ell_1(n)} \to \{0,1\}^{\ell_2(n)}\}$, where ℓ_1 and ℓ_2 are two functions such that $\ell_1(n) > \ell_2(n)$ for every $n \in \mathbb{N}$, is $\mathsf{collision\ resistant}$ if for every probabilistic polynomial-time algorithms A, there exists a negligible function $\mathsf{negl}(\cdot)$ such that

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

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

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[11], 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 7 ([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 $b \in \{0,1\}$, outputs a string $s \in \{0,1\}^{p(n)}$ for some fixed polynomial $p(\cdot)$. If $b = 0$, then the algorithm $F(s, \cdot)$ computes an injective function $f_s(\cdot)$ over $\{0,1\}^n$, and if $b = 1$, then the algorithm $F(s, \cdot)$ computes a function $f_s(\cdot)$ over $\{0,1\}^n$ whose image size is at most $2^{n-\ell}$.*

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.

3 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 $C\colon \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ that represents the adjacency matrix of a graph on 2^n vertices.

[11]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.

In terms of Definition 1, we have the $q(n) = \binom{n/2}{2}$ and the relation $\mathcal{S}$ is such that $\big(x_1, \ldots, x_{q(n)}, f(x_1), \ldots, f(x_{q(n)})\big) \in \mathcal{S}$ 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 $b \in \{0,1\}$.[12]

Hardness of the Ramsey problem. We say that the Ramsey problem is *hard* if there exists an efficiently samplable distribution $\mathcal{D} = \{C\colon \{0,1\}^n \times \{0,1\}^n \to \{0,1\}\}$ 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 2 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 2.4), 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 $\mathcal{H} = \{h\colon \{0,1\}^n \to \{0,1\}^{n/4}\}$.

Theorem 1. *The Ramsey problem is hard assuming the existence of collision resistant hash functions.*

In the proof of Theorem 1 we use a construction of Ramsey graphs given in Proposition 3 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 $h\colon \{0,1\}^{n+\ell} \to \{0,1\}^n$, 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 8 (The graph-hash product). *Given a graph $G = (V, E)$, where $|V| = \{0,1\}^n$, and a hash function $h\colon \{0,1\}^{n+\ell} \to \{0,1\}^n$, define the graph $G \otimes h = (V', E')$ as a graph on vertices $V' = \{0,1\}^{n+\ell}$ 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 1.

Proof of Theorem 1. 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 $\mathcal{G} = \{g\colon \{0,1\}^k \times \{0,1\}^k \to \{0,1\}\}$ 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 3, 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 $g \leftarrow \mathcal{G}$.

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

2. Set $G = (V, E)$ to be the graph with $|V| = 2^k$ vertices induced by g (see Proposition 3).
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 $\mathsf{Win}(A, g, h)$. That is,

$$\Pr[\mathsf{Win}(A, g, h)] \geq \epsilon.$$

Let $(v_1, \ldots, v_{2k})$ the solution found by A, and let $v'_i \triangleq h(v_i)$ for $i \in [2k]$. Let $\mathsf{Distinct}$ 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

$$\begin{aligned}\Pr[\mathsf{Win}(A, g, h)] = &\Pr[\mathsf{Win}(A, g, h) \mid \mathsf{Distinct}] \cdot \Pr[\mathsf{Distinct}] + \\ &\Pr[\mathsf{Win}(A, g, h) \mid \neg\mathsf{Distinct}] \cdot \Pr[\neg\mathsf{Distinct}] \geq \epsilon \end{aligned} \quad (1)$$

We first argue that

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

Indeed, by the definition of the event $\mathsf{Distinct}$, 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 3 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 (1), we get that

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

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 1 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 1).

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].

3.1 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 6 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 2. *The colorful Ramsey problem is hard assuming the existence of a collision resistant hash function family.*

In the proof of Theorem 2, we use an explicit construction of a colorful Ramsey coloring.

Lemma 1. *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 $K_1, \dots, K_k$ 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.

Color(u, v) :

```
1. Let $(u_1, \dots, u_{m/2})$ and $(v_1, \dots, v_{m/2})$ 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 2.

Proof of Theorem 2. We adapt the graph-hash product operation (see Definition 8) 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 $h \colon \{0,1\}^{n+\ell} \to \{0,1\}^n$, we define a coloring of the full graph on $2^{n+\ell}$ vertices $\psi \otimes h$ as follows. For any edge $e = (u, v) \in \{0,1\}^{n+\ell} \times \{0,1\}^{n+\ell}$, 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 1. Sample a collision resistant hash function $h \leftarrow \mathcal{H}$, where $\mathcal{H} = \{h \colon \{0,1\}^n \to \{0,1\}^{n'}\}$ 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 $(i_1, v_1), (i_2, v_2), (i_3, v_3) \in \{0,1\}^{n/2} \times \{0,1\}^{n/2}$ the solution found by A, and let $v_i' = h(i, v_i)$ for $i \in [3]$. We first observe that by Lemma 1 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. $\square$

3.2 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 9 (PWPP^n_k complete problem)**.** *Given a circuit $C\colon \{0,1\}^n \to \{0,1\}^k$, find $x \neq y$ such that $C(x) = C(y)$. Moreover, we define $\mathsf{PWPP} = \mathsf{PWPP}^n_{n-1}$.*

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 1 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]).[13] We will use the following theorem.

Proposition 8. *There exists an explicit k-Ramsey graph on N vertices, where $k \leq 2^{(\log\log N)^2}$.*[14]

Applying the same proof as that of Theorem 1, but using the explicit construction of Ramsey graphs above, results with:

Theorem 3. *Fix a family of collision resistant hash functions $\mathcal{H} = \{h\colon \{0,1\}^n \to \{0,1\}^{2^{\sqrt{\log(n/2)}}}\}$. There exists a* deterministic *reduction from RAMSEY to breaking the collision resistance of $\mathcal{H}$.*

As a corollary of Theorem 3 we obtain that the class defined by the Ramsey problem (i.e., the class RAMSEY) includes the class $\mathsf{PWPP}^n_{2^{\sqrt{\log(n/2)}}}$.

Corollary 1. *$\mathsf{PWPP}^n_{2^{\sqrt{\log(n/2)}}} \subseteq \mathsf{RAMSEY}$.*

Proof of Theorem 3. 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 8 and execute the Ramsey problem solver

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

[14]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 8) will directly imply the result in Theorem 3 based on a weaker assumption on the hash function family.

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 8 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 2 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 2. *PWPP $\subseteq$ COLORFUL-RAMSEY.*

3.3 The Ramsey problem and Multi-CRH

In Theorem 1 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.[15]

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]).[16] The result of Erdös [Erd47] on the abundance of $(2\log N)$-Ramsey graphs (see Proposition 3 and Appendix B.2) holds as is for bipartite graphs.

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 $\mathcal{D} = \{C\colon \{0,1\}^n \times \{0,1\}^n \to \{0,1\}\}$ 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

$$\Pr_{C \leftarrow \mathcal{D}}[u_1, \ldots, u_{n/4}, v_1, \ldots, v_{n/4} \leftarrow A(1^n, C)\ ;\ \exists b \in \{0,1\}, \forall i,j \in [n/4]\colon C(u_i, v_j) = b] \le \mathsf{negl}(n),$$

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 $\mathcal{H}_n = \{h\colon \{0,1\}^{\ell_1(n)} \to \{0,1\}^{\ell_2(n)}\}$, where ℓ_1 and ℓ_2 are two functions such that $\ell_1(n) > \ell_2(n)$ for

[15] 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).

[16] 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.

every $n \in \mathbb{N}$, is *k-multi-collision resistant* if for every probabilistic polynomial-time algorithms A, it holds that

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

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 4. *If the bipartite Ramsey problem is hard, then there exists a family $\mathcal{H} = \{h \colon \{0,1\}^n \to \{0,1\}^{n/2}\}$ of $n/4$-MCRH functions.*

Furthermore, if there exists a family $\mathcal{H} = \{h \colon \{0,1\}^n \to \{0,1\}^{\sqrt{n}/8}\}$ 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 $\mathcal{D} = \{C \colon \{0,1\}^n \times \{0,1\}^n \to \{0,1\}\}$ 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 $v_1, \dots, v_{n/2} \in \{0,1\}^n$ to be arbitrary $n/2$ distinct vertices on the right side. We define the hash function $h_C \colon \{0,1\}^n \to \{0,1\}^{n/2}$ to be the concatenation of the bits $C(x, v_1), \dots, C(x, v_{n/2})$.

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

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 $x_1, \dots, x_{n/4}$ be a $(n/4)$-multi-collision in h_c and denote by $y = h_C(x_1) \in \{0,1\}^{n/2}$. Without loss of generality, assume that the string y has more 1's than 0's and denote by $I = \{i_1, \dots, i_{n/4}\} \subseteq [n/2]$ a set of size $n/4$ of (distinct) indices for which $y_i = 1$. The collection of vertices $x_1, \dots, x_{n/4}$ on the left side and the vertices $\{v_{i_1}, \dots, v_{i_{n/4}}\}$ 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 1. First, following the proof of Proposition 3, 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 1. 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 $u_1, \dots, u_{n/4}, v_1, \dots, v_{n/4}$, 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].

4 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 10 (White-box property testing)**.** *An ϵ-tester for a graph property Π is a probabilistic machine that, on input a Boolean circuit $C\colon \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ 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)-$\mathsf{balanced}$ 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 2.4.

Theorem 5. *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 $b \in \{0,1\}$, executes the algorithm $s \leftarrow \mathsf{Gen}(1^n, b)$ and samples a n-wise-independent hash-function $h\colon \{0,1\}^{p(n)} \to \{0,1\}^n$. It outputs $s' = (s,h)$.

2. The algorithm Eval', on input $s' = (s,h)$ and $x \in \{0,1\}^n$, outputs $h(\mathsf{Eval}(s,x))$.

Claim 1. *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* $\{0,1\}^n$ *such that with very high probability for any* $y \in \{0,1\}^n$ *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* $\{0,1\}^n$ *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 $y \in \{0,1\}^n$ and all possible n tuples we get that the total probability that there exists a bin with more than n elements is at most $2^n \cdot \binom{2^n}{n} \cdot 2^{-n^2} \leq \exp(-n)$. $\square$

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 7, there exists a polynomial-size Boolean circuit $B_{n,k,\delta}$ that acts on inputs from $\{0,1\}^n \times \{0,1\}^n$ 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 1. *Better constructions of such explicit circuits (cf., Proposition 7) 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 $\{0,1\}^n \times \{0,1\}^n$. 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 $b \leftarrow \{0,1\}$ chosen uniformly at random. Then, we output the following circuit $C_{s'}$ that represents a graph G': On input a pair of vertices $u \in U = \{0,1\}^n, v \in V = \{0,1\}^n$, 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 1 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 $k' = 1.01k \le 0.52n$. 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 1, (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 $|A_i| \ge 2^k \cdot i$ and similarly by I_R the set of indices j for which $|B_j| \ge 2^k \cdot j$. This implies that

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

where the inequality $\sum_{i \notin I_L \text{ or } j \notin I_R} |A_i| \cdot |B_j| \le n^2 \cdot 2^{k'} \cdot 2^k \cdot n$ follows since $|I_L|, |I_R| \le n$ (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 $i, j \le n$, in A_i (resp. B_j) there must exist at least $|A_i|/i \ge 2^k$ (resp $|B_j|/j \ge 2^k$) 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 (2), the total number of edges in H is at least

$$\begin{aligned}\sum_{i \in I_L, j \in I_R} |A_i| \cdot |B_j| \cdot (1/2 - \delta) &\ge (1/2 - \delta) \cdot \sum_{i \in I_L, j \in I_R} |A_i| \cdot |B_j| \\ &\ge (1/2 - \delta) \cdot 2^{2k'} - (1/2 - \delta) \cdot 2^{k'} \cdot 2^k \cdot n^3 \\ &\ge (1/2 - \delta) \cdot 2^{2k'} - 2^{k' + k + 3\log n} \\ &= (1/2 - \delta - 2^{-k' + k + 3\log n}) \cdot 2^{2k'}. \end{aligned}$$

Since $k' = 1.01k$ and $k = 0.51n$, we have that $2^{-k'+k+3\log n} = 2^{-0.01k + 3\log n} \le 2^{-n/1000}$ for large enough n. Thus, letting $\delta' \triangleq \delta + 2^{-n/1000} \le 2^{-n/2000}$, the number of edges in H is at least

$$\begin{aligned}\sum_{i \in I_L, j \in I_R} |A_i| \cdot |B_j| \cdot (1/2 - \delta) &\ge (1/2 - \delta - 2^{-n/1000}) \cdot 2^{2k'} \\ &= (1/2 - \delta') \cdot 2^{2k'}. \end{aligned}$$

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 $A_1, \dots, A_{2^{n/3}} \subseteq U$. 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 $B_1, \dots, B_{2^{n/3}} \subseteq V$ 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 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 $f\colon \{0,1\}^n \to \{0,1\}^n$ the relation is:[17]

$$\mathcal{R}_{GK}^{f} = \{(f, C) \mid a = \mathsf{COM}^f(\tilde{C}) \ \wedge \ C(a) = f(a) \ \wedge \ |\tilde{C}| \leq 2^{n/10}\},$$

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

[17]The bound on the size of $\tilde{C}$ in [GK03] can be any super-polynomial function.

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:

$$\mathcal{R}^f_{\mathsf{PPP}} = \{(a, b) \in \{0,1\}^n \times \{0,1\}^n \mid a \neq b \ \wedge \ f(a) = f(b) \vee f(a) = 0\}.$$

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:

$$\mathcal{R}^f = \mathcal{R}^f_{GK} \cup \mathcal{R}^f_{\mathsf{PPP}}.$$

6 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 6. *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 6. 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:

2.1. Define $f^*: \{0,1\}^n \to \{0,1\}^n$ such that $f^*(x) = \mathsf{MostFrequent}_{f \in L} f(x)$.

2.2. Find a solution $x_1, \ldots, x_k$ relative to f^*.

2.3. Query $x_1, \ldots, x_k$.

2.4. If all query results are consistent with f^*, then output $x_1, \ldots, x_k$.

2.5. 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 6 as a generalization of that result.[18]

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 Section 2 and Table 2). 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 while replacing the use of a random oracle with a pseudorandom function[19].

[18]We thank Aviad Rubinstein for referring us to [GR16].

[19]A pseudorandom function is a (keyed) function that cannot be distinguished from a random function by any polynomially bounded adversary

7 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.

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, KS17]. 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?

Acknowledgments

We thank Gil Segev for fruitful discussions concerning the question of minimal assumptions leading to the hardness of Ramsey.

References

[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.

[AS08] Noga Alon and Joel Spencer. *The Probabilistic Method*. John Wiley, third edition, 2008.

[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.

[BDRV17] Itay Berman, Akshay Degwekar, Ron D. Rothblum, and Prashant Nalini Vasudevan. Multi collision resistant hash functions and their applications. *IACR Cryptology ePrint Archive*, page 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.

[BGI$^+$12] 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.

[BKP17] Nir Bitansky, Yael Tauman Kalai, and Omer Paneth. Multi-collision resistance: A paradigm for keyless hash functions. *IACR Cryptology ePrint Archive*, page 488, 2017.

[BPR15] Nir Bitansky, Omer Paneth, and Alon Rosen. On the cryptographic hardness of finding a Nash equilibrium. In *FOCS*, pages 1480–1498, 2015.

[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.

[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.

[Con08] David Conlon. A new upper bound for the bipartite ramsey problem. *Journal of Graph Theory*, 58(4):351–356, 2008.

[CZ16] Eshan Chattopadhyay and David Zuckerman. Explicit two-source extractors and resilient functions. In *STOC*, pages 670–683. ACM, 2016.

[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.

[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.

[GGH$^+$13] 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.

[GK03] Shafi Goldwasser and Yael Tauman Kalai. On the (in)security of the Fiat-Shamir paradigm. In *FOCS*, pages 102–113, 2003.

[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 *CRYPTO*, pages 579–604, 2016.

[GPW15] Mika Göös, Toniann Pitassi, and Thomas Watson. Deterministic communication vs. partition number. In *FOCS*, pages 1077–1088, 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.

[Had00] Satoshi Hada. Zero-knowledge and code obfuscation. In *ASIACRYPT*, pages 443–457, 2000.

[HNY16] Pavel Hubácek, Moni Naor, and Eylon Yogev. The journey from NP to TFNP hardness. *ECCC*, 23:199, 2016. To appear in ITCS 2017.

[HO12] Brett Hemenway and Rafail Ostrovsky. Extended-DDH and lossy trapdoor functions. In *PKC*, pages 627–643, 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.

[HY17] Pavel Hubácek and Eylon Yogev. Hardness of continuous local search: Query complexity and cryptographic lower bounds. In *SODA*, pages 1352–1371, 2017.

[IN88] Russell Impagliazzo and Moni Naor. Decision trees and downward closures. In *3rd Annual Structure in Complexity Theory Conference*, pages 29–38, 1988.

[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.

[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.

[KNY17] Ilan Komargodski, Moni Naor, and Eylon Yogev. Collision resistant hashing for paranoids: Dealing with multiple collisions. *IACR Cryptology ePrint Archive*, page 486, 2017.

[KOS10] Eike Kiltz, Adam O'Neill, and Adam D. Smith. Instantiability of RSA-OAEP under chosen-plaintext attack. In *CRYPTO*, pages 295–313. Springer, 2010.

[Kra01] Jan Krajícek. On the weak pigeonhole principle. *Fundamenta Mathematicae*, 170:123–140, 2001.

[KS17] Ilan Komargodski and Gil Segev. From Minicrypt to Obfustopia via private-key functional encryption. In *EUROCRYPT*, pages 122–151, 2017.

[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.

[MP91] Nimrod Megiddo and Christos H. Papadimitriou. On total functions, existence theorems and computational complexity. *Theor. Comput. Sci.*, 81(2):317–324, 1991.

[Nao92] Moni Naor. Constructing ramsey graphs from small probability spaces. *IBM Research Report RJ 8810*, 1992. Available at: http://www.wisdom.weizmann.ac.il/~naor/PAPERS/ramsey.ps. Accessed: 2017-08-01.

[NN93] Joseph Naor and Moni Naor. Small-bias probability spaces: Efficient constructions and applications. *SIAM J. Comput.*, 22(4):838–856, 1993.

[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*, pages 387–398, 1996.

[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.

[She11] Alexander A. Sherstov. The pattern matrix method. *SIAM J. Comput.*, 40(6):1969–2000, 2011.

[Sim98] Daniel R. Simon. Finding collisions on a one-way street: Can secure hash functions be based on general assumptions? In *EUROCRYPT*, pages 334–345, 1998.

[SW14] Amit Sahai and Brent Waters. How to use indistinguishability obfuscation: deniable encryption, and more. In *STOC*, 2014.

A 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 11 (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 2 (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 $C\colon \{0,1\}^{2n\log n} \to (\{0,1\}^n)^n$ 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 $C(i) \in (\{0,1\}^n)^n$. Therefore, the problem is formulated as follows.

Problem 1 (The Sunflower problem). *Given a circuit $C\colon \{0,1\}^{2n\log n} \to (\{0,1\}^n)^n$ 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 2, 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 $\mathcal{D} = \{C\colon \{0,1\}^{2n\log n} \to (\{0,1\}^n)^n\}$ 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 7. *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 $T_1, \ldots, T_n \subseteq \{0,1\}^n$ 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,

$$\mathcal{F}_{\mathsf{no}} = \{S \in (\{0,1\}^n)^n \mid \forall i \in [n]\colon |S \cap T_i| = 1\}.$$

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 $h\colon \{0,1\}^{2n\log n} \to \{0,1\}^{n\log n}$ and define the circuit $C \otimes h\colon \{0,1\}^{2n\log n} \to (\{0,1\}^n)^n$ as: Given an index $i \in \{0,1\}^{2n\log n}$, we first hash it down to $h(i) \in \{0,1\}^{n\log n}$ 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. □

B Deferred Proofs for Ramsey Graphs

B.1 An upper bound on colorful Ramsey

We restate and prove Propositions 5 and 6.

Proposition 9 (Restatement of Proposition 5)**.** *For every $k > 2$ and $m > 1$, it holds that $R(\underbrace{k, \ldots, k}_{m \text{ times}}) \le m^{mk}$.*

Proposition 10 (Restatement of Proposition 6)**.** *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 5. Notice that the function $R(\cdot)$ is monotone, namely, if $k'_i \le k_i$ for every $i \in [m]$, then

$$R(k'_1, \ldots, k'_m) \le R(k_1, \ldots, k_m).$$

By Proposition 4, we know that

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

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 $R(\underbrace{2, \dots, 2}_{m \text{ times}})$. Notice that $R(\underbrace{2, \dots, 2}_{m \text{ times}}) = 2$.

The value of $R(\underbrace{k, \dots, k}_{m \text{ times}})$ 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 $R(\underbrace{k, \dots, k}_{m \text{ times}})$ 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

$$R(\underbrace{k, \dots, k}_{m \text{ times}}) \le m^{mk}.$$

□

Proof of Proposition 6. It is enough to show that $R(\underbrace{k, \dots, k}_{m \text{ times}}) \le N$. Thus, by Proposition 5, it is enough to show that $m^{mk} \le N$. Indeed, plugging in the value of m from the proposition, we get that

$$\begin{aligned}
\log(m^{mk}) &= mk \cdot \log m \\
&= k \cdot \frac{(\log N)/k}{\log\log N - \log k} \cdot \log\left(\frac{(\log N)/k}{\log\log N - \log k}\right) \\
&= \frac{\log N \cdot (\log\log N - \log k) - \log N \cdot \log(\log\log N - \log k)}{\log\log N - \log k} \\
&= \log N - \frac{\log N \cdot \log(\log\log N - \log k)}{\log\log N - \log k} \le \log N.
\end{aligned}$$

Thus, $m^{mk} \le N$, as required. □

B.2 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 11 (Restatement of Proposition 3)**.** *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

$$\binom{N}{k} \cdot 2 \cdot 2^{-\binom{k}{2}} \le \left(\frac{Ne}{k}\right)^k \cdot 2 \cdot 2^{-\binom{k}{2}}.$$

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 $1-1/N^{\Omega(\log\log N)}$. $\square$