DISS. ETH NO. 22433

Tradeoffs Between Computational Complexity and Information Content

A thesis submitted to attain the degree of

DOCTOR OF SCIENCES

of

ETH ZURICH

(Dr. sc. ETH Zurich)

presented by

Kfir Shlomo Barhum

M. Sc. in Computer Science and Applied Mathematics,
Weizmann Institute of Science

born June 23, 1980 in Tel-Aviv,
citizen of Israel

accepted on the recommendation of

Prof. Dr. Juraj Hromkovič, examiner
Prof. Dr. Peter Widmayer, co-examiner
Prof. Dr. Stefan Wolf, co-examiner

2015

Abstract

In this dissertation, we study trade-offs between computational complexity and information content of problems arising in different facets of theoretical computer science.

In the first part of the dissertation, we study the information content of online problems. For the online Steiner tree problem, we give matching upper and lower bounds on the advice complexity of the problem for the entire range of advice bits and thus resolve the problem. Then, we continue to study the randomness complexity of the disjoint path allocation problem. Our result shows that, in some cases, it is possible to use random bits instead of advice bits, while maintaining the same performance of the algorithm. In light of previous work on the power of advice bits for this problem, our algorithm is optimal.

The second part of the dissertation is devoted to constructions of cryptographic primitives, focusing on constructions of a universal one-way hash function from a one-way function. First, we relate the regularity assumptions made about the one-way function to the efficiency of the construction, measured by the number of calls it makes to the one-way function. Qualitatively speaking, the stronger the assumptions about the structure of the underlying one-way function are, the better the performance of the construction is. We then study the limits of black-box constructions for this task, which, loosely speaking, are constructions that only use the functionality of the one-way function, but not its description. Since their introduction by Naor and Yung (STOC, 1989) twenty-five years ago, no lower bound on the number of calls to the one-way function such a construction has to make was known other than the trivial one of a single call. We give a first lower bound of an almost-linear number of calls. Moreover, if the function is regular, our bound is tight.

Finally, in the last part of the dissertation, we continue the study of a recently introduced model of communication complexity with advice. We establish that the equality problem admits a protocol of polylogarithmic communication, provided a laconic advice of just one bit. For the divisibility problem, we design a protocol with sublinear communication and advice.

Zusammenfassung

In dieser Dissertation untersuchen wir das Verhältnis zwischen der Berechnungskomplexität und dem Informationsgehalt von Problemen, die in verschiedenen Zusammenhängen innerhalb der theoretischen Informatik auftreten.

Im ersten Teil der Dissertation untersuchen wir den Informationsgehalt von Online-Problemen. Für das Online-Steinerbaum-Problem geben wir übereinstimmende obere und untere Schranken für die Advice-Komplexität an, für das gesamte mögliche Spektrum von verwendeten Advice-Bits. Damit haben wir die Advice-Komplexität dieses Problems abschliessend geklärt. Weiterhin untersuchen wir die Anzahl von Zufallsbits, die notwendig und hinreichend ist, um ein Problem aus dem Bereich der Kommunikationsalgorithmen in Netzwerken, das sogenannte Disjoint-Path-Allocation-Problem, zu lösen. Unsere Ergebnisse zeigen, dass in einigen Fällen Zufallsbits anstelle von Advice-Bits verwendet werden können, ohne die Qualität der berechneten Lösung zu beeinträchtigen. Unter Berücksichtigung vorangegangener Arbeiten über die Ausdrucksstärke von Advice-Bits für dieses Problem ist unser Algorithmus optimal.

Der zweite Teil dieser Dissertation widmet sich Konstruktionen von kryptographischen Grundfunktionen, mit dem Schwerpunkt auf Konstruktionen einer universellen Einweg-Hashfunktion aus einer Einwegfunktion. Zunächst bringen wir die Regularitätsannahmen für die Einwegfunktion in Verbindung zur Effizienz der Konstruktion, gemessen in der Anzahl von Aufrufen der Einwegfunktion. Qualitativ gesprochen wird die Konstruktion umso effizienter, je stärker die Annahmen über die Struktur der zugrundeliegenden Einwegfunktion sind. Wir untersuchen dann die Grenzen von Black-Box-Konstruktionen für diese Aufgabe. Dies sind grob gesagt Konstruktionen, die nur die Funktionalität der Einwegfunktion verwenden, aber nicht deren Beschreibung. Seit der Einführung dieses Konzepts durch Naor und Yung (STOC, 1989) vor 25 Jahren ist keine bessere untere Schranke für die Anzahl der Aufrufe der Einwegfunktion bekannt als die triviale Schranke, dass ein Aufruf nötig ist. Wir geben eine erste nichttriviale untere Schranke an von einer fast linearen Anzahl benötigter Aufrufe. Weiterhin ist unsere Schranke bestmöglich für den Fall regulärer Funktionen.

Abschliessend setzen wir im letzten Teil der Dissertation die Untersuchung eines kürzlich eingeführten Modells der Kommunikationskomplexität mit Advice fort. Wir zeigen, dass für das Gleichheitsproblem ein Protokoll mit polylogarithmischer Kommunikation existiert, das einen lakonischen Advice

von nur einem Bit erhält. Für das Teilbarkeitsproblem entwerfen wir ein Protokoll mit sublinearer Kommunikation und sublinearem Advice.

Acknowledgments

First and foremost, I would like to thank my adviser Juraj Hromkovič. From the day I joined his group Juraj has always been very motivating and supportive of my work. I am grateful for the great opportunity I had to conduct research under his wise supervision. Many of the problems addressed in this dissertation were suggested by him, and his passion for science was always an inspiration.

I am grateful to both Peter Widmayer and Stefan Wolf for serving on my dissertation committee and for reviewing my thesis.

I would like to thank my co-authors Ueli Maurer and Thomas Holenstein. I have learned a lot from our interaction.

I am thankful to David Adjiashvili, Sandro Coretti and Hans-Joachim Böckenhauer for reading early drafts of my work and for their valuable suggestions regarding its presentation, often shortly before a deadline.

The warm and friendly atmosphere at the ITE group is owed to all its members. In particular I want to thank Dennis Komm, Björn Steffen, Jasmin Smula, Hans-Joachim Böckenhauer, Jan Lichtensteiger and Giovanni Serafini for their kind way and for their friendship.

I would like to thank collectively all the friends I met since I had arrived at ETH, and especially Annina, Daniel, David, Deniz, Leander, Malin, Matthias, Sandro and Stefan, who all helped making Zurich a second home.

Last but not least, I want to thank my family and close friends from back home: My parents Arik and Rimona, my sister Noa, and Sella for their endless love and support, and Amir, Dima, Gilad, Maya and Sella, for walking along the road all those years; for all our great moments together and for always being there for me.

Zurich, December 2014 Kfir Barhum

Contents

Chapter 1

Introduction

3 x 3 macht 6 - widdewidde
Wer will's von uns lernen?
Alle gross und klein
trallalala lad' ich zu uns ein.

– Pippi Langstrumpf

A general phenomenon common perhaps to all areas of engineering and prominent when dealing with computational problems is the following: The more we know about a problem a-priori, the better we can perform when solving it. The research area of theoretical computer science allows to address such problems in a rigorous manner, using abstract mathematical models in which precise statements can be made. In a nutshell, this dissertation is concerned with studying this phenomenon in three different computational settings: online computing, constructions of cryptographic primitives and distributed computing.

In the setting of online computing, the algorithm receives its input in a piece-wise manner over time and must commit to its partial output before receiving the next part of the input. Here, the performance of the algorithm is measured with respect to the quality of the solution it computes. A-priori information about the problem instance can be quantified using a model, where some function of the entire input (called advice) is available to the algorithm. Trying to characterize the information content of the problem, one studies trade-offs between the availability of information and the quality of the solution.

In the context of cryptography, the current state-of-the-art allows only conditional results. Specifically, the existence of efficient secure cryptographic primitives (e.g., a secure encryption scheme) is established under unproven

assumptions (e.g., the existence of some hard computational tasks, such as inverting an easily computable function). One usually considers different computational assumptions and studies their cryptographic consequences (e.g., secure encryption), where a typical problem is to come-up with such construction. These are algorithms that implement a cryptographic primitive from a more basic one, while establishing the security of the former assuming only that of the latter. The performance of the construction is usually measured either by its complexity in some parameter, e.g., the number of invocations of the basic cryptographic primitive or its running time, or in the tightness of its security, which measures how well the security of the constructed primitive relates to that of the basic one.

In this case, one can consider the following two forms of a-priori information about the problem: One the one hand, it is possible to impose further structural requirements on the class of primitives considered. Effectively, we would now only require a construction that is guaranteed to implement a secure primitive for a sub-class of the basic primitive that satisfies the additional properties. On the other hand, we could allow the construction some auxiliary information about every instance that implements the primitive, which is now instance-dependent, in a form of a non-uniform advice. As we shall see, in some cases such assumptions lead to a dramatic improvement in the quality of the construction.

The last setting we consider is that of distributed computing, where some input is distributed among two parties and their goal is to compute some known function of it. Here one studies the efficiency of the interaction, namely, the minimal amount of communication required to compute the function. In this case, one studies trade-offs between auxiliary information in the form of advice given to one of the parties, and the quality of the obtained protocol.

Before we introduce each of the considered models in more detail, we first give an overview of the dissertation.

1.1 Overview of the Dissertation

In this dissertation we study trade-offs between information about a computational problem and its computational complexity. We study aspects of such trade-offs in three different computational models.

In the first part of the dissertation we continue the study of the information content of online problems. Using the model of Hromkovič et al. [HKK10],

we study different trade-offs of information resources and the performance of the algorithm: In Chapter 2 we focus on the well-known Steiner tree problem in the online setting and give a full characterization of its advice complexity. In Chapter 3 we study the online disjoint path allocation problem. Here we characterize the exact power of random bits. We show that in some cases it is possible to replace advice bits with random bits while maintaining the same performance of the algorithm.

In the second part of the dissertation, we study constructions of cryptographic primitives. Focusing on constructions of universal one-way hash functions from one-way functions, we study the relation between the efficiency of the construction and the assumption made about the one-way function and the possible gain from auxiliary information about these assumptions.

In Chapter 4, we relate the regularity assumptions made on the one-way function to the overall performance of the construction. For the special case where the one-way function is regular, that is, a one-way function for which each image has the same number of preimages, we observe that very little information about the function, namely, its regularity parameter, leads to an exponential improvement in the efficiency of the construction, reducing the number of calls to the one-way function made by the construction from linear to logarithmic.

Next, we fully explain this gap: In Chapter 5, we develop a framework for proving lower bounds on constructions of cryptographic primitives and in Chapter 6 we give our main application of the framework: An almost linear lower bound on the number of calls made by a construction of a universal one-way hash function from a one-way function. Our bound holds already for the cases considered in Chapter 4 and thus completely settles the question for constructions from regular one-way functions.

In the last part of the dissertation, we consider a relatively new model of communication complexity with advice, where we study trade-offs of information versus efficiency of communication. In Chapter 7, we focus on divisibility and equality. For equality, we show that a laconic advice of one bit already allows a protocol with only polylogarithmic communication. Finally, we give lower and upper bounds for the communication complexity of divisibility with advice.

Most of the results explained in this dissertation have already been published. The results on the online Steiner tree problem from Chapter 2 were presented at SOFSEM 2014 [Bar14]. At this conference a joint paper [BBF+14] containing the results on the online disjoint path allocation problem from Chapter 3 was

presented. The construction of a universal one-way function from a one-way function depicted in Chapter 4 was presented at LATINCRYPT 2012 [BM12]. A paper based on the results presented in Chapters 5 and 6 was published at TCC 2013 [BH13], and a full version of it is available as a technical report [BH12].

The rest of this chapter is organized as follows. In the next three sections, we give a general introduction to the three corresponding parts of the dissertation and in Section 1.5 we present some mathematical preliminaries.

1.2 Online Problems and their Advice Complexity

Many important practical computational problems are best formulated in an online scenario, where the input arrives piecewise over time and an online algorithm has to irrevocably compute a part of the output for any given part of the input. Thus, online algorithms are a realistic model for making decisions under uncertainty.

As opposed to classical computational problems, in the online setting the full input to the problem is not known in advance, but is revealed in a step-wise manner, and after each step the algorithm has to commit to a part of its solution.

More formally, an online problem is a triplet $U = (\mathcal{I}, \mathcal{S}, \text{cost})$. $\mathcal{I}$ is the set of instances to the problem, where an instance to an online problem is a vector $I = (x_1, \ldots, x_n)$, where its i'th component, x_i is the part of the input revealed at time-step i. The set $\mathcal{S}$ is a relation[1] on $\mathcal{I} \times \mathcal{Y}$, where $\mathcal{Y}$ is the set of all possible output vectors that describes for each instance its admissible answer vectors, and $\text{cost} : \mathcal{S} \to \mathbb{R}$ is a function that assigns a value for every instance and a possible solution it.

An online algorithm is a machine A that interacts with an instance I of an online problem U in a step-wise manner as follows: In time step i, the input part x_i is revealed to the machine, and it proceeds with its computation and outputs y_i. The output sequence of the algorithm $\mathsf{A}(I) = (y_1, \ldots, y_n)$ is a

[1] More precisely, we require that $\mathcal{S}$ contains only pairs of vectors of matching size, has at least one solution for every instance and moreover, it should satisfy the following condition on prefixes of instances: For every instance $(I, Y) \in \mathcal{S}$ and every I', let p denote the size of the maximal common prefix of I and I'. Then we require that there exists a solution to I' such that maximal common prefix of Y and Y' is at least p.

solution to I if $(I, \mathsf{A}(I)) \in \mathcal{S}$. An online algorithm solves U (or is an online algorithm for U) if its output sequence on every instance is a solution.

We shall consider the problem of optimizing U with respect to minimization (or maximization) of the cost function. In the discussion that follows, we shall focus on minimization problems, but maximization problems are treated analogously in a straightforward manner.

We consider a choice function $\mathsf{Opt} : \mathcal{I} \to \mathcal{Y}$ that assigns for each instance a solution of minimal value. Note that the online problems that are usually considered are reasonable online formulations of classical offline problems. In this case it will be convenient to think of Opt as an (offline) computationally unbounded algorithm that has access to the entire input sequence in advance and for every instance outputs an optimal solution.

Classically, the performance of online problems is measured using the so-called competitive analysis introduced by Sleator and Tarjan [ST85], where the cost of the solution computed by the online algorithm is compared to that of an optimal (offline) algorithm that knows the complete input beforehand in a worst-case sense. Formally, let $c \geq 1$.

Definition 1 (c-competitive online algorithm)**.** *An online algorithm* A *is* c-competitive *for the minimization problem U if there exists a constant α such that, for every instance I,*

$$\mathrm{cost}(\mathsf{A}(I)) \leq \ c \cdot \mathrm{cost}(\mathsf{Opt}(I)) + \alpha \ . \tag{1.1}$$

If $\alpha = 0$, then A *is* strictly c-competitive.

In general, $c = c(\cdot)$ is a function of some parameter of the instance.

In the case we consider a maximization problem, Opt assigns a solution of maximal value to every instance, and the inequality in (1.1) becomes $\mathrm{cost}(\mathsf{A}(I)) \geq \ \mathrm{cost}(\mathsf{Opt}(I))/c - \alpha$.

For a very good introduction to online algorithms, we refer to the textbook by Borodin and El-Yaniv [BEY98].

1.2.1 Online Algorithms with Advice

In recent years, motivated among other reasons by the fact that for some problems (e.g., Knapsack [BKKR12]) no deterministic algorithm can admit any competitive ratio, the natural question, "how much information about the

future is needed in order to produce a competitive solution?", was posed by Dobrev et al. [DKP08] and Böckenhauer et al. [BKK+09], and independently by Emek et al. [EFKR11]. In this dissertation we use the framework of Hromkovic et al. [HKK10], that unifies the models and allows to study the information content of an online problem, and poses the question in its full generality: "What is the exact power of advice bits for some specific online problem?".

In online computation with advice, the algorithm's machine has access to a special infinite advice string ϕ, produced by an oracle that has access to the entire input. The general goal is to try to characterize the dependence of the achievable competitive ratio on the maximal number of advice bits read from the advice tape.

An online algorithm A with advice computes the output sequence as before, but additionally may access a special advice tape. For some fixed advice ϕ, we denote by A^{ϕ} the algorithm with advice ϕ.

The dependence of the quality of the solution on the number of advice bits read is formalized by the following definition: Let $c > 0$ and $b \geq 0$:

Definition 2. *The algorithm* A *is* c-competitive with b bits of advice *if there is a constant* α *such that, for every instance* I*, there exists an advice* ϕ *such that* $\mathrm{cost}(\mathsf{A}^{\phi}(I)) \leq c \cdot \mathrm{cost}(\mathsf{Opt}(I)) + \alpha$ *holds, and at most the first* b *bits of* ϕ *are accessed during the computation of* A^{ϕ} *on* I*.*

In general, $c = c(\cdot)$ and $b = b(\cdot)$ are functions that depend on some parameter of the instance, typically its length.

We observe that in all cases considered in the literature, the advice function is a computable function of the input, although this is not required from the definition.

In general, the design of online algorithms with advice requires an understanding of the information content of the problem considered, and offers a characterization of the b most relevant bits of information of the instance I.

A number of online problems have already been analyzed within this model, such as paging [BKK+09], buffer management [DHZ12], job shop scheduling [BKK+09, KK11], the k-server problem [BKKK11], online set cover [KKM12], string guessing [BHK+14], metrical tasks systems [EFKR11], graph exploration [DKM12], independent set [DKK12], knapsack [BKKR12], and graph coloring problems [FKS12, SSU13, BBHK12, BBH+13].

For further introduction to the advice complexity of online problems, see [BKK+09, HKK10].

1.3 Complexity-Based Cryptography

Complexity-based cryptography is the research field that studies the possibility of basing cryptography on complexity-theoretic assumptions. The current state of knowledge in theoretical computer science is such that we do not know how to formally prove (using only "standard" mathematical axioms) the existence of *any* practical cryptographic primitive (for example, a secure encryption system). At the same time, some specific assumptions are widely believed to hold, and in fact, have become an integral part of our every-day life: Buying a book on-line from a website or reading our email privately are just two such examples. Yet, it is possible that in the future the cryptographic constructions, which are based on those assumptions, will be broken, and the underlying assumption disproved.

Consequently, over the last few decades this has led to the development of a complexity-based cryptographic theory that is independent of specific assumptions. On the one hand, this theory studies the minimal assumptions needed to allow certain primitives, and on the other hand, the implications of the existence of those primitives. Instead of basing the security of a cryptographic primitive (e.g., a symmetric encryption system) on a *specific* mathematical assumption (e.g., the assumption that factoring a large number is not algorithmically feasible), the idea is to model the general hardness assumptions made about such a function, and then to prove the security of some cryptographic construction using only those general properties (which are common to a large class of functions).

A key concept that models an abstract hard function is a *one-way function*. Loosely speaking, these are functions that are easy to compute but hard to invert. The "easy-to-compute" part essentially means that there is an efficient procedure that evaluates the function, whereas the "hard-to-invert" requires that any efficient computational procedure when given input y such that $y = f(x)$ for a "typical" x, fails to invert it (to compute an x' such that $f(x') = y$) except with negligible probability (e.g., the algorithm can always guess such an input, but the probability that it guesses correctly is small). A typical input for f is modeled as one chosen uniformly at random from its domain $\{0,1\}^n$. For each security parameter ρ, the domain and range of the function are $\{0,1\}^n$ and $\{0,1\}^m$, respectively, where $n = n(\rho)$ and $m = m(\rho)$

are functions of the security parameter. Formally, we define:

Definition 3 (OWF). *A family of functions* $\{f_\rho\}_{\rho\in\mathbb{N}^+}$*, where* $f_\rho : \{0,1\}^{n(\rho)} \to \{0,1\}^{m(\rho)}$ *is a* one-way function *if:*

1. *There exists an efficient algorithm that given* 1^ρ *and* x*, outputs* $f_\rho(x)$*.*
2. *For any efficient randomized algorithm* A*, the function that maps* ρ *to*

$$\Pr_{x\xleftarrow{r}\{0,1\}^{n(\rho)},A}\left[A(1^\rho, f_\rho(x)) \in f_\rho^{-1}(f_\rho(x))\right]$$

is negligible[2]*.*

That is, finding *any* element x' that maps to $f(x)$ should be difficult, and note that the probability is taken over both the randomness of A and a uniform argument to f (denoted under the Pr sign). The reason we need to give A access to 1^ρ is two-fold: First note that it may be possible that for $\rho \neq \rho'$ it is the case that $m(\rho) = m(\rho')$, in which case we orient the algorithm towards a specific one. The second reason being that for a shrinking f, an efficient algorithm on input from $\{0,1\}^{m(\rho)}$ may not have enough time to output any string from $\{0,1\}^{n(\rho)}$, and we want to rule out this case.

As noted above, one-way functions are of fundamental importance in complexity-based cryptography. Their existence implies the existence of other fundamental primitives. Pseudo-random generators (algorithms that are able to extend a random string such that their output "looks" random to any feasible distinguishing algorithm), symmetric-key encryption systems, digital signatures and universal one-way hash functions (UOWHFs) are just a few such examples. The latter (a UOWHF) is a shrinking function for which finding a second preimage is infeasible.

Loosely speaking, a UOWHF is an efficiently computable, compressing, and keyed function, with the property that it is infeasible for an adversary to win the following game: First, it commits to some input value v from the domain. Next, a random key is chosen and given to the adversary. Finally, the adversary "wins" the game if it outputs v' different from v such that they both map to the same value under the function (using the chosen random key). Such an v' is called a *non-trivial collision* for v. Note that since the function is length-decreasing, there exist many inputs with non-trivial collisions. A function that enjoys this property is sometimes called a second-preimage collision-resistant function. For each security parameter ρ, the key

[2]Recall that a function g is negligible if it vanishes faster than the inverse of *any* polynomial. That is if for every c and all sufficiently large n, $g(n) < n^{-c}$ holds.

space, domain and range of the function are $\{0,1\}^{\kappa}$, $\{0,1\}^{m}$ and $\{0,1\}^{m'}$, respectively, which depend on the security parameter ρ.

Formally, we have:

Definition 4 (UOWHF). *A family of functions $h = \{h_\rho\}_{\rho\in\mathbb{N}^+}$, where $h_\rho : \{0,1\}^{\kappa(\rho)} \times \{0,1\}^{m(\rho)} \to \{0,1\}^{m'(\rho)}$ is a* universal one-way hash function *if*

1. *There exists an efficient algorithm that given $1^\rho, k$ and v outputs $h_\rho(k,v)$.*
2. *$m'(\rho) < m(\rho)$.*
3. *For any pair of efficient randomized algorithms (B_1, B_2) the function mapping ρ to*

$$\Pr_{\substack{(v,\sigma)\xleftarrow{r}B_1(\rho)\\ k\xleftarrow{r}\{0,1\}^{\kappa(\rho)}\\ v'\xleftarrow{r}B_2(k,v,\sigma)}}[h_\rho(k,v) = h_\rho(k,v') \wedge v \neq v']$$

is negligible.

The second output σ of the algorithm B_1 allows it to save its state, to be used by B_2 after the key is chosen.

Two important cryptographic primitives of significant practical importance that can be obtained [NY89] directly from any UOWHF are digital fingerprints and digital signatures.

For a comprehensive introduction to modern cryptography, we refer to the textbook of Goldreich [Gol01].

1.3.1 Cryptographic Constructions and Black-Box Constructions

In its most general sense, a construction of a cryptographic primitive P from another primitive Q is a mathematical proof of $\exists Q \Rightarrow \exists P$. Of course, whereas from a purely theoretical point of view such a proof is satisfactory, in cryptography, a field with a vast practical impact, explicit and constructive constructions are sought. A constructive proof would provide an algorithm that implements P using an algorithm that implements Q, along with a security proof that P is secure assuming the security of Q for their respective security notions.

While a priori such a construction (which is actually an algorithm along with its security proof) may assume an arbitrary structure, it turns out that the vast majority of constructions of primitives found in the literature follow a very specific pattern known as *fully black-box*. Informally, an algorithm A makes black-box usage of some algorithm B, if it only uses its functionality. That is, its only interaction with B's code is executing it on some inputs. An algorithm A is a black-box implementation of primitive P (using primitive Q) if it efficiently implements P using Q only in a black-box manner. The term *implements* refers to the syntactical properties (for example, computing a function with the correct domain and range). Moreover, almost all security proofs given for cryptographic constructions are black-box security reductions from the problem of breaking P to that of breaking Q. That is, they show an efficient algorithm for breaking Q that only makes black-box usage of a potential adversary for P with an analysis that relates the performance of the adversary of P to the one constructed for Q. A fully-black-box construction of a primitive P from a primitive Q is a black-box implementation of P from Q along with a black-box security reduction. Reingold et al. [RTV04] were the first to formally define the notion of a fully-black-box construction and study the various nuances of such constructions.

1.3.2 Impossibility Results and Lower Bounds

Failing to come up with constructions of some primitives from others has led researchers to study the impossibility of such constructions. In such a case, a proof that primitive P cannot be implemented efficiently from primitive Q is sought. However, if one assumes that primitive P does exist, then P can be implemented from Q in a trivial manner (by simply ignoring it). Thus, when considering impossibility proofs, we need to be more careful. Using the notion of a black-box construction, the impossibility results considered are those that actually use Q (both in the construction and the security proof). A *black-box separation* of P from Q is a proof that primitive P cannot be fully-black-box constructed from primitive Q.

In the literature, we identify two main types of impossibility results: Black-box separation of a cryptographic primitive P from a primitive Q and lower bounds on a complexity measure for any black-box construction of a primitive P from a primitive Q. Bounds of the former type are usually sought after unsuccessful attempts to construct P from Q were made. Bounds of the latter type are sought when some construction that implements P from Q is known, but is not optimal in some complexity parameter (i.e., running-time, number

of queries to Q, seed length). In this case, a possible result would yield that any black-box construction of P from Q must make at least q queries to the underlying primitive Q. Of course, q may depend on the specific parameters required from P, or the assumptions made about Q.

As two important examples, we mention the work of Impagliazzo and Rudich [IR89], who ruled out a black-box construction of a secret key-agreement from a one-way permutation[3] and that of Simon [Sim98], who proved that there is no black-box construction of a collision-resistant hash function[4] from a one-way function.

1.4 Communication Problems with Advice

The research field of communication complexity concerns with the efficiency of *interaction*. The theory of communication complexity quantifies and studies the amount of communication required for different settings of distributed computing between two entities that are allowed to communicate over some channel. In this case, local computations are assumed to be unbounded, i.e., we do not limit the entities with respect to the time and space complexities of their local computations, and are solely interested in the amount of information exchanged during the computation, measured usually by the total number of bits exchanged by the parties. The communication between the parties is guided by a protocol which specifies how each message sent depends on the input and the messages sent previously.

More formally, two computationally unbounded players Bob and Charlie hold partial inputs $x \in \mathcal{X}$ and $y \in \mathcal{Y}$, respectively, to a function $f : \mathcal{X} \times \mathcal{Y} \to \{0,1\}$ known to both of them. The parties interact according to a protocol π, which is modeled as a finite sequence of functions $(M_1, \ldots, M_r)$, where $M_i : \mathcal{X} \times (\{0,1\}^+)^{i-1} \to \{0,1\}^+$ for an odd i and $M_i : \mathcal{Y} \times (\{0,1\}^+)^{i-1} \to \{0,1\}^+$ for an even i, specifying the i'th message of the protocol. The computation proceeds as follows: In the first round, the message $m_1 = M_1(x)$ is sent by Bob, in the second round $m_2 = M_2(y, M_1(x))$ is sent by Charlie, and in general, in the i'th round message $M_i(x, m_1, \ldots, m_{i-1})$ (respectively, $M_i(y, m_1, \ldots, m_{i-1})$) for odd (resp., even) i is sent.

[3] A one-way function that is additionally a permutation on its domain.

[4] These cryptographic hash-functions require that no collision to a randomly chosen function can be found, where the adversary has the freedom to choose both inputs to the function *after* receiving the key. This implies that every collision-resistant hash function is in particular a universal one-way hash function.

The transcript of a protocol on inputs x and y is $\pi(x,y) \stackrel{\text{def}}{=} (m_1, \ldots, m_r)$. The length of the transcript $|\pi(x,y)|$ is defined as the total length of all the messages exchanged.

A protocol is correct if, for all $x \in \mathcal{X}$ and $y \in \mathcal{Y}$, there exists a referee function EVAL, such that $\text{EVAL}(\pi(x,y)) = f(x,y)$. Put differently, if it is possible to compute $f(x,y)$ only by looking at the transcript.

Finally, the communication complexity of a protocol π is $\max_{x,y} |\pi(x,y)|$ and the deterministic communication complexity of a function $f : \mathcal{X} \times \mathcal{Y} \to \{0,1\}$ is defined as $\mathsf{CC}(f) \stackrel{\text{def}}{=} \min_\pi \max_{x,y} |\pi(x,y)|$, where the minimum is taken over all correct protocols for f.

Observe that for every function there exists a trivial protocol: Bob sends his entire input to Charlie, who locally computes $f(x,y)$ and announces the output. The question one is usually interested in, is "What is the minimal amount of communication between the parties required to compute f?".

For example, the equality function, $\text{EQ} : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ defined by $\text{EQ}(x,y) = 1$ iff $x = y$, can be shown to require a deterministic communication complexity of at least $n+1$ bits.

In the literature many variations of this model are studied, and in particular various models that involve randomness and non-determinism. For a thorough introduction to communication complexity we refer to the textbooks of Hromkovič [Hro97] and Kushilevitz and Nisan [KN97].

1.4.1 Communication Complexity with Advice

First observe that simply adding an advice, which depends on the inputs, of even just one bit to the classical model of communication complexity does not seem to make sense, as the advice bit $f(x,y)$ immediately yields a trivial protocol for the problem.

However, motivated by the problem of proving polynomial lower-bounds on the efficiency of dynamical data structures, Pătraşcu [Pat10] has recently suggested the following model, where Bob has input $x \in \mathcal{X}$ (just as before), but Charlie is given as input k elements $y_1, \ldots, y_k$ from $\mathcal{Y}$. Then, a third party Alice, the advisor, receives both inputs and computes an advice string which she sends to Bob and then remains silent. Finally, Bob and Charlie are presented an index i and are allowed to interact by exchanging messages, where their goal is to compute $f(x, y_i)$.

More formally, a protocol π with m advice bits for the k-instance problem is $\pi = (\pi_a, M_1, \ldots, M_r)$, where $\pi_a : \mathcal{X} \times \mathcal{Y}^k \to \{0,1\}^m$ is the advice function of the protocol and

$$M_i : \mathcal{X} \times [k] \times \{0,1\}^m \times (\{0,1\}^+)^{i-1} \to \{0,1\}^+$$

for an odd i and

$$M_i : \mathcal{Y}^k \times [k] \times (\{0,1\}^+)^{i-1} \to \{0,1\}^+$$

for an even i.

The computation on inputs $x, y_1, \ldots, y_j, i$ proceeds similarly to before: First, the advice $a = \pi_a(x, y_1, \ldots, y_k)$ is given to Bob, and then the interaction continues as follows: Bob sends message $m_1 = M_1(x, i, a)$ to Charlie, who replies with message $m_2 = M_2(y_1, \ldots, y_k, i, m_1)$ and so forth.

We stress that it is essential that only Bob receives the advice. Otherwise, for example, whenever $m > \log(\mathcal{X})$, the advice could already encode Bob's input, and Charlie could locally compute the answer.

Also, note that the problem is only interesting in the case where $m < k$. Otherwise, a trivial protocol always exists, where the advice just encodes the answer vector $(f(x, y_1), \ldots, f(x, y_k))$.

As before, for $y^{(k)} \stackrel{\text{def}}{=} (y_1, \ldots, y_k)$, the transcript $\pi(x, y^{(k)}, i)$ of π for inputs $x, y^{(k)}$ and i is is the list $(m_1, \ldots, m_r)$ of all messages exchanged during its computation on inputs $x, y^{(k)}$. As before, a protocol is correct if it is possible to compute $f(x, y_i)$ only by looking at its transcript. We define the communication complexity of f for k inputs with m bits of advice as

$$\mathsf{CC}^{\mathsf{k}}_{\mathsf{m}}(f) \stackrel{\text{def}}{=} \min_{\pi} \max_{x, y^{(k)}, i} \left| \pi(x, y^{(k)}, i) \right| \quad ,$$

where the minimum is over all protocols π that correctly compute $f(x, y_i)$ for every input $(x, y^{(k)}, i)$.

As mentioned, Pătraşcu offered a plausible approach for lower bounds to a host of dynamic data structure problems via a series of reduction from the problem of set-disjointness in the communication complexity with advice model on which super-polynomial lower bounds are conjectured. In particular, one such problem is subgraph connectivity, where, after a preprocessing of an undirected graph, the data structure supports on/off operation for vertices and queries for pair of vertices u, v asking whether there is a path using only

"on" vertices from u to v. Another problem is Langerman's problem, where it is required to maintain updates on an array of length n and support answering the zero-partial-sum question, namely, does there exist a non-empty subset of indices that sum to zero. We refer to Section 1.1 in [Pat10] for a complete taxonomy.

Thus, the communication complexity with advice model is well-motivated, whose study offers a promising approach towards polynomial lower bounds on the aforementioned problems.

1.5 Mathematical Preliminaries

Throughout the thesis we use standard notations from the literature, which we assume the reader is familiar with. For the sake of completeness, we bring a short overview.

We denote by $\mathbb{N} \stackrel{\text{def}}{=} \{0, 1, 2, \ldots\}$ the set of natural numbers and by $\mathbb{N}^+ \stackrel{\text{def}}{=} \mathbb{N} \setminus \{0\}$. Similarly, we denote by $\mathbb{R}$ and $\mathbb{R}^+$ the set of real numbers and positive real numbers, respectively. For two integers $n, m \in \mathbb{N}^+$, we denote by $[n]$ the set $\{1, \ldots, n\}$ and by $\gcd(n, m)$ and $\text{lcm}(n, m)$ their greatest common divisor and least common multiplier, respectively. All logarithms considered in this dissertation are to the base 2. For two bit-strings x and y we denote by $x \| y$ their concatenation, by $(x)_q$ the q'th bit of x, and by $(x)_{1,\ldots,q}$ the first q bits of x.

We use the standard Landau notation O, Ω, o and ω. Additionally, for a function $f : \mathbb{N} \to \mathbb{R}$, we denote by $\tilde{O}(f)$ the set of all functions that are asymptotically dominated by f ignoring logarithmic factors, i.e., $g \in \tilde{O}(f)$ if and only if there exist $c, d \in \mathbb{N}$ such that $g(n) \leq cf(n)\log^d(n)$ for all sufficiently large n.

We shall introduce more specific notations in the relevant chapters as needed.

1.5.1 Notions of Efficiency of Computation

A function $p = p(\rho)$ is polynomial if there exists a c such that $p(\rho) = \rho^c$. A machine M is efficient if there exists a polynomial p such that on every input $x \in \{0, 1\}^*$, $M(x)$ halts after at most $p(|x|)$ steps. We shall use the terms machine and algorithm synonymously. A function $s : \mathbb{N}^+ \to \mathbb{N}^+$ is a security

function if for every $\rho \in \mathbb{N}^+$ it holds that $s(\rho+1) \geq s(\rho)$, and s is efficiently computable (i.e., there exists an efficient machine M that on input 1^ρ outputs $s(\rho)$). For a security function s we define $\frac{1}{s} : \mathbb{N}^+ \to \mathbb{R}^+$ as $\frac{1}{s}(\rho) \stackrel{\text{def}}{=} \frac{1}{s(\rho)}$. A function $f : \mathbb{N}^+ \to \mathbb{R}^+$ is negligible if it vanishes faster than the inverse of any polynomial, i.e., if for all polynomial security functions p it holds that $f(\rho) < \frac{1}{p(\rho)}$ for all sufficiently large ρ. Sometimes we shall denote $\texttt{negl}(n)$ to denote some specific negligible function that is clear from the context.

1.5.2 Probability Theory

We use capital letters to denote random variables and small letters for specific values they assume. For a random variable X we denote by $\mathbf{E}[X]$ and $\mathbf{V}[X] = \mathbf{E}[(X - \mathbf{E}[X])^2]$ its expectation and variance, accordingly. We recall a few of their properties: Let X and Y be two random variables, and let $a, b \in \mathbb{R}$. The expectation is linear, i.e., $\mathbf{E}[aX + Y] = a\,\mathbf{E}[X] + \mathbf{E}[Y]$. The variance of a random variable satisfies $\mathbf{V}[aX + b] = a^2\,\mathbf{V}[X]$. If additionally X and Y are independent, we have $\mathbf{V}[X + Y] = \mathbf{V}[X] + \mathbf{V}[Y]$.

For an event A, we denote its indicator random variable (which assumes the value 1 whenever A happens and 0 otherwise) by $\mathbb{1}_A$, and its complement event by $\overline{A}$. We implicitly make use of the fact that $\mathbf{E}[\mathbb{1}_A] = \Pr[A]$. The support of a random variable X is defined as $\mathrm{Supp}(X) = \{x : \Pr[X = x] > 0\}$. For a non-empty set S we denote by $x \xleftarrow{r} S$ choosing an element x uniformly at random from S and for a random variable X we denote $x \xleftarrow{r} X$ for choosing x according to X.

When we want to emphasize the considered random experiment with some fixed partial randomness, we describe the random experiment under the Pr (respectively, $\mathbf{E}$ or $\mathbf{V}$) symbol. For example, when X and Y are two independent random variables and we consider the random experiment where Y is fixed for some value y, we shall write $\Pr_{x \xleftarrow{r} X}[x = 2y]$ (instead of $\Pr[X = 2y]$) to stress that we consider the random experiment of choosing a random x.

For further introduction to probability theory, we refer to the textbook of Ross [Ros10]. Throughout the dissertation, we make extensive use of the probabilistic method. We bring next some probabilistic inequalities about concentration of a random variable around its expected value.

Markov's Inequality. Let X be a non-negative random variable with finite expectation. Let $a > 0$. Then:

$$\Pr[X \geq a\,\mathbf{E}[X]] \leq \frac{1}{a} \ . \tag{1.2}$$

As a corollary from Markov's inequality we immediately obtain Chebyshev's inequality:

Chebyshev's Inequality. Let X be a random variable with finite expectation and variance. Let $a > 0$. Then:

$$\Pr[|X - \mathbf{E}[X]| \geq a] \leq \frac{\mathbf{V}[X]}{a^2} \ . \tag{1.3}$$

Next we bring two useful inequalities about the sum of *independent* random variables:

The Chernoff Bound. Let $\{X_i\}_{i=1}^t$ be independent identically distributed random variables with $\Pr[X_i = 1] = \Pr[X_i = 0] = \frac{1}{2}$, and let $a > 0$. Set $X \stackrel{\text{def}}{=} \sum_{i=1}^t X_i$. Then

$$\Pr[X > \mathbf{E}[X] - a], \Pr[X < \mathbf{E}[X] - a] < \exp\left(\frac{-2a^2}{t}\right) \ . \tag{1.4}$$

Finally, we present a more general version of the Chernoff bound for bounded independent random variables.

The Hoeffding Bound. Let $\{X_i\}_{i=1}^t$ be independent random variables, where $X_i \in [a_i, b_i]$, and let $k > 0$. Set $X \stackrel{\text{def}}{=} \sum_{i=1}^t X_i$. Then

$$\Pr\left[X - \mathbf{E}[X] \geq k\right] \leq \exp\left(\frac{-2k^2}{\Sigma_{i=1}^t (b_i - a_i)^2}\right)$$

and

$$\Pr\left[X - \mathbf{E}[X] \leq k\right] \leq \exp\left(\frac{-2k^2}{\Sigma_{i=1}^t (b_i - a_i)^2}\right) \ .$$

The proofs of the inequalities can be found in Appendix D.1 in [Gol08] and in [Hoe63].

1.5.3 Combinatorial Hash Functions

In Chapter 4, we shall make use of the following combinatorial hash functions.

t-wise Independent Hashing. Let $\mathcal{G}_n^m \stackrel{\text{def}}{=} \{g_k\}_{k\in K}$ be a family of functions, where $g_k : \{0,1\}^n \to \{0,1\}^m$.

$\mathcal{G}_n^m$ is t-wise independent if for all range elements $y_1, \ldots, y_t \in \{0,1\}^m$ and all distinct domain elements $x_1, \ldots, x_t \in \{0,1\}^n$ it holds that

$$\Pr_{k \stackrel{r}{\leftarrow} K} [g_k(x_1) = y_1 \wedge \cdots \wedge g_k(x_t) = y_t] = 2^{-tm} .$$

Put differently, for any distinct t domain elements, the distribution of their images on the range under a uniform key from the family is perfectly uniform.

The family is called constructible if, for all $y_1, \ldots, y_s$ and distinct $x_1, \ldots, x_s$ where $s \leq t$, it is possible to sample a function uniformly subject to $g_K(x_1) = y_1, \ldots, g_K(x_s) = y_s$.

There exist t-wise independent hash functions where the description of a function is linear in the logarithm of the domain size.

Specifically, taking $K = \{0,1\}^n \times \{0,1\}^n \times \{0,1\}^n$, for $k = (a_2, a_1, a_0) \in K$ the family $\mathcal{G}_n^n = \{g_k : \{0,1\}^n \to \{0,1\}^n\}_K$ defined by $g_k(x) = a_2 \cdot x^2 + a_1 \cdot x + a_0$, where the operations and a_2, a_1, a_0 and x are understood as operations and elements, respectively, in the finite field of cardinality 2^n, is a constructible three-wise independent family.

It is easy to check that if $\mathcal{G}_n^n$ is a t-wise independent family, then by truncating the last $n - l$ bits of $\mathcal{G}_n^n$ one gets a t-wise independent family $\mathcal{G}_n^l$. Moreover, if $\mathcal{G}_n^n$ is constructible, then so is $\mathcal{G}_n^l$.

We will also need the following fact: If $\mathcal{G}_n^l$ is a constructible t-wise independent family, then for any $x \neq y \in \{0,1\}^n$ it is possible to sample a uniform g_k from $\mathcal{G}_n^l$ subject to the condition that $g_k(x) = g_k(y)$.

For further introduction to t-wise independent hashing we refer to Appendix D.2 in [Gol08].

Part I

Randomness and Advice Complexity Trade-Offs in Online Algorithms

Chapter 2

Steiner Trees

I shall be telling this with a sigh
Somewhere ages and ages hence:
Two roads diverged in a wood, and I-
I took the one less traveled by,
And that has made all the difference.

---

from "The Road Not Taken" by R. Frost

2.1 Introduction

In this chapter, we focus on the advice complexity of the online version of the well-studied minimum Steiner tree problem (ST). In the offline setting, an instance I to ST is a graph $G = (V, E)$ endowed with a weight function on the edges $w : E \to \mathbb{R}^+$ and a set $T \subseteq V$ of vertices called terminals. A subgraph σ of G is a solution to the instance if every pair of terminals is connected in it. The cost of a solution σ, denoted $\text{cost}(\sigma)$, is the sum of the weights of the edges in it, and a solution is optimal if there exists no other solution with smaller cost.

Following previous work of Imase and Waxman [IW91], we consider the following natural online version of the minimum Steiner tree problem. Given a (known) weighted graph G, the terminals appear in a step-wise manner, and the algorithm maintains a subset of the edges as its solution. Upon receiving a new terminal, the algorithm extends the current solution so that the new terminal is connected to the old ones. The entire graph is known in advance, and only the specific subset of terminal vertices (and an ordering on it) is part of the instance.

More formally, given a ground graph G with a weight function w, an instance to $\mathsf{ST}(G, w)$ is an ordered list of vertices called terminals $[v_1, v_2, \ldots, v_N]$, where $v_i \in V$. At time step i, the algorithm receives terminal v_i and extends its current solution by choosing additional edges from G. The augmented solution computed by the algorithm by the end of step i is a solution to the offline problem on G with $\{v_1, \ldots, v_i\}$. As in the offline case, the cost of the solution is the total weight of edges chosen by the algorithm. An instance for $\mathsf{ST}(G, w)$ with N terminals is encoded canonically as a binary string of length $N \cdot \lceil \log(|V|) \rceil$.

An online algorithm with advice for $\mathsf{ST}(G, w)$ is strictly c-competitive using b advice bits if, for every instance, there exists an advice string ϕ such that the total weight of edges chosen by the algorithm during its computation with ϕ is at most c times the weight of the edges of an optimal solution to the offline problem, and at most b bits are read from ϕ. In general, $c = c(\cdot)$ and $b = b(\cdot)$ are function of some parameter of the input, typically the input length.

2.1.1 Contributions of this Chapter

We obtain a complete and exact characterization of the power of advice bits for the online Steiner tree problem.

In Section 2.2, we first give a variant of the greedy algorithm of [AA92] (without advice), which is $O(\log(N))$-competitive on an input with N terminals, and then show that our modified algorithm, which we call terminal-greedy algorithm, is $O(\log(\frac{N}{q}))$-competitive, utilizing an advice of size $q \cdot \log(|V|)$. Informally, the advice we employ is a description of the q most expensive terminals, namely, the q terminals for which the terminal-greedy algorithm added the largest total weight of edges during its execution without advice.

In Section 2.3, we complement our algorithm with a matching lower bound, for the full range of advice bits.We revisit the construction of [IW91], that shows a matching lower bound of $\Omega(\log(N))$ for the competitive ratio in the standard online setting (without advice). Inspired by their construction, we introduce diamond graphs and study their properties. The construction they use can be viewed as a degenerated diamond graph. Our analysis takes a new approach using probabilistic arguments and requires a more general class of graphs in order to handle algorithms that use advice.

For every q such that $0 \leq q \leq N - 1$ and an online algorithm taking advice of size $q \cdot \log(|V|)$, we construct a *different* instance distribution on a suitable diamond graph. We then employ the mechanism developed earlier

in order to show that for this graph there exists an instance for which the algorithm is $\Omega(\log(\frac{N}{q}))$-competitive. Our lower bound here holds already for the unweighted case, where $w(e) = 1$ for every $e \in E$.

We observe that a partial and weaker result of a matching lower bound for some values of advice size $q \log(|V|)$ can be obtained using the original construction presented in [IW91], albeit using a different analysis. We emphasize that our new construction is essential for the proof of a matching lower bound for the full range $0 \leq q \leq N - 1$ of online algorithms using $q \cdot \log(|V|)$ advice bits.

2.1.2 Related Work

Imase and Waxman [IW91] were the first to study the Steiner tree problem in the online setting and showed a tight bound of $\Theta(\log(N))$ for its competitive-ratio. Alon and Azar [AA92] show that almost the same lower bound holds also for the case, where the vertices are points in the Euclidean plane. Berman and Coulston [BC97] and Awerbuch et al. [AAB04] study a generalized version of the problem and related problems. More recently, Garg et al. [GGLS08] considered a stochastic version of the online problem.

2.1.3 Further Notation

For two understood objects a and b, (i.e., instances, paths, etc.) we denote their concatenation by $a \circ b$. For a graph $G = (V, E)$ and two vertices $s, t \in V$, we denote by $s \rightsquigarrow t$ a simple path from s to t in G.

2.2 The Terminal-Greedy Algorithm

In this section we present an $O(\log(\frac{N}{q}))$-competitive algorithm that utilizes $q' \stackrel{\text{def}}{=} q \cdot \log(|V|)$ advice bits, for any $q \in [N - 1]$.

Observe that an advice of size $(N - 1) \log(|V|)$ is always sufficient in order to obtain an optimal solution, since the algorithm is required to make its first decision only upon receiving the second vertex. Therefore, one could canonically encode the rest of the input using $(N - 1) \log(|V|)$ bits.

Recall that the online greedy algorithm that connects the current terminal v_i using the shortest weighted path to a vertex from the current solution is $O(\log(N))$-competitive. Our algorithm is obtained by a modification of the greedy algorithm. Whereas the greedy algorithm connects the next vertex to the current solution by using the shortest path to *any* vertex of the current solution, the terminal-greedy algorithm connects a new vertex using a shortest path to one of the terminals of the current input, ignoring possible shorter paths connecting to some non-terminal vertices already chosen by the solution.

A version of the following lemma was used in the proof of [AA92] for the standard greedy algorithm and still holds for our terminal-greedy algorithm.

Lemma 5. *Let* OptVal *denote the value of the optimal solution to an instance. Let* $k \in \mathbb{N}$*. The number of steps in which the terminal-greedy algorithm (without advice) adds edges of total weight greater than* $2 \cdot \mathsf{OptVal}/k$ *is at most* $k - 1$*.*

Proof: Let S denote the set of such terminals, that is a vertex $v \in S$ if and only if, at the point v presented to the algorithm, the algorithm added a total edge-weight of at least $2 \cdot \mathsf{OptVal}/k$. Observe that in particular, the distance between any two vertices in S is greater than $2 \cdot \mathsf{OptVal}/k$. Now, consider T', an optimal Steiner tree for the vertices in S. Its cost is at most OptVal, since S is a subset of the terminal set of the instance. Let us double the edges in T' and direct them according to some depth first search traversal of T' starting at one of the vertices in S. The depth first search traversal of T' can be decomposed to $|S|$ paths between the vertices of S (according to their order of appearance on the traversal). Now, on the one hand, by the first observation the total edge weight of these paths is greater than $2 \cdot |S| \cdot \mathsf{OptVal}/k$, but on the other hand, their total weight is at most $2 \cdot \mathsf{OptVal}$. It follows that $|S| < k$. ∎

The terminal-greedy algorithm utilizes its advice as a list of q vertices from the instance. Intuitively, the vertices given as advice are the q most expensive ones for the input when given in an online fashion (without advice). The challenge is to show that no further costs are incurred by the algorithm using this approach.

Next, we describe the terminal-greedy algorithm with advice of size $q \cdot \log(|V|)$:[1] When the algorithm receives its first terminal vertex v_1 from the

[1]Fomally, recall that the advice string is infinite, and therefore another $2\log(|V|)$ advice bits are given at the beginning of the input, encoding the value q. In our setting this can be ignored, incurring an additive constant imprecision of at most 1.

instance it computes the optimal (offline) Steiner Tree for the terminal set that consists of the q vertices given as advice along with v_1. Then, it sets the computed tree for this terminal set (which consists of $q+1$ vertices) as the current solution.

For $i \geq 2$, upon receiving a terminal v_i, the algorithm proceeds as follows: If v_i has already appeared in the advice, it does nothing. Otherwise, the algorithm computes the shortest path (in G) from v_i to all the terminals that have previously appeared as part of the instance $(v_1, \ldots, v_{i-1})$ and connects v_i using the shortest path among those $i-1$ paths (and to the lexicographically first in case that there is more than one).

Theorem 6. *Let $1 \leq q \leq N-1$. The terminal-greedy algorithm with $q \cdot \log(|V|)$ advice bits is $O(\log(\frac{N}{q}))$-competitive.*

Proof: Using induction one shows that, in every step, the chosen subgraph is a solution to the current instance. The rest of the proof is concerned with showing the bound on the cost of the algorithm.

We show that, for every $q > 0$, there exists a set of size q such that, for every instance with N terminals with optimal (offline) solution of value OptVal, the solution computed by the algorithm has cost at most $O(\mathsf{OptVal} \cdot \log(\frac{N}{q}))$.

For any $v_i \in \{v_1, \ldots, v_N\}$ we denote by $c(v_i)$ the cost incurred when adding vertex v_i according to the terminal-greedy algorithm (without advice). That is, $c(v_i)$ is the sum of the weights of all the edges chosen at step i in order to connect v_i to the solution. Let us sort the vertices of the instance according to their costs. That is, let $[v'_1, v'_2, \ldots, v'_N]$ be the sorted permutation of $[v_1, \ldots, v_N]$, where $c(v'_1) \geq c(v'_2) \geq \cdots \geq c(v'_N)$.

We claim that the terminal-greedy algorithm with advice $[v'_1, \ldots, v'_q]$ is $\log(\frac{N}{q})$-competitive. Indeed, the tree computed by the algorithm after v_1 is received has cost at most OptVal, as the optimal tree is one possible solution to it. Now, whenever a vertex v_i is received, it behaves exactly[2] as the greedy-terminal algorithm without advice would, and therefore its cost for this vertex is $c(v_i)$.

By Lemma 5, we know that for every $i \in [n]$ it holds that $c(v'_i) \leq (2 \cdot \mathsf{OptVal})/i$ as otherwise, since the v_i's are sorted, we get i vertices that each incurs a cost of more than $(2 \cdot \mathsf{OptVal})/i$. Since $c(v_1) = c(v'_n) = 0$, the total cost of the algorithm is bounded by

[2]Note that in general this does not hold for the 'standard' greedy-algorithm.

$$\begin{aligned}
\mathsf{OptVal} + \sum_{i=q+1}^{N-1} c(v'_i) &\leq \mathsf{OptVal} + 2 \cdot \mathsf{OptVal} \sum_{i=q+1}^{N-1} \frac{1}{i} \\
&= \mathsf{OptVal} \left(1 + 2(\sum_{i=1}^{N-1} \frac{1}{i} - \sum_{i=1}^{q} \frac{1}{i})\right) \\
&< \mathsf{OptVal} \left(1 + \frac{2}{\log(e)} \cdot \log(\frac{N-1}{q}) + \frac{1}{q}\right) \quad ,
\end{aligned}$$

where in the last inequality we used the fact that $\sum_{i=1}^{k} \frac{1}{i} = \ln(k) + \gamma + \frac{1}{2k} \pm o(\frac{1}{k})$, where $\gamma \approx 0.5772$ is the Euler-Mascheroni constant. Finally, recall that a subset of the vertices of size q can be described using $q \cdot \log(|V|)$ bits. The bound follows. ∎

2.3 A Matching Lower Bound

In this section we show a lower bound matching the competitive ratio guarantee of the algorithm presented in Section 2.2. As mentioned, our construction holds already for the unweighted case where $w(e) = 1$, thus we omit w from our notation.

2.3.1 Edge-Efficient Algorithms

It will be useful for us to analyze the performance of algorithms that enjoy a canonical structure and have some guarantees on their behavior. We identify such a class of algorithms next. An online algorithm A for ST is edge-efficient if, for every instance I, when removing any edge from the solution $\mathsf{A}(I)$, the resulting graph is not a solution. That is, removing any edge from $\mathsf{A}(I)$ disconnects two terminals $v, v' \in I$.

The next lemma shows that edge-efficient algorithms are as powerful as general algorithms and therefore we can focus our analysis on them.

Lemma 7. *For every deterministic online algorithm* A *for* ST *there exists an edge-efficient algorithm* A' *such that, for every instance* I*, we have* $\mathrm{cost}(\mathsf{A}'(I)) \leq \mathrm{cost}(\mathsf{A}(I))$.

Proof: Let A be an algorithm for ST. We describe the behavior of the corresponding algorithm A' and denote its current solution by σ'.

The algorithm A' maintains a simulation of the run of A. Each time a new vertex v is received, A' forwards v to A and receives from A an updated solution σ.

Now, A' computes the shortest distances on σ from v to all the vertices of its current solution σ' and chooses a vertex w from σ' that attains the minimal distance. Next, it connects v to w using the corresponding shortest path on σ. That is, it adds to σ' all the vertices and edges from σ along the path.

Using induction, one proves that σ' computed by A' is contained in σ and hence $\mathrm{cost}(\mathsf{A}'(I)) \leq \mathrm{cost}(\mathsf{A}(I))$. Furthermore, in every step, the solution σ' is a tree, where all its leaves are vertices from the instance I and therefore removing any edge from σ' disconnects the solution. ∎

2.3.2 Diamond Graphs and Our Instance Distribution

For vertices s and t and a list of natural numbers $[\ell_1, \ell_2, \ldots, \ell_n]$, we define the diamond graph of level n on vertices s and t, denoted $D_n[\ell_1, \ldots, \ell_n](s,t)$, recursively as follows:

1. The graph $D_0[\,](s,t)$ (of level $n = 0$ with an empty list) consists of the vertices s and t and the single edge (s,t).

2. Given $G(s',t') \stackrel{\text{def}}{=} D_n[\ell_1, \ldots, \ell_n](s',t')$, a diamond graph of level n on vertices s', t', the graph $D_{n+1}[z, \ell_1, \ell_2, \ldots, \ell_n](s,t)$ is constructed as follows: We start with the vertices: s, t and $m_1, \ldots, m_z$. Next, we construct the following $2z$ copies of $G(s',t')$: $G(s,m_1), \ldots, G(s,m_z)$ and $G(m_1,t), \ldots, G(m_z,t)$, where $G(x,y)$ is a copy of the graph $G(s',t')$, where the vertices s' and t' are identified with x and y. Finally, the resulting graph is the union of the $2z$ diamond graphs $G(s,m_1), \ldots, G(s,m_z)$, $G(m_1,t), \ldots, G(m_z,t)$.

We call the parameter ℓ_i the width of level i of the graph, and the vertices $m_1, \ldots, m_{\ell_1}$ the middle vertices of $D_n[\ell_1, \ldots, \ell_n](s,t)$. Note that the graphs in the union are almost disjoint, that is, any two of them share at most one vertex (and no edges).

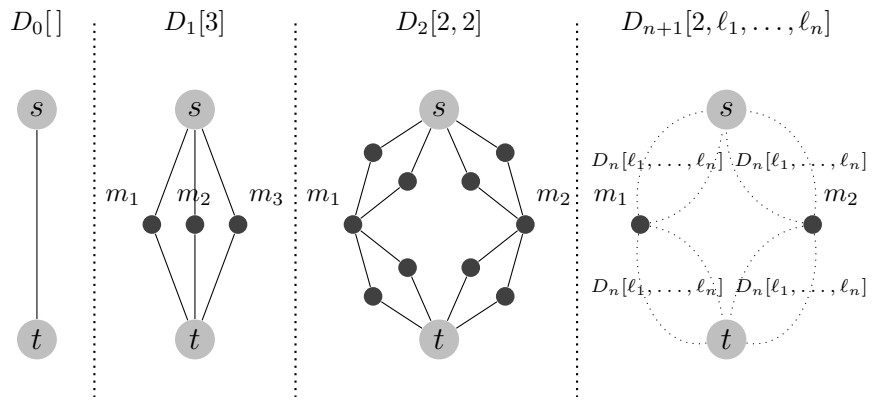

Figure 2.1: Diamond Graphs

For a fixed $n \in \mathbb{N}$ our instance distribution generates simultaneously an instance I that contains $N+1 = 2^n+1$ terminals, and a path P from s to t of length $N = 2^n$, which is an optimal solution to it.[3] The first two vertices are always s and t, and vertices along the path are chosen level by level, where choosing the vertices of level $i+1$ can be thought of as a refinement of the path along the vertices of level i. The idea is that the algorithm has to connect all the level-i vertices before level-$(i+1)$ vertices are revealed. Formally, the instance of $\mathsf{ST}(D_n[\ell_1,\ldots,\ell_n](s,t))$ is generated according to Process 1.

The following propositions follow by simple induction and the definitions of $D_n[\ell_1,\ldots,\ell_n](s,t)$, GenerateInstance and GeneratePath.

Proposition 8. *The graph $D_n[\ell_1,\ldots,\ell_n](s,t)$ contains $2^n \cdot \prod_{i=1}^{n} \ell_i$ edges.*

Proposition 9. *Let $n \geq 1$. A simple path $s \rightsquigarrow t$ on $D_n[\ell_1,\ldots,\ell_n](s,t)$ is of the form $s \rightsquigarrow x \rightsquigarrow t$ for some $x \in \{m_1,\ldots,m_{\ell_1}\}$ and contains exactly 2^n edges.*

Proposition 10. *The path P computed during the execution of* GenerateInstance$(D_n[\ell_1,\ldots,\ell_n](s,t))$ *is a solution to the generated instance that contains exactly 2^n edges.*

Proposition 11. *During the run of* GeneratePath$(D_k[\ell'_1,\ldots,\ell'_k](u,v))$, *when the algorithm adds a vertex x as the next vertex of the instance I*

[3]For simplicity of presentation, we use an instance of $N+1$ instead of N terminals.

Process 1 GenerateInstance

Input: A graph $D_n[\ell_1, \ldots, \ell_n](s, t)$
Output: An instance I of $\mathsf{ST}(D_n[\ell_1, \ldots, \ell_n](s, t))$

1: $I \leftarrow [s]$ ▷ Every instance starts with the vertex s
2: $I \leftarrow I \circ [t]$ ▷ followed by the vertex t.
3: $P \leftarrow \textsc{GeneratePath}(D_n[\ell_1, \ldots, \ell_n](s, t))$
4:
5: **procedure** $\textsc{GeneratePath}(D_k[\ell'_1, \ldots, \ell'_k](u, v))$
6: **if** $k = 0$ **then**
7: **return** $e = (u, v)$
8: **else**
9: Choose $x \xleftarrow{r} \{m_1, \ldots, m_{\ell'_1}\}$ ▷ $m_1, \ldots, m_{\ell'_1}$ are the middle vertices of $D_k[\ell'_1, \ldots, \ell'_k](u, v)$
10: $I \leftarrow I \circ [x]$
11: $P_1 \leftarrow \textsc{GeneratePath}(D_{k-1}[\ell'_2, \ldots, \ell'_k](u, x))$
12: $P_2 \leftarrow \textsc{GeneratePath}(D_{k-1}[\ell'_2, \ldots, \ell'_k](x, v))$
13: **return** $P_1 \circ P_2$
14: **end if**
15: **end procedure**

(Line 10), both u and v have already appeared in I and no other vertex from $D_k[\ell'_1, \ldots, \ell'_k](u, v)$ is contained in I.

Lemma 12. *Let A be an edge-efficient algorithm, and consider an execution of* $\textsc{GenerateInstance}(D_n[\ell_1, \ldots, \ell_n](s, t))$. *The number of edges added to the solution by A during every call to* $\textsc{GeneratePath}(D_k[\ell'_1, \ldots, \ell'_k](u, v))$ *is at least* $W_k \stackrel{\text{def}}{=} \sum_{i=1}^{k} \left(\frac{2^k}{2^i} \sum_{j=1}^{2^{i-1}} X_{i,j} \right)$, *where the $X_{i,j}$'s are independent Bernoulli random variables with* $\Pr[X_{i,j} = 0] = 1/\ell'_i = 1 - \Pr[X_{i,j} = 1]$.

Before proving the lemma, we prove the following proposition on the structure of the current solution restricted to the subgraph $D_k[\ell'_1, \ldots, \ell'_k](u, v)$ when $\textsc{GeneratePath}(u, v)$ is called.

Proposition 13. *Let $k \in \{1, \ldots, n\}$ and let A be an edge-efficient algorithm. Consider an execution of* $\textsc{GeneratePath}(D_n[\ell_1, \ldots, \ell_n](s, t))$. *Whenever a call* $\textsc{GeneratePath}(D_k[\ell'_1, \ldots, \ell'_k](u, v))$ *is made, either (1) the current solution chosen by A restricted to the subgraph $D_k[\ell'_1, \ldots, \ell'_k](u, v)$ contains no edges, or, (2) $D_k[\ell'_1, \ldots, \ell'_k](u, v)$ contains a simple path of the form $u \rightsquigarrow y \rightsquigarrow v$ for some $y \in \{m_1, \ldots, m_{\ell'_1}\}$, where $m_1, \ldots, m_{\ell'_1}$ are the middle vertices $D_k[\ell'_1, \ldots, \ell'_k](u, v)$, and no other edges.*

Proof: By Proposition 11, we know that the vertices u and v have already appeared in the instance, and therefore they are connected in the current solution, and, in particular, by some simple path $u \rightsquigarrow v$. Consider the first edge (u, z) of this path. If z is contained in $D_k[\ell'_1, \ldots, \ell'_k](u, v)$, then, since the only way to reach a vertex outside of $D_k[\ell'_1, \ldots, \ell'_k](u, v)$ is through the vertices u and v, the entire path is contained in $D_k[\ell'_1, \ldots, \ell'_k](u, v)$. Conversely, if z is not contained in $D_k[\ell'_1, \ldots, \ell'_k](u, v)$, by the same argument, it follows that the path $u \rightsquigarrow v$ does not contain any inner vertex of $D_k[\ell'_1, \ldots, \ell'_k](u, v)$.

We argue that in both cases no other edges of the current solution are incident to $D_k[\ell'_1, \ldots, \ell'_k](u, v)$. Assume the contrary, and let $e \in D_k[\ell'_1, \ldots, \ell'_k](u, v)$ be such an edge (not in $u \rightsquigarrow v$ in the first case and any edge in $D_k[\ell'_1, \ldots, \ell'_k](u, v)$ in the second case).

Observe that, since e is an internal edge of $D_k[\ell'_1, \ldots, \ell'_k](u, v)$ not on the path $u \rightsquigarrow v$ and since by Proposition 11 at this point no other internal vertex is chosen to the current instance, the vertices of the current instance remain connected after removing e. This contradicts the edge-efficiency property of the current solution chosen by A. ∎

Proof: [of Lemma 12] We use induction on k, the parameter of the diamond subgraph. For $k = 0$, the claim holds trivially since $W_0 = 0$ and at least zero edges are added. Let $k > 0$, and assume that the claim holds for all $k' < k$. Let GeneratePath($D_k[\ell'_1, \ldots, \ell'_k](u, v)$) be a call made during the execution of GenerateInstance($D_n[\ell_1, \ldots, \ell_n](s, t)$). By Proposition 13, the solution chosen limited to $D_k[\ell'_1, \ldots, \ell'_k](u, v)$ either (1) has no edges, or, (2) has exactly one simple path between u and v, which by Proposition 9 has the form $u \rightsquigarrow y \rightsquigarrow v$, for $y \in \{m_1, \ldots, m_{\ell'_1}\}$. Without loss of generality, we assume that the path is of the form $u \rightsquigarrow m_1 \rightsquigarrow v$. In the first case, after lines 9 and 10, the algorithm connects the vertex x to the graph, which must be via the vertex u or v, in which case $\frac{2^k}{2}$ edges are added. In the second case, with probability $1 - 1/\ell'_1$ the vertex x is chosen from $m_2, \ldots, m_{\ell'}$, in which case as before, it must be connected using a path from u or v, which adds $\frac{2^k}{2}$ edges.

We conclude that the number of edges added to the solution due to the choice of the vertex x is at least $\frac{2^k}{2} X_{1,1}$, where $X_{1,1}$ is distributed according to $\Pr[X_{1,1} = 0] = 1/\ell'_1 = 1 - \Pr[X_{1,1} = 1]$.

Additionally, using the inductive hypothesis, the algorithm adds $W'_{k-1} = \sum_{i=1}^{k-1} \left(\frac{2^{k-1}}{2^i} \sum_{j=1}^{2^{i-1}} X'_{i,j} \right)$ and $W''_{k-1} = \sum_{i=1}^{k-1} \left(\frac{2^{k-1}}{2^i} \sum_{j=1}^{2^{i-1}} X''_{i,j} \right)$ edges during GeneratePath($D_{k-1}[\ell'_2, \ldots, \ell'_k](s, x)$) and

GeneratePath($D_{k-1}[\ell'_2, \ldots, \ell'_k](x,t)$), respectively, where $X'_{i,j}$ and $X''_{i,j}$ are Bernoulli random variables distributed according to $\Pr[X'_{i,j}=0] = \Pr[X''_{i,j}=0] = 1/\ell'_{i+1} = 1 - \Pr[X''_{i,j}=1] = 1 - \Pr[X'_{i,j}=1]$.

Moreover, since the random choices of GeneratePath are independent, we have that the Bernoulli random variables are independent. Setting $X_{i+1,j} \stackrel{\text{def}}{=} X'_{i,j}$ and $X_{i+1,2^{i-1}+j} \stackrel{\text{def}}{=} X''_{i,j}$ for all $i \in \{1,\ldots,n\}$ and $j \in \{1,\ldots,2^{i-1}\}$, we obtain that during the execution of GeneratePath the algorithm adds at least $\frac{2^k}{2} X_{1,1} + W'_{k-1} + W''_{k-1} = \sum_{i=1}^{k} \left(\frac{2^k}{2^i} \sum_{j=1}^{2^{i-1}} X_{i,j} \right)$ edges to the solution. The lemma is proved. ∎

Corollary 14. *Any deterministic algorithm* A *for* ST*, when given an instance generated by* GeneratePath($D_n[\ell_1, \ldots, \ell_n](s,t)$)*, outputs a solution that contains at least*

$$\sum_{i=1}^{\log(N)} \left(\frac{N}{2^i} \sum_{j=1}^{2^{i-1}} X_{i,j} \right)$$

edges, where the $X_{i,j}$'s are as in Lemma 12.

Proof: By Proposition 7, we may assume that A is an edge-efficient algorithm. The corollary follows by Lemma 12. ∎

We refer to an edge added due to some $X_{i,j} = 1$ as an edge of level i and say that in this case the algorithm made a wrong choice on $X_{i,j}$. Indeed, in this case it was possible to connect some vertices u and v through a middle vertex m such that the algorithm would not have had to add edges due to $X_{i,j}$.

2.3.3 Deriving the Lower Bound

In this section, we show that for every algorithm with advice size $q \cdot \log(|V|)$ the terminal-greedy algorithm is best possible.

The input distribution we use is a diamond graph with parameters that depend on the advice length of the specific algorithm it seeks to fail. Consider an algorithm taking $N \cdot 2^{-j(N)} \cdot \log(|V|)$ advice bits, where $\log(N) \geq j(N) \geq 0$.

We can assume that $\log(N) \geq j(N) > 10$ and, furthermore, that $j(N)$ is an even integer number. The first assumption is trivial to satisfy, since every algorithm is at most strictly 1-competitive and so for a constant $j(N)$ the

asymptotic bound already holds. The second assumption incurs an additive term of 2 (recall that the bound we show is logarithmic). Therefore, both assumptions are made without loss of generality.

Set $j'(N) \stackrel{\text{def}}{=} \frac{j(N)}{2}$ and consider the diamond graph with $\log(N)$ levels $D_n[\ell_1, \ldots, \ell_n](s,t)$, where the first $\log(N) - j'(N)$ levels are of width 2 and the last $j'(N)$ levels are of width N^2. That is, $\ell_1 = \cdots = \ell_{\log(N)-j'(N)} = 2$ and $\ell_{\log(N)-j'(N)+1} = \cdots = \ell_{\log(N)} = N^2$. For the rest of this section we refer to this graph as G.

We can show that for every online algorithm with $q \cdot \log(|V|)$ advice bits there exists an input on which it does not perform better than $\Omega(\log(\frac{N}{q}))$ compared to an optimal offline solution:

Theorem 15. *Let A be an online algorithm for* ST *taking $q' \stackrel{\text{def}}{=} q \cdot \log(|V|)$ advice bits, where $q \stackrel{\text{def}}{=} N \cdot 2^{-j(N)}$ and $\log(N) \geq j(N) \geq 0$. Then A has a competitive ratio of at least $\Omega(\log(\frac{N}{q}))$.*

We present an overview of the proof. Recall that for a fixed advice string $\phi \in \{0,1\}^{q'}$, the algorithm A "hard-wired" with ϕ (denoted A^ϕ) is a deterministic online algorithm and therefore Corollary 14 establishes that on a random instance of GenerateInstance($D_n[\ell_1, \ldots, \ell_n](s,t)$) it chooses at least W_n edges.

We show that an instance chosen by GenerateInstance contains roughly $N \cdot \log(\frac{N}{q})$ edges with very high probability, and then use the union bound to show that there exists an instance that makes all of the $2^{q'}$ algorithms choose this number of edges.

Using the machinery developed for general diamond graphs and the properties of GenerateInstance we show that, by our choice of $j'(N)$, it holds that $\log(|V|)$ is not too large, and for each of the last $j'(N)$ levels of the graph, a fixed deterministic algorithm chooses a linear (in N) number of edges with probability roughly $2^{-q'} = 2^{-q \cdot \log(|V|)}$.

Finally, we use the probabilistic method and show that there exists an instance on G, such that every A^ϕ chooses many edges on every level τ of the last $j'(N)$ levels in G.

Proof: Using Proposition 8 we have that the number of edges in G is $2^n \cdot 2^{(n-j'(N))} \cdot (N^2)^{j'(N)} < 4^n \cdot (N^2)^{j'(N)}$. Since the number of vertices in a graph is at most twice the number of its edges, we obtain that $|V|$, the number of vertices in G, is at most $2 \cdot 4^n \cdot (N^2)^{j'(N)}$, and therefore

$\log(|V|) < 4 \cdot \log(N) \cdot j'(N)$. Therefore, we can bound the advice size by

$$q' = 2^{n-j(N)} \cdot \log(|V|) < 2^{n-j(N)} \cdot 4 \cdot \log(N) \cdot j'(N) = 4 \cdot 2^{n-2 \cdot j'(N)} \cdot n \cdot j'(N) \ .$$

By Lemma 12 and Corollary 14, for every level τ, where $n - j'(N) < \tau \leq n$, the probability that an edge-efficient deterministic algorithm is correct on at least $\frac{2^{\tau-1}}{4}$ of its choices for level τ (i.e., at least this number of $X'_{\tau,j}s$ are 0) can be computed as

$$\begin{aligned}
\Pr\left[\exists S \subset \left[2^{\tau-1}\right] : |S| = \frac{2^{\tau-1}}{4} \wedge \forall p \in S : X_{\tau,p} = 0\right] & \\
&< \binom{2^{\tau-1}}{\frac{2^{\tau-1}}{4}} \cdot \left(\frac{1}{N^2}\right)^{\frac{2^{\tau-1}}{4}} \\
&\leq \left(2^{\tau-1}\right)^{\frac{2^{\tau-1}}{4}} \left(\frac{1}{N^2}\right)^{\frac{2^{\tau-1}}{4}} \\
&= \left(\frac{2^{\tau-1}}{2^{2n}}\right)^{\frac{2^{\tau-1}}{4}} \\
&\leq \left(\frac{1}{2^n}\right)^{\frac{2^{\tau-1}}{4}} \\
&= 2^{-\left(\frac{n \cdot 2^{\tau-1}}{4}\right)} \\
&\leq 2^{-\left(\frac{n \cdot 2^{n-j'(N)}}{4}\right)} \ .
\end{aligned}$$

Next we apply the union bound twice: The probability p that there exists a level $n - j'(N) < \tau \leq n$ for which one of the $2^{q'}$ deterministic algorithms makes more than $\frac{2^{\tau-1}}{4}$ correct choices can be bounded as follows:

$$\begin{aligned}
p &< 2^{q'} \cdot j'(N) \cdot 2^{-\left(\frac{n \cdot 2^{n-j'(N)}}{4}\right)} && (2.1) \\
&< 2^{\left(4 \cdot 2^{(n-2 \cdot j'(N))} \cdot n \cdot j'(N)\right)} \cdot 2^{\log(j'(N))} \cdot 2^{-\left(\frac{n \cdot 2^{n-j'(N)}}{4}\right)} && (2.2) \\
&< 2^{\left(5 \cdot 2^{(n-2 \cdot j'(N))} \cdot n \cdot j'(N)\right)} \cdot 2^{-\left(\frac{n \cdot 2^{n-j'(N)}}{4}\right)} && (2.3) \\
&= 2^{\left(\left(n \cdot 2^{n-j'(N)}\right)\left(5 \cdot 2^{-j'(N)} \cdot j'(N) - \frac{1}{4}\right)\right)} < 1 && (2.4)
\end{aligned}$$

In turn, observe that this implies that there exists a fixed instance I' such that the algorithm A, for every choice of advice of length q', and for every level τ in the range, makes at least $\frac{3 \cdot 2^{\tau-1}}{4}$ incorrect choices, each of which results in an addition of $\frac{N}{2^\tau}$ edges by the algorithm. Therefore, for this instance, the algorithm chooses a solution that contains at least

$$\sum_{\tau=\log(N)-j'(N)+1}^{\log(N)} \frac{N}{2^\tau} \cdot \frac{3 \cdot 2^{\tau-1}}{4} = \frac{3}{8} \cdot N \cdot j'(N) \in \Omega(N \cdot j(N)) = \Omega\left(N \cdot \log(\frac{N}{q})\right)$$

edges.

On the other hand, recall that, by Proposition 10, since I' is just one of the possible instances generated by GenerateInstance, there exists a solution that consists of N edges. The lower bound of $\Omega(\log(\frac{N}{q}))$ on the competitive ratio follows. ∎

Chapter 3

Disjoint Path Allocation

3.1 Introduction

An important example of an online problem which models a real-life scenario is call admission in communication networks, where a central authority has to admit or reject requests for communication between certain pairs of nodes in the network. We shall focus on the special case called the Linear Disjoint Path Allocation Problem.

Here, the communication network is simply modeled by a path of length L, where the $L + 1$ vertices correspond to the nodes of the network which might want to communicate with each other using the links modeled by the edges of the path. We assume that a communication request between two vertices is issued at some point in time, but is of unbounded duration. Moreover, we assume that any link of the path is only capable of serving one communication request. Thus, admitting a request between two vertices on the path prevents any vertex in between from participating in any communication. The goal is to admit as many requests as possible.

More formally, the online linear disjoint path allocation problem (LDPA) is the following maximization problem. At the first time step, the number L is revealed, describing the path $P = (v_0, \ldots, v_L)$. In every successive time step, a subpath of P, called a request, is presented to the algorithm. An algorithm maintains a solution to the problem as follows: At the beginning, the solution is empty. At every step, if the edges of the path presented do not intersect with those of *any* of the subpaths chosen to the current solution, the algorithm may choose the path to the current solution. A path that was not chosen by the end of step i may not be added at a later step.

In this case, the cost of a solution is the number of subpaths chosen to it and it is optimal if no other solution is of a larger cost.

At times, when the input sequence I is clear from the context, we omit the explicit dependence on it and write Opt and OptVal instead of $\mathsf{Opt}(I)$ and $\mathrm{cost}(\mathsf{Opt}(I))$, respectively.

The disjoint path allocation problem can be analyzed with respect to two different parameters: the number n of requested subpaths and the length L of the path. For the parameter n, the randomized setting was analyzed in [BEY98] and the advice complexity in [BKK+09].

Lastly, we recall that an $O(\log(L))$-competitive algorithm for LDPA can be implemented using $\lceil \log\log(L) \rceil$ random bits, and this matches the lower bound on the competitive ratio that can be obtained by any randomized online algorithm for the problem (Theorems 13.7 and 13.8 in [BEY98]).

3.1.1 Contributions of this Chapter

We continue the study of the exact power of randomness in online algorithms (focusing on LDPA) and establish a trade-off between the number of random bits available to the online algorithm and its competitive ratio.

We prove that any number $b \leq \log\log L$ of advice bits can be replaced by the same number of random bits while achieving (almost) the same competitive ratio in expectation. Thus, in some sense, a small number of random bits is as powerful as a small number of advice bits for this problem.

Specifically, we obtain an online algorithm that uses only b random bits and enjoys a competitive ratio of $(L^{\frac{1}{2^b}} \cdot 2^{b+1})$ for any $b \in \{0, \ldots, \lceil \log\log(L) \rceil\}$. Indeed, for $b = 0$, we obtain the greedy algorithm (without advice), and for $b = \lceil \log\log(L) \rceil$, we obtain the randomized algorithm from [BEY98]. We mention that in light of the results presented in [BBF+14], this is essentially best possible, even when using advice bits for this range of the parameter b, recalling that advice bits are always at least as strong as bits.

3.2 A Barely Random Algorithm

3.2.1 Partitioning Edges and Intervals

In this section, we identify the edge (v_t, v_{t+1}) with its right vertex and call it edge $t+1$. That is, a request $[i, j]$ contains the edges $\{i+1, \ldots, j\}$. Requests

intersect if they contain a mutual edge. A set of requests covers a request if every edge in the request is contained in some request from the set.

It will be useful to think of an edge t as the bit string of length $\log(L+1)$ that denotes its binary expansion. For simplicity of presentation, for the rest of this section, we assume that $L = 2^\ell - 1$ for some integer ℓ, but our results here hold for any L (just think of the natural embedding of the path to the first L vertices in a path of length $L' - 1$, where L' is the smallest power of 2 larger than L).

We partition the edges into ℓ levels, where an edge e belongs to level $\lambda(e)$, where $\lambda\colon E \to \mathbb{N}$ is given by $\lambda(e) \stackrel{\text{def}}{=} \max\{\, t : 2^t \text{ divides } e \,\}$. Alternatively, $\lambda(e)$ is the largest t such that, in the binary representation of e, the t rightmost bits of e are zero. That is, the edge $10^{\ell-1}$ is the only edge of level $\ell - 1$, the edges $10^{\ell-2}$ and $110^{\ell-2}$ are the only edges of level $\ell - 2$, and in general, there are exactly $2^{\ell-j-1}$ edges of level j.

It will be useful to consider a coarser partition to blocks of B levels. To this end, for every $B \in \mathbb{N}^+$, we define the B-block of an edge $\lambda_B\colon E \to \mathbb{N}$ by $\lambda_B(e) \stackrel{\text{def}}{=} \lfloor \frac{\lambda(e)}{B} \rfloor$. It is immediate that $\lambda_B(e) = i$ if and only if $\lambda(e) \in \{iB, iB+1, \ldots, (i+1)B-1\}$. We extend λ (respectively, λ_B) to any request r by setting $\lambda(r) = \max_{e \in r} \lambda(e)$ (resp., $\lambda_B(r) = \max_{e \in r} \lambda_B(e)$). We call a request a level-t (resp., B-block i) request if $\lambda(r) = t$ (resp., $\lambda_B(r) = i$).

3.2.2 The B-Block-Greedy Algorithm

Let Opt be an optimal solution for an LDPA instance with value OptVal. We denote by o_t the number of requests in Opt for which $\lambda(r) = t$. Similarly, we set $o'_i \stackrel{\text{def}}{=} \sum_{j=0}^{B-1} o_{iB+j}$, the number of requests in Opt for which $\lambda_B(r) = i$. It holds that $\mathsf{OptVal} = \sum_{i=0}^{\lceil \ell/B \rceil - 1} o'_i = \sum_{t=0}^{\ell-1} o_t$. We shall make use of the following proposition and the corollary that follows it.

Proposition 16. *If a request contains two different edges of level t, then it contains an edge of level at least $t+1$.*

Proof: Let $e < e'$ be two edges of level t in r. It holds that $e = x10^t$ and $e' = y10^t$ for some $x, y \in \{0,1\}^{\ell-t}$. Now, observe that the edge $e + 10^t$ is of level at least $t+1$, and is contained in the request since $e < e + 10^t \leq e'$. ∎

Corollary 17. *Every request r has exactly one edge of level $\lambda(r)$.*

Proof: Towards contradiction, let $e, e' \in r$ be two edges of maximal level $\lambda(r)$. By Proposition 16, r contains an edge of level at least $\lambda(r)+1$, which contradicts the maximality of the levels of e and e'. ∎

Corollary 17 asserts that every request has exactly one edge of maximal level. We call this edge the level-edge of the request. Additionally, for any edge e, any solution to an LDPA instance contains at most one request with e as its level edge (this is true for any edge, and in particular for the level edges).

Proposition 18. *Let r be a level-t request. Then, for any solution of a* LDPA *instance and any $t' \geq t$, the solution contains at most one level-t' request that intersects with r.*

Proof: Suppose that some solution contains two level-t' requests, r' and r'', that intersect with r. Let e' and e'' be their level-edges of level t', respectively. It must hold that $e' \neq e''$, as otherwise r' and r'' overlap and cannot both be contained in the solution, and, without loss of generality, we assume that $e' < e''$. Observe that, by Proposition 16, the request $[e'-1, e'']$ contains an edge f of level at least $t'+1$. Now, since r intersects with both requests, the request $[e'-1, e'']$ is covered by the union of the three requests, and therefore f is contained in at least one of r, r' and r'', which is a contradiction to the fact that all three requests are of level at most t'. ∎

We are now ready to present and analyze the i-th B-block greedy algorithm for LDPA. The algorithm B-Block-Greedy$_i$ takes all the requests offered from B-block i as long as they do not intersect with requests already chosen to the current solution.

Proposition 19. *For any instance of* LDPA*, B-Block-Greedy$_i$ chooses at least $2^{-B} \cdot o'_i$ requests.*

Proof: We show that, for every edge that the algorithm chooses, at most 2^B requests chosen by Opt from block i intersect with it. Let r be a request of level $\lambda(r) = t = iB + j$ for some $j \in \{0, \ldots, B-1\}$ chosen by B-Block-Greedy$_i$ and let e, an edge of level t, be its level-edge. That is, the binary representation of e is of the form $e = x10^j0^{iB}$ for some $x \in \{0,1\}^{\ell-1-j-iB}$.

Applying Proposition 18 to Opt yields the following.

Claim 20. *for any level $t' \geq t$,* Opt *contains at most one level-t' request that intersects with r.*

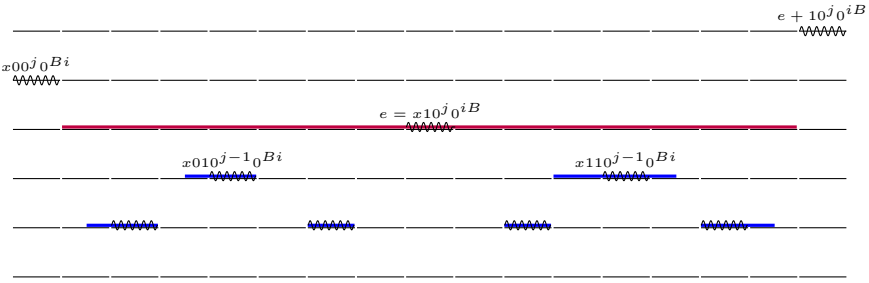

Figure 3.1: A block-i request of level $t = iB + j$. The request on line 3 is the longest possible request with level-edge e. The edge e is of the form $x10^j 0^{Bi}$ for some $x \in \{0,1\}^{\ell - Bi - j - 1}$. The blue requests are chosen by Opt.

Moreover we need the following claim.

Claim 21. *For any level $t' < t$,* Opt *contains at most $2^{(t-t')}$ level-t' requests that intersect with r.*

The second claim can be shown as follows. Let r' be a level-t' request that intersects with r, and let e' be its level-edge. We argue that e' is of the form $xy10^{t'}$ for some $y \in \{0,1\}^{t-t'}$ and x as before. Assuming otherwise, without loss of generality that $e' < x0^{t-t'}10^{t'}$ (the case where $e' > x1^{t-t'}10^{t'}$ is handled similarly), it holds that e' is of the form $x'10^{t'}$ as it is a level-t' edge, and therefore $x' < x0^{t-t'}$. Now, consider the request $[e'-1, e]$. It is covered by the requests r' and r (as they intersect and contain both edges), and additionally, contains the edge $x0^{t+1}$, since $e' < x0^{t+1} < e$. Therefore, one of the requests (without loss of generality r') contains the edge $x0^{t+1}$. This immediately leads to a contradiction, since it now holds that $\lambda(r') \geq \lambda(x0^{t+1}) \geq t+1 > t' = \lambda(r')$. The claim follows using Corollary 17 and the fact that there are $2^{(t-t')}$ edges of the form $xy10^{t'}$, where $y \in \{0,1\}^{t-t'}$.

Combining both claims, we observe that the bound on the total number of requests in B-block i that intersect with r and are contained in Opt is $(B - j + 1) + \sum_{k=1}^{j-1} 2^{(j-k)}$, and is maximized when the request is chosen from the uppermost level of the block, in which case it holds that at most $1 + \sum_{j=1}^{B-1} 2^j \leq 2^B$ requests in Opt intersect with r. It follows that the algorithm chooses at least $2^{-B} \cdot o'_i$ requests. ∎

3.2.3 A Competitive Algorithm using Blocks of Size B

Let $B \in \{1, \ldots, \ell\}$. We define the (randomized) B-Block-Greedy algorithm as the algorithm that chooses uniformly at random a block $i \xleftarrow{r} \{0, \ldots, \lceil \ell/B \rceil - 1\}$, and behaves according to B-Block-Greedy$_i$. The worst-case expected value of this algorithm is

$$\begin{aligned}
&\mathop{\mathbf{E}}_{i \xleftarrow{r} \{0,\ldots,\lceil \ell/B \rceil - 1\}} \left[\text{cost}(B\text{-Block-Greedy}_i)\right] \\
&\qquad \geq \sum_{i=0}^{\lceil \ell/B \rceil - 1} \frac{2^{-B}}{\lceil \ell/B \rceil} \cdot o'_i \\
&\qquad = \frac{2^{-B}}{\lceil \ell/B \rceil} \sum_{t=0}^{\ell-1} o_t \\
&\qquad = \frac{2^{-B}}{\lceil \ell/B \rceil} \cdot \mathsf{OptVal}.
\end{aligned}$$

Put differently, we obtain a $(\lceil \ell/B \rceil \cdot 2^B)$-competitive algorithm. Note that choosing a random B-block out of the $\lceil \ell/B \rceil$ possible blocks is the only random choice of the algorithm and can be done using $\lceil \log(\lceil \ell/B \rceil) \rceil$ random bits.[1]

3.2.4 A Competitive Algorithm Using b Random Bits

So far our analysis was made in terms of the block size B. Next, we present our main theorem for this section, which delineates explicitly the competitive ratio obtained as a function of the number of available random bits b.

Theorem 22. *The randomized B-Block-Greedy algorithm that uses b random bits is $(L^{\frac{1}{2^b}} \cdot 2^{b+1})$-competitive.*

Proof: Observe that the smaller the block size B is, the better is the competitive ratio of the algorithm. Using b bits it is possible to randomly choose of one of 2^b blocks. Therefore, we can use any $B \in \mathbb{N}$ such that

[1] In fact, in general, an implementation that uses only $\lceil \log(\lceil \ell/B \rceil) \rceil$ bits obtains a $(\lceil \ell/B \rceil \cdot 2^{B+1}\rceil$-competitive ratio (that is, it incurs an additional factor of 2). However, whenever $\lceil \ell/B \rceil$ is a power of two (as is the case in the next subsection) we can save this factor while still using $\lceil \log(\lceil \ell/B \rceil) \rceil$ random bits.

$2^b \cdot B \geq \ell$ holds. Setting a block size of $B = \lceil \ell/2^b \rceil$ levels in $(\lceil \ell/B \rceil \cdot 2^B)$, we obtain a competitive ratio of $2^{\lceil \ell/2^b \rceil} \cdot \lceil \ell/\lceil \frac{\ell}{2^b} \rceil \rceil \leq L^{1/2^b} \cdot 2^{b+1}$ using b random bits. ∎

Part II

Constructions and Separations of Cryptographic Primitives from One-Way Functions

Chapter 4

A Universal One-Way Hash Function from a One-Way Function with Regularity Assumptions

> Now will I sing to my wellbeloved a song of my beloved touching his vineyard. My wellbeloved hath a vineyard in a very fruitful hill: And he fenced it, and gathered out the stones thereof, and planted it with the choicest vine, and built a tower in the midst of it, and also made a winepress therein: and he looked that it should bring forth grapes, and it brought forth wild grapes.
>
> Isaiah 5 : 1-2

4.1 Introduction

A main task in cryptographic research is to construct a (strong) cryptographic primitive P from a (weaker) cryptographic primitive Q, for example to construct a pseudo-random generator from a one-way function (OWF). This chapter is concerned with constructing a universal one-way hash function (UOWHF), a fundamental cryptographic primitive, from a OWF.

The term "construct" means that one gives an efficient reduction of the problem of breaking the underlying primitive Q to the problem of breaking

the constructed primitive P. For two primitives P and Q, the most basic question is whether P can be constructed *in principle* from Q, meaning that the construction and the reduction must be efficient (i.e., polynomial-time) and that the reduction translates a non-negligible probability of breaking P into a non-negligible probability of breaking Q.

The principle possibility of constructing a UOWHF from a OWF was proved by Rompel [Rom90], using a highly inefficient construction and reduction. When trying to improve the construction, one can choose two orthogonal routes. Either one improves the construction for a general OWF, or one makes specific assumptions about the OWF allowing for special-purpose constructions that do not necessarily work in general, and can hence be more efficient. Of course, a key issue is how restrictive or how plausible the assumption one has to make is.

The best known *general* construction of a universal one-way hash function from any one-way function $f : \{0,1\}^n \to \{0,1\}^m$, due to Haitner et al. [HHR+10], has output length $\tilde{O}(n^7)$ and $\tilde{O}(n^5)$ for the uniform and non-uniform cases, respectively. The best known special-purpose construction is due to Naor and Yung [NY89] and makes a single call to f (per argument to the constructed UOWHF), and the output length is linear in n, but the assumption one needs to make is that f is *injective*, which is a very strong assumption.

In this chapter we investigate the middle grounds between completely general constructions and those requiring such a very specific assumption. Concretely, we investigate the trade-off between the regularity assumption for f and the efficiency of the construction. The regularity is characterized by how concentrated the *preimage size spectrum*, the random variable $|f^{-1}(f(X))|$ corresponding to the preimage size of the function value $f(X)$ for a uniformly random argument X, is. For injective functions, the preimage size spectrum is constant 1. Prior to our work, we do not know of any specific construction for a function which is anywhere between regular and arbitrary.

We relate the assumptions made about the spectrum of f to the efficiency of the overall construction. Qualitatively speaking, the more is assumed about the regularity of f, the more efficient is the resulting construction.

4.1.1 Contributions of this Chapter

A first result on the way to fully utilizing an assumption about the regularity of a function is an almost optimal construction of a universal one-way hash

function from a regular (or almost regular) one-way function. Recall that a function is 2^r-regular if for every image there are 2^r preimages.

Following previous work, for simplicity of presentation, we assume that for a one-way function f the input length n is the security parameter. For this case, we get a construction with output length and key length $O(n \cdot \alpha(n) \cdot \log(n))$, where the construction makes $O(\alpha(n) \cdot \log(n))$ invocations to f for any super-constant function $\alpha(n)$. This improves on [SY90] by a factor of $\log(n)$ (see Section 4.1.2 for comparison with previous work).

We introduce a natural relaxation of the notion of regularity:

Definition 23 (roughly-regular function)**.** *A function $f : \{0,1\}^n \to \{0,1\}^m$ is called (r,s)-*roughly-regular*, if for every x in $\{0,1\}^n$ it holds that $|f^{-1}(f(x))|$ lies in the interval $[r, rs]$. A family of functions $f = \{f_n\}_{n>0}$ is called (r,s)-*roughly-regular*, where $(r,s) = (r(n), s(n))$, if for every n it holds that f_n is $(r(n), s(n))$-roughly regular.*

We call r and s the *regularity* and the *roughness* parameters of f, respectively. Indeed, whenever the roughness parameter is trivial, that is, $s(n) = 1$ for all n, this definition coincides with the standard definition of an r-regular function. This definition, we argue, is both intuitive and quantifies the irregularity of a function.

In Section 4.4 we utilize the ideas developed in Section 3 and improve on [HHR$^+$10] with the most general version (Theorem 37). We establish a trade-off between the regularity assumption made about the underlying one-way function and the overall efficiency of the construction. When f is a $(2^{r(n)}, 2^{s(n)})$-roughly regular one-way function, we show a construction with output length and key length of $\tilde{O}(n \cdot s^4)$ for the non-uniform model and of $\tilde{O}(n \cdot s^6)$ for the uniform model. Indeed, our construction ties up both ends of the existing constructions: When s is constant, we get an almost linear construction, and when $s = O(n)$ our construction matches that of [HHR$^+$10].

The analysis of the construction presented in Section 4.3 improves by a factor of $O(\log^2(n))$ on the construction presented in Section 4.4 when instantiated with a 2^r-regular function. A very recent subsequent work by Yu et al. [YGLW14] improves on the result presented in Section 4.3, where they give a construction that makes only a super-constant number of calls.

4.1.2 Related Work

Inaccessible Entropy.

Our work uses the framework of [HHR$^+$10] for constructing UOWHFs from OWFs using the notion of inaccessible entropy. Inaccessible entropy was first introduced in [HRVW09] and along with work done in [HNO$^+$09] and [HRV10], it completes the construction of the fundamental cryptographic primitives: universal one-way hash functions, pseudo-random generators and commitment schemes using this notion.

A Regularity-Efficiency Trade-Off for the Construction of a UOWHF.

In [SY90] it was first shown how to construct a UOWHF for the almost-regular case. Our construction achieves the same query complexity to the underlying one-way function ($O(\alpha(n) \cdot \log(n))$ calls), but is superior in two aspects: It makes its queries to the underlying one-way function in a non-adaptive manner, and our resulting primitive has an output (and seed) length of $n \cdot log(n) \cdot \alpha(n)$ whereas the construction from [SY90] has an additional $\log(n)$ factor.

While for the almost-regular case the improvement is not dramatic, we believe that our analysis, which extends the approach suggested in [HHR$^+$10], sheds more light on what is achieved at each step. The way the almost-regularity property of the underlying one-way function is utilized later allows to generalize it to any level of regularity. This is in contrast to the construction in [SY90] which is more ad-hoc.

4.2 Preliminaries

4.2.1 Notations and Basics

For a function $f : X \to Y$, we define the preimage spectrum function $\pi_f : X \to \mathbb{N}$, where $\pi_f(x) = |f^{-1}(f(x))|$. For understood $Y_1 \times \cdots \times Y_n$ we denote by $\phi_i : Y_1 \times \cdots \times Y_n \to Y_i$ the projection onto the i'th component. We extend this to a set $S \subseteq Y_1 \times \cdots \times Y_n$ with $\phi_i(S) \stackrel{\text{def}}{=} \{\phi_i(s) : s \in S\}$. A non-decreasing function $f : \mathbb{N} \to \mathbb{N}$ is called super-constant if for all $c \in \mathbb{N}$ there exists an $n \in \mathbb{N}$, such that $f(n) > c$.

The next lemma shows that ignoring an unlikely event of a random variable that takes a value in some limited range, does not change much its expected value.

The standard proof is omitted.

Lemma 24. *Let X be a random variable with $\mathrm{Supp}(X) \subset [0, l]$ and A an event that happens with probability at most ϵ. Then:*

$$\left|\mathbf{E}[X] - \mathbf{E}\left[X|\overline{A}\right]\right| \leq 2 \cdot l \cdot \epsilon \ . \tag{4.1}$$

As noted, we focus on families of functions where the input domain parameter n equals to the security parameter, i.e., $n(\rho) = \rho$, in which case we parameterize the family by n. Additionally, we slightly abuse notation when referring to a function $f : \{0,1\}^n \to \{0,1\}^{m(n)}$, where formally $f = \{f_n : \{0,1\}^n \to \{0,1\}^{m(n)}\}_{n\in\mathbb{N}}$ is a parametrized family of functions, and often we omit the security parameter when referring to f_n or other parametrized values.

4.2.2 Entropy Measures

For a random variable X and $x \in \mathrm{Supp}(X)$ the point-wise entropy of X is $H_X(x) \stackrel{\text{def}}{=} -\log(\Pr[X = x])$. The Shannon entropy $H(X)$ and min-entropy $H_\infty(X)$ of X are defined as:

$$H(X) \stackrel{\text{def}}{=} \mathbf{E}[H_X(X)] \ , \qquad H_\infty(X) \stackrel{\text{def}}{=} -\log\left(\max_{x\in\ \mathrm{Supp}(X)} \Pr[X = x]\right) \ .$$

These measures extend naturally to the case of a joint distribution of two random variables X, Y. Namely, the conditional point-wise entropy for $(x, y) \in \mathrm{Supp}(X, Y)$ is $H_{X|Y}(x, y) \stackrel{\text{def}}{=} -\log(\Pr[X = x|Y = y])$ and the conditional Shannon entropy is

$$H(X|Y) = \mathop{\mathbf{E}}_{(x,y)\xleftarrow{r}(X,Y)}[H_{X|Y}(x, y)] = \mathop{\mathbf{E}}_{y\xleftarrow{r}Y}[H(X|Y = y)] = H(X, Y) - H(Y).$$

The next definition measures the average and absolute guarantees as for the preimage-size of f in terms of entropy bits.

Definition 25 (preimage entropy measures)**.** *For a function $f : \{0,1\}^n \to \{0,1\}^{m(n)}$, define its* real preimage-entropy *as $H_p(f) \stackrel{\text{def}}{=} H(X|f(X))$, where X is uniformly distributed on $\{0,1\}^n$. f has* min-preimage-entropy *at least $k = k(n)$ (and denote this by $H_{p,\min}(f) \geq k$), if there is a negligible function $\epsilon = \epsilon(n)$ such that*

$$\Pr_{x \stackrel{r}{\leftarrow} \{0,1\}^n} [H_{X|f(X)}(x, f(x)) \geq k] \geq 1 - \epsilon.$$

As the argument X in the definition is uniform, we have that for all x it holds that $H_{X|f(X)}(x, f(x)) = \log(\pi_f(x))$.

4.2.3 Collision Finders and Accessible Entropy

Definition 25 captures the average and absolute preimage set size guarantees for f. Clearly, when f is shrinking it has high preimage-entropy. Recall that our goal is to build a universal one-way hash function, namely, a shrinking function for which there exist many preimages, but at the same time any efficient algorithm, when given an x, cannot compute a different preimage from $f^{-1}(f(x))$.

Definition 26 (f-collision-finder)**.** *Let $f : \{0,1\}^n \to \{0,1\}^{m(n)}$ be a function. An f-collision-finder is a randomized algorithm A such that $A(x) \in f^{-1}(f(x))$ for every $x \in \{0,1\}^n$.*

The requirement that $A(x)$ outputs a preimage of $f(x)$ can be made without loss of generality, as every algorithm A can be changed to one that outputs x whenever $A(x) \notin f^{-1}(f(x))$.

Using the notion of an f-collision-finder, one can define a computational analog of the definitions of real- and min-preimage-entropy of f. The analogous definitions capture the maximal, average, and absolute size of the preimage sets that are accessible to any efficient algorithm.

Definition 27. *A function $f : \{0,1\}^n \to \{0,1\}^{m(n)}$ has* accessible max-preimage-entropy *at most $k = k(n)$ if there exists a family of sets $\{S_x\}_{x \in \{0,1\}^n}$ such that for any efficient randomized f-collision-finder A, there exists a negligible function $\epsilon = \epsilon(n)$ such that for all sufficiently large n:*

1. $\Pr_{x \stackrel{r}{\leftarrow} \{0,1\}^n, A} [A(x) \in S_x] \geq 1 - \epsilon.$

2. $\log(|S_x|) \leq k$ *for all* x*.*

Definition 28. *A function* $f : \{0,1\}^n \to \{0,1\}^{m(n)}$ *has* accessible average max-preimage-entropy *at most* $k = k(n)$ *if it satisfies Definition* 27 *where instead of (2.) we have:*

2. $\mathop{\mathbf{E}}_{x \xleftarrow{r} \{0,1\}^n} [\log(|S_x|)] \leq k.$

We stress that these two definitions[1] are different from the classical definitions of Shannon entropy. As they capture the inputs accessible only to *efficient* algorithms, both definitions only bound from above the performance of such algorithms. Specifically, for an arbitrary function, we do not know how to compute exactly (as in the standard definition of entropy) these bounds. Nevertheless, as we see next, these bounds are a useful tool (see also [HRVW09]). We use the notation $H^{\text{eff}}_{p,\max}(f) \leq k$ and $H^{\text{eff}}_{p,\text{avg}-\max}(f) \leq k$ to denote that the corresponding bound holds.

The next two definitions are used to distinguish between two types of 'entropy gaps':

Definition 29. *A function* $f : \{0,1\}^n \to \{0,1\}^{m(n)}$ *has an* average inaccessible preimage-entropy gap $\Delta = \Delta(n)$*, if there exists some* $k = k(n)$ *such that:*

$$H^{\text{eff}}_{p,\text{avg}-\max}(f) \leq k \leq k + \Delta \leq H_p(f) \ . \tag{4.2}$$

That is, there is a gap of Δ between its average accessible max-preimage-entropy and its preimage-entropy. At times we will refer to this gap as an average entropy gap or a weak type of gap.

Definition 30. *A function* $f : \{0,1\}^n \to \{0,1\}^{m(n)}$ *has an* absolute inaccessible preimage-entropy gap $\Delta = \Delta(n)$*, if there exists some* $k = k(n)$ *such that:*

$$H^{\text{eff}}_{p,\max}(f) \leq k \leq k + \Delta \leq H_{p,\min}(f) \ . \tag{4.3}$$

At times we will refer to this gap as an absolute or strong gap.

[1] In fact, one may consider a weaker notion of algorithm-dependent accessible max-preimage-entropy and algorithm-dependent accessible average max-preimage-entropy where the sets $\{S_x\}$ may also depend on the algorithm. Such a definition would only require that for every algorithm there exist sets $\{S_{A,x}\}$. This weaker variant of Definitions 27 and 28 is enough for the purpose of constructing a universal one-way hash function and potentially may be easier to satisfy. In this work we do not make use of the weaker definition.

An important observation is that UOWHFs are just length-decreasing functions with accessible max-preimage-entropy 0, and an appropriate absolute entropy gap. Haitner et al. observed that it is possible to achieve a noticeable gap of inaccessible entropy as an intermediate step, and then amplify it and transform it into a UOWHF.

4.2.4 Entropy Measures for t-fold Parallel Repetitions

For a function $f : X \to Y$ we define its t-fold parallel repetition $f^t : X^t \to Y^t$ as $f^t(x_1, \ldots, x_t) = (f(x_1), \ldots, f(x_t))$. It is well-known that using the definition of conditional entropy, properties of $\log(\dot{})$ and noting that choosing a random $x^t \in X^t$ can be done by t independent choices of x,

$$H_p(f^t) = H(X_1, \ldots, X_t | f(X_1), \ldots, f(X_t)) = H(X_1|f(X_1)) + \ldots \\ +H(X_t|f(X_t)) = t \cdot H_p(f) \ . \quad (4.4)$$

The corresponding computational bound is given by the following claim and its corollary. Namely, the accessible preimages of the t-fold repetition of f come from the product set of the accessible preimages set of f:

Lemma 31. *Let $f : X \to Y$ with accessible max-preimage-entropy at most $k(n)$, with sets S_x (as in Definition 27). Then for $t = poly(n)$ any efficient f^t-collision-finder A' outputs a collision (except with negligible probability) from the set $S_{x^t} \stackrel{\text{def}}{=} S_{x_1} \times \cdots \times S_{x_t}$.*

Proof: Let A' be an f^t-collision-finder algorithm with probability ϵ to output a collision $x'_1, \ldots, x'_t$ outside of S_{x^t}. Observe that this implies that for a randomly chosen coordinate $i \stackrel{r}{\leftarrow} [t]$ it holds that $\Pr[\phi_I(f^t(X^t)) \notin S_{\phi_I(X^t)}] \geq \epsilon/t$. This calls for the following f-collision-finder A: on input x choose uniformly at random a location i from $[t]$ and uniformly at random inputs $x_1, x_2, \ldots, x_{i-1}, x_{i+1}, \ldots, x_t$ from X. Set $x_i = x$ and return $\phi_i(A'(x_1, \ldots, x_t))$. It follows that A outputs a collision for f outside of S_x with probability greater than ϵ/t. The lemma follows. ∎

Using linearity of expectation, the union bound, Definitions 27 and 28, and the fact that $\log(|S_{x^t}|) = \Sigma_{i=1}^{t} \log(|S_{x_i}|)$, we get:

Corollary 32.

1. *If $H^{\text{eff}}_{p,\max}(f) \leq k$ then $H^{\text{eff}}_{p,\max}(f^t) \leq t \cdot k$.*
2. *If $H^{\text{eff}}_{p,\text{avg}-\max}(f) \leq k$ then $H^{\text{eff}}_{p,\text{avg}-\max}(f^t) \leq t \cdot k$.*

4.2.5 An Overview of the Construction of Haitner et al.

The construction consists of two independent parts. First they show how to get a function with a noticeable gap of average inaccessible entropy from any one-way function. Specifically, they show that a prefix of a random length of a three-wise independent hashing of the output already has some weak form of an average entropy gap. Namely, on average over the inputs to the new construction, there is a noticeable gap of $\Delta = \Omega(\log n/n)$ between the real preimage-entropy and the average accessible max-preimage-entropy.

The second part of the construction starts with any function with some noticeable gap Δ and shows how to obtain a UOWHF. This is achieved using the following steps:

1. Gap amplification and transformation of an average type gap into an absolute type of gap.
2. Entropy reduction.
3. Output length reduction.
4. Random inputs collision-resistance to a UOWHF.
5. Removing the non-uniformity.

The composition of Steps 2 through 5 of their construction[2] is summarized in the following theorem, which we later use in a black-box manner:

Theorem 33.

1. *There exists an explicit black-box construction taking parameters a function $\psi = \{\psi_n\}_{n\in\mathbb{N}}$, where $\psi_n : \{0,1\}^{\lambda(n)} \to \{0,1\}^{m(n)}$, and a number $\tau = \tau(n)$ such that if $\Delta \stackrel{\text{def}}{=} H_{p,\min}(\psi_n) - H^{\text{eff}}_{p,\max}(\psi_n) \in \omega(\log(n))$, and $\tau \in [H_{p,\min}(\psi_n), H_{p,\min}(\psi_n) + \frac{\Delta}{2}]$, the construction implements a UOWHF with output length and key length $O(\lambda(n))$ making a single call to ψ.*
2. *Moreover, for all efficiently computable $l = l(n)$ there exists an explicit black-box construction taking parameters ψ (as before) and sets of numbers $\tau = \tau(n) = \{\tau_{n,i}\}_{i=1}^{l(n)}$, such that if one of $\{(\psi_n, \tau_{n,i})\}_{i=1}^{l(n)}$ satisfies the condition of part (1.), the construction implements a UOWHF with output length $O(\lambda(n) \cdot l(n))$ and key length of $O(\lambda(n) \cdot l(n) \cdot \log(l(n)))$.*

Here, τ is a "good enough" estimation of the real-min preimage entropy of f.

[2] Lemmas $5.3 - 5.4, 5.6$ in [HHR$^+$10].

4.3 UOWHF from a 2^r-Regular OWF

Let $f : \{0,1\}^n \to \{0,1\}^m$ be a 2^r-regular one-way function. Our construction also works in two steps: First we obtain an entropy gap of $O(\log(n))$ applying f only once and use a variant of the Naor-Yung construction. Next, we show that the type of gap that we get by our first step is almost of the required absolute type. Namely, the average entropy gap is essentially concentrated on a smaller interval of size almost $O(\log(n))$, and in this case the structured gap can be transformed to an absolute type of gap and increased to $\omega(\log n)$ via taking only a super-logarithmic number of independent samples. The main result of this section is:

Theorem 34. *Let $f : \{0,1\}^n \to \{0,1\}^{m(n)}$ be a 2^r-regular one-way function, where $r = r(n)$ is efficiently computable. Then there exists an explicit black-box construction of a universal one-way hash function based on f with output length and key length of $O(n \cdot \log(n) \cdot \alpha(n))$ that makes $O(\alpha(n) \cdot \log(n))$ calls to f for any super-constant function $\alpha(n)$.*

4.3.1 Inaccessible Entropy from 2^r-Regular One-Way Functions

In this case f has exactly 2^{n-r} different images. If we randomly distribute the images among b buckets, we expect to have roughly $\frac{2^{n-r}}{b}$ images in each bucket. Consider the composed function, $F(x,g) = (g(f(x)), g)$ where g is the description of a three-wise independent hash-function from some family $\mathcal{G}$. We show that an appropriate choice of the family $\mathcal{G}$ allows us to reduce the preimage inaccessibility of F to the hardness of the underlying function f.

For injective one-way functions this was already observed in [NY89]. The difference is that for an injective f, the resulting F is already a universal one-way hash function, whereas in the case where f is regular we get:

Lemma 35. *Let $f : \{0,1\}^n \to \{0,1\}^{m(n)}$ be a 2^r-regular one-way function, where $r = r(n)$ is efficiently computable. Let $d > 0$ and let $\mathcal{G} = \mathcal{G}(n) \stackrel{\text{def}}{=} \mathcal{G}_n^{(n-r)-4d\log(n)}$ be a family of constructible three-wise independent hash functions. Then the function $F : \{0,1\}^n \times \mathcal{G} \to \{0,1\}^{(n-r)-4d\log(n)} \times \mathcal{G}$ given by: $F(x,g) = (g(f(x)), g)$ satisfies the following properties:*

1. $H_p(F) \geq r + 3d\log(n)$.
2. $H^{\text{eff}}_{p,\max}(F) \leq r$.

Proof:

1. Recall that when the input is chosen uniformly at random from the input space the preimage-entropy of F is just the expected log-value of the size of the preimage set. We first compute the expected number of preimages for a fixed x with some random g from $\mathcal{G}$. Since $F(x,g)$ already determines g it follows that any potential preimage must have the same g component.

For all fixed x, g we have the set equality:

$$F^{-1}(F(x,g)) = \bigcup_{y':g(f(x))=g(y')} \left(f^{-1}(y') \times \{g\}\right) \tag{4.5}$$

and the union is over disjoint sets. We get that:

$$\begin{aligned} \pi_F(x,g) &= \pi_f(x) + \sum_{y' \neq f(x)} \mathbb{1}_{g(f(x))=g(y')} \cdot |f^{-1}(y')| \\ &= 2^r \cdot \left(1 + \sum_{y' \neq f(x)} \mathbb{1}_{g(f(x))=g(y')}\right) \ , \end{aligned} \tag{4.6}$$

where due to the regularity it holds that $\pi_f(x) = |f^{-1}(y)| = 2^r$.

Now we observe that for every fixed $x \in \{0,1\}^n$ and $y' \neq f(x)$

$$\mathop{\mathbf{E}}_{g \xleftarrow{r} \mathcal{G}} \left[\mathbb{1}_{g(f(x))=g(y')}\right] = 2^{-(n-r-4d\log(n))} \ , \tag{4.7}$$

where the equality is due to due to the pair-wise independence of $\mathcal{G}$.

Using (4.6), (4.7) and linearity of expectation we have that

$$\mathop{\mathbf{E}}_{g \xleftarrow{r} \mathcal{G}} \left[\pi_F(x,g)\right] = 2^r \cdot \left(1 + \sum_{y' \neq f(x)} \mathop{\mathbf{E}}_{g \xleftarrow{r} \mathcal{G}} \left[\mathbb{1}_{g(f(x))=g(y')}\right]\right) > 2^{r+4d\cdot\log(n)} \ , \tag{4.8}$$

where again due to the regulariry the summation is over $2^{n-r} - 1$ indicators.

Furthermore, as the family is three-wise independent, we also have that for different $f(x), y', y''$ it holds that the random variables $\mathbb{1}_{g(f(x))=g(y')}$ and $\mathbb{1}_{g(f(x))=g(y'')}$ are independent (and in particular, uncorrelated) and therefore

$$\mathop{\mathbf{V}}_{g\xleftarrow{r}\mathcal{G}}[\pi_F(x,g)] = (2^r)^2 \cdot \sum_{y'\neq f(x)} \mathop{\mathbf{V}}_{g\xleftarrow{r}\mathcal{G}}\left[\mathbb{1}_{g(f(x))=g(y')}\right] \leq \left(2^{r+2\cdot d\cdot\log(n)}\right)^2 \ . \quad (4.9)$$

where the equality holds for the sum of uncorrelated random variables and the the inequality holds as for all indicator random variables $\mathbf{V}[\mathbb{1}_A] \leq \mathbf{E}[\mathbb{1}_A]$ and using (4.7). Now, the Chebyshev inequality establishes that for all $\alpha > 0$:

$$\Pr_{g\xleftarrow{r}\mathcal{G}}\left[\left|\pi_F(x,g) - \mathop{\mathbf{E}}_{g\xleftarrow{r}\mathcal{G}}[\pi_F(x,g)]\right| > \alpha\cdot 2^{r+2\cdot d\cdot\log(n)}\right] < \frac{1}{\alpha^2} \ . \quad (4.10)$$

Whenever the event in (4.10) does not happen plugging (4.8) we obtain

$$\pi_F(x,G) \geq 2^{r+4\cdot d\cdot\log(n)} - \alpha\cdot 2^{r+2\cdot d\cdot\log(n)} \geq 2^{r+3.5\cdot d\cdot\log(n)} \ ,$$

for all fixed α and sufficiently large n.

Finally, recall that due to the regularity we always have $\pi_F(x,g) \geq 2^r$ and so using conditional expectation on the event from (4.10) with $\alpha = 5$ and plugging (4.8) we obtain:

$$\mathop{\mathbf{E}}_{g\xleftarrow{r}\mathcal{G}}[\log(\pi_F(x,g))] > r + \frac{24}{25}\cdot(3.5\cdot d\cdot\log(n)) \geq r + 3\cdot d\cdot\log(n) \ . \quad (4.11)$$

As this holds for every fixed x, it also holds for a random one, and we are done.

2. We show that any efficient algorithm that finds a collision for a random input (X,G) outside of $f^{-1}(X)\times\{G\}$ leads to one that inverts f. Let A_F be an F-collision-finder. We denote the randomness taken by A_F explicitly (as an additional argument) by r, and for some fixed randomness r and a fixed input x, denote by $\epsilon_{x,r} \stackrel{\text{def}}{=} \Pr_{g\xleftarrow{r}\mathcal{G}}[A_F(x,g,r) \notin f^{-1}(f(x))\times\{g\}]$.

We show[3] that the algorithm B inverts $y = f(x')$, where $x' \xleftarrow{r} \{0,1\}^n$ uniformly at random, with probability at least ϵ/n^{-4d}.

[3] In similar manner to [NY89] and [HHR+10].

Algorithm $B(y)$ (on input $y \in \{0,1\}^n$)

. Choose x uniformly at random from $\{0,1\}^n$.
. Choose randomness r for A_F uniformly at random.
. Choose g uniformly at random subject to $g(f(x)) = g(y)$.
. **return** $\phi_1(A_F(x,g,r))$.

It holds that

$$\begin{aligned}
\epsilon_{x,r} &= \sum_{y \neq f(x)} \Pr_{g \xleftarrow{r} \mathcal{G}} \left[\phi_1(A_F(x,g,r)) \in f^{-1}(y)\right] \qquad (4.12)\\
&= \sum_{y \neq f(x)} \Pr_{g \xleftarrow{r} \mathcal{G}} \left[\phi_1(A_F(x,g,r)) \in f^{-1}(y) \mid g(f(x)) = g(y)\right] \\
&\qquad \cdot \Pr_{g \xleftarrow{r} \mathcal{G}} \left[g(f(x)) = g(y)\right] \\
&= \sum_{y \neq f(x)} \Pr_{g \xleftarrow{r} \mathcal{G}} \left[\phi_1(A_F(x,g,r)) \in f^{-1}(y) \mid g(f(x)) = g(y)\right] \\
&\qquad \cdot 2^{-(n-r-4d\log(n))} \ .
\end{aligned}$$

It follows that conditioned on the random choices $X = x$ and $R = r$ of the algorithm B, for X' uniform on $\{0,1\}^n$ and $Y = f(X')$ we obtain:

$$\begin{aligned}
& \Pr_{x' \xleftarrow{r} \{0,1\}^n} [B(f(x')) \in f^{-1}(f(x')) \mid X = x \wedge R = r] \\
= & \Pr_{y \xleftarrow{r} f(X')} [B(y) \in f^{-1}(y) \mid X = x \wedge R = r] \\
\geq & \sum_{y \neq f(x)} \Pr[Y = y] \cdot \Pr_{g \xleftarrow{r} \mathcal{G}} \left[\phi_1(A_F(x,g,r)) \in f^{-1}(y) \mid g(f(x)) = g(y)\right] \\
\geq & \ 2^{-4d\log(n)} \cdot \epsilon_{x,r} \ . \qquad (4.13)
\end{aligned}$$

As X and R are chosen uniformly at random, and by the fact that $\epsilon = \mathbf{E}[\epsilon_{X,R}]$ we get that the algorithm inverts f with probability at least $n^{-d} \cdot \epsilon$. Thus we have shown that A_F's output on input (x,g) is limited to $f^{-1}(f(x)) \times \{g\}$. By the regularity of f we have that its size is exactly 2^r. Thus F has accessible max-preimage-entropy at most r. ∎

We next show that by a more careful analysis of the amplification results in an almost-linear construction.

4.3.2 Amplifying the Entropy Gap and Converting Average to Absolute Entropy Gaps

Lemma 36 (Fast gap amplification and real- to min- preimage-entropy conversion)**.** *Let f and F be as in Lemma 35, F^t be the t-fold application of F and $\alpha(n)$ be any super-constant function. Then for $t = \alpha(n) \cdot \log(n)$, F^t has a strong inaccessible entropy gap of $\alpha(n) \cdot \log^2(n)$. Moreover, the following entropy-gap holds:*

1. $H_{p,\min}(F^t) \geq t(r + 2d\log(n))$.
2. $H^{\text{eff}}_{p,\max}(F^t) \leq t \cdot r$.

Proof:

1. We first show that by Markov's inequality, the probability that the point-wise entropy exceeds its expected value by more than $\alpha(n) \cdot \log(n)$ bits is negligible. Specifically, from Equation (4.8) in the analysis of Lemma 35 we find that the expected value of the preimage size of inputs to F is at most $2^{r+4d\log(n)+1}$. Markov's inequality asserts that the probability we get an input with more than $2^{\alpha(n)\cdot\log(n)} \cdot 2^{r+4d\log(n)+1}$ preimages is at most negligible, and let us denote this event by A. Whenever this event does not happen, the point-wise real-entropy of the preimage set is limited to an interval of size $O(\alpha(n) \cdot \log(n))$, as we always have a at least r bits of point-wise entropy due to the regularity of f. I.e.,

$$r < \log(\pi_F(X,G)) \,|\, \bar{A} < r + (\alpha(n) + 4d)\log(n) + 1 \ .$$

By Lemma 24, we known that

$$\begin{aligned}\mathbf{E}[\log(\pi_F(X,G)) \,|\, \bar{A}] &> \mathbf{E}[\log(\pi_F(X,G))] - \mathtt{negl}(n) \\ &\geq r + 3d\log(n) - \mathtt{negl}(n) \\ &\geq r + 2.75d\log(n) \ .\end{aligned}$$

Now, the Hoeffding bound for t independent samples from the probability space $(X,G) \,|\, \bar{A}$ (with $k = 0.75d \cdot \log n \cdot t$) asserts that a super-logarithmic number of repetitions suffice to bound from below the min-preimage-entropy of the t-fold application of F. Using the union bound we have that for a polynomial number of repetitions, the probability that A happens in any of the repetitions is negligible, and find that

$$\Pr_{(x,g)^t \xleftarrow{r} (\{0,1\}^n \times \mathcal{G})^t} \left[\log\left(\pi_{F^t}((x,g)^t)\right) < t(r + 2d\log(n))\right] \leq \exp\left(\frac{-\Omega(t)}{\alpha^2(n)}\right) + \mathtt{negl}(n). \quad (4.14)$$

The choice of $t = O(\alpha^3(n) \cdot \log(n))$ ensures that this happens with negligible probability.

2. This is just the t-fold accessible max-preimage-entropy we get by the second part of Lemma 35 and Corollary 32. ∎

Proof: [Theorem 34] By Lemma 36, F^t already has the required strong type of entropy gap between its accessible max-preimage-entropy and its real min-preimage-entropy. Moreover, it tells us exactly where this gap is (there are at most $t \cdot r$ bits of accessible max-preimage-entropy, and $\tau = t(r + 2\log n)$ is a good estimation of $H_{p,\min}F^t$. Now, note that if $\alpha(n)$ is a super-constant function, then so is $\alpha'(n) = \alpha^{1/3}(n)$. Finally, utilizing Theorem 33 yields a UOWHF with output length and key length $O(\alpha(n) \cdot \log(n) \cdot n)$ making $O(\alpha(n)\log(n))$ calls. ∎

4.4 A Construction from a OWF with Arbitrary Regularity Assumptions

The main theorem proved in this section is:

Theorem 37. *Let $f : \{0,1\}^n \to \{0,1\}^{m(n)}$ be a $(2^r, 2^s)$-roughly-regular one-way function, where $r = r(n)$ and $s = s(n)$ are efficiently computable. Then there exists an explicit construction of a UOWHF with output length and key length of $\tilde{O}(n \cdot s^6(n))$ (respectively, $\tilde{O}(n \cdot s^4(n))$) in the uniform (resp., non-uniform) model.*

4.4.1 $\Omega(\log(n)/s(n))$ Bits of Average Inaccessible Entropy

Haitner et al. showed that for a general one-way function f, a random truncation of a hashing of $f(x)$ using a three-wise independent family of hash

functions yields an average entropy gap of $\Omega(\log(n)/n)$ entropy bits. We observe that a modification of their first step achieves an average inaccessible entropy gap of $\Omega(\log(n)/s(n))$ bits from any $(2^{r(n)}, 2^{s(n)})$-roughly-regular one-way function.

The idea is to divide the images $f(x)$ (and respectively, the inputs x) into buckets, such that every bucket contains images with roughly the same number of preimages. Let $d > 0$. We set $m \stackrel{\text{def}}{=} s(n)/d\log(n)$ and $\mathcal{J} \stackrel{\text{def}}{=} \{j_0, \ldots, j_{m-1}\}$, where $j_i \stackrel{\text{def}}{=} n - r(n) - s(n) + i \cdot d\log(n)$, and show that truncating the output of the application of a three-wise independent hashing of $f(x)$ to the random length $J - d\log n$ yields a function with the required gap. Recall that

$$H_{f(X)}(f(x)) \in \left[j_i, j_{i+1}\right) \iff \pi_f(x) \in \left(2^{r+s-(i+1)(d\log(n))}, 2^{r+s-i(d\log(n))}\right] .$$

Let us denote $q_i \stackrel{\text{def}}{=} \Pr\left[H_{f(X)}(f(X)) \in [j_i, j_{i+1})\right]$ for $i \in \{0, \ldots, q-2\}$ and $q_{m-1} \stackrel{\text{def}}{=} \Pr\left[H_{f(X)}(f(X)) \in [j_{m-1}, j_m]\right]$.

By the roughly-regularity assumption about f, it holds that $\sum_{i=1}^{m} q_j = 1$. Now, we set $\mathcal{G} \stackrel{\text{def}}{=} \mathcal{G}_n^n$, a family of three-wise independent hash functions, $\mathcal{X} \stackrel{\text{def}}{=} \{0,1\}^n$ and define

$$F : \mathcal{X} \times \mathcal{G} \times \mathcal{J} \to \mathcal{X} \times \mathcal{G} \times \mathcal{J}$$

as

$$F(x, g, j) = (g(f(x))_{1,\ldots,j-d\log n} \| 0^{n-j+d\log n}, g, j) \ ,$$

where we denote the domain and range of F by $\mathcal{Z} \stackrel{\text{def}}{=} \mathcal{X} \times \mathcal{G} \times J$. Note that without loss of generality, we implicitly assumed that $r + s < n - d\log(n)$. It is easy to check that any one-way function must satisfy this, as otherwise, the algorithm that guesses a random preimage on input y, inverts f with non-negligible probability.

Lemma 38. *The function F as defined above has an average preimage-entropy gap of $\Omega(s(n)/log(n))$ bits.*

Proof: Recall that our goal in this step is to achieve an average inaccessible entropy gap of $\Omega(\log(n)/s(n))$ bits. That is, we need to show that for each $z = (x, g, j_i)$ there exists a set S_z, such that: (1) any efficient collision-finder outputs an element of S_z (except for an event that happens with negligible probability) and (2) $\mathbf{E}_{z \stackrel{r}{\leftarrow} \mathcal{Z}}[\log(\pi_F(z)) - \log(|S_z|)] \in \Omega(\log(n)/s(n))$.

In a similar manner to the regular case, the set of inputs accessible by an efficient algorithm is limited only to those with relatively few images, where "few" corresponds to the length of the random truncation. Essentially, we show that when we hash to length j_i, any preimages an efficient algorithm finds are either already preimages of $f(x)$ (we refer to these as 'trivial' collisions) [4] or stem from some non-trivial collision, that is $F(x', g, j_i) = F(x, g, j_i)$ but $f(x) \neq f(x')$. For the latter, we further distinguish between those x that have significantly fewer preimages than expected for a random function with output length j_i, and the rest. More precisely, we consider those preimages $z' = (x', g, j_i)$ for which $\pi_f(x') \leq 2^{n-j_{i+1}}$ and call these 'j_{i+1}-light' preimages of $f(x)$. The remaining 'heavy' collisions stem from inputs z' for which $\pi_f(x') > 2^{n-j_{i+1}}$. We define:

$$T_z \stackrel{\text{def}}{=} T_{(x,g,j_i)} = f^{-1}(f(x)) \times \{g\} \times \{j\},$$

$$L_z \stackrel{\text{def}}{=} L_{(x,g,j_i)} = \{x' \in \{0,1\}^n \mid g(f(x))_{1,\ldots,j_i - d\log n} = g(f(x'))_{1,\ldots,j_i - d\log n} \wedge H_{f(X)}(f(x')) \geq j_{i+1} \wedge x' \notin f^{-1}(f(x))\} \times \{g\} \times \{j\}$$

and

$$H_z \stackrel{\text{def}}{=} H_{(x,g,j_i)} = \{x' \in \{0,1\}^n | g(f(x))_{1,\ldots,j_i - d\log n} = g(f(x'))_{1,\ldots,j_i - d\log n} \wedge H_{f(X)}(f(x')) < j_{i+1} \wedge x' \notin f^{-1}(f(x))\} \times \{g\} \times \{j\} \ ,$$

where T, L and H stand for 'trivial', 'light' and 'heavy', respectively. It follows that for every z,

$$F^{-1}(F(z)) = T_z \cup L_z \cup H_z \ , \tag{4.15}$$

where the union is over disjoint sets.

The rest of the proof is involved with proving that indeed the only accessible sets to any efficient algorithm are $T_z \cup L_z$, and that they constitute a large fraction of the preimage set $F^{-1}(F(z))$. The analysis follows the construction from [HHR$^+$10] and is brought for completeness in Section 4.6. ∎

4.4.2 Faster Amplification of the Inaccessible Entropy Gap of F

Our goal in this section is to amplify the entropy gap of F from the previous section. We show how to construct a function F' with $\omega(\log(n))$ bits of

[4] Note that the definition of a one-way function does not rule out the possibility that given a preimage it is difficult to compute other preimages from $f^{-1}(f(x))$.

inaccessible entropy with an absolute type of gap.

Haitner et al. [HHR+10] assert that independent repetitions of F achieve both these goals. They show that $\tilde{O}(n^4)$ repetitions are enough for getting this gap from an arbitrary one-way function. We are able to utilize the information about the underlying f (and in turn about that of F) and get a faster convergence, using the roughly-regularity assumption.

Set $\Delta \stackrel{\text{def}}{=} (c \cdot \log(n)/s(n))$ as the entropy gap of F, where c is the constant corresponding to the Ω notation, and fix k, such that F has preimage-entropy $H_p(F) = k + \Delta$. Lemma 38 asserts that $H^{\text{eff}}_{p,\text{avg}-\max}(F) \leq k$. Using (4.4) and Corollary 32 we know that for the t-fold parallel repetition of F it holds that

$$H_p(F^t) = t \cdot (k + \Delta) \ , \tag{4.16}$$

$$H^{\text{eff}}_{p,\text{avg}-\max}(F^t) \leq t \cdot k \ . \tag{4.17}$$

Thus for F^t we obtain an average entropy gap of $t \cdot \Delta$ bits.

Using the analysis of Lemma 38 and Lemma 31 we get that for an input $z^t = (z_1, \ldots, z_t)$ to F^t, the only accessible inputs to F^t are those that are contained in $S_z = (T_{z_1} \cup L_{z_1}) \times \cdots \times (T_{z_t} \cup L_{z_t})$, and that the set of preimages of z^t is just $F^{t^{-1}}(F^t(z^t)) = (T_{z_1} \cup L_{z_1} \cup H_{z_1}) \times \cdots \times (T_{z_t} \cup L_{z_t} \cup H_{z_t})$, except for an event B_1 that occurs with negligible probability. Next, we would like to apply the Hoeffding bound to get the required gap. Similarly to Lemma 36, we show that although for some inputs the preimage size of F may be very large (a priori there may be inputs with up to 2^n preimages, but not more, since $F(x, g, j_i)$ determines (g, j_i) uniquely as part of its output), this is not likely. First observe that $\log(\pi_F(z)) \in [r, n]$ for all z. This is due to the fact that every image of $f(x)$ contains at least $2^{r(n)}$ preimages. We show that we can bound this also from above: except with negligible probability we have that for any super-constant function $\alpha(n)$: $\log(|T_z \cup L_z|) \leq \log(\pi_F(z)) < r(n) + s(n) + d\log(n) + \alpha(n)\log(n)$. Consider $\pi_F(Z)$ for a uniformly chosen random input $Z = (X, G, J)$. This value is maximized for $J = j_0$ because of the inclusion $\phi_1(F^{-1}(F(x, g, j_i'))) \subset \phi_1(F^{-1}(F(x, g, j_i)))$ for $j_i \leq j_i'$. It follows that in order to bound $\mathbf{E}\left[\pi_F(X, G, J)\right]$ it is sufficient to bound $\mathbf{E}\left[\pi_F(X, G, j_0)\right]$.

As in Lemma 35, using the three-wise independence of $\mathcal{G}$, and the roughly-regularity of f we have that for fixed x it holds that:

$$\mathop{\mathbf{E}}_{g \xleftarrow{r} \mathcal{G}} \left[\pi_F(x, g, j_0)\right] \leq 2^{r(n)+s(n)+d\log(n)+2} \ .$$

Next, fix any super-constant function $\alpha(n)$. Markov's inequality asserts that

$$\Pr_{g \xleftarrow{r} \mathcal{G}}\left[\pi_F(x,g,j_0) \geq 2^{r(n)+s(n)+d\log(n)} \cdot 2^{\alpha(n)\log(n)}\right] \leq \mathtt{negl}(n) \ .$$

Denote the event that this happens in any of the repetitions by B_2 and note that it happens only with negligible probability (as t is polynomial in n and using the union bound). Let us summarize: Whenever B_2 does not occur,

$$r(n) \leq \log(|T_Z \cup L_Z|) \leq \log(\pi_F(Z)) \leq r(n) + s(n) + (d+\alpha(n))\log(n) \ .$$

When this is the case, both quantities are within an interval of size $s' \stackrel{\text{def}}{=} 3 \cdot \max\{s(n), \alpha(n)\cdot\log(n)\}$.

By Lemma 24 we know that the preimage-entropy and the average accessible max-preimage-entropy values change by at most a negligible quantity when ignoring an event of negligible probability. Specifically, we get that whenever $\overline{B_1} \wedge \overline{B_2}$ happen we have:

$$k' \stackrel{\text{def}}{=} \mathop{\mathbf{E}}_{z \xleftarrow{r} \mathcal{Z}}\left[\log(|S_z|) \,|\, \overline{B_1} \wedge \overline{B_2}\right] \leq \mathop{\mathbf{E}}_{z \xleftarrow{r} \mathcal{Z}}\left[\log(|S_z|)\right] + \mathtt{negl}(n) \tag{4.18}$$

and

$$k'' \stackrel{\text{def}}{=} \mathop{\mathbf{E}}_{z \xleftarrow{r} \mathcal{Z}}\left[\log(\pi_F(z)) \,|\, \overline{B_1} \wedge \overline{B_2}\right] \geq \mathop{\mathbf{E}}_{z \xleftarrow{r} \mathcal{Z}}\left[\log(\pi_F(z))\right] - \mathtt{negl}(n) \tag{4.19}$$

with a gap of $\Delta' \stackrel{\text{def}}{=} k'' - k' \geq \Delta - \mathtt{negl}(n)$.

The Hoeffding bound yields that setting $t \stackrel{\text{def}}{=} O(\frac{s'^2(n)\cdot s^2(n)}{\log(n)})$ assures that the inaccuracies due to the sampling of the independent inputs to F are already smaller than the accumulated gap. By the union bound we have

$$\Pr_{z^t \xleftarrow{r} \mathcal{Z}^t}\left[\log\left(|S_{z^t}|\right) > t\cdot k' + \frac{c}{6}\cdot s'(n)\cdot\sqrt{t\cdot\alpha(n)\cdot\log(n)}\right] \leq \mathtt{negl}(n) \tag{4.20}$$

and

$$\Pr_{z^t \xleftarrow{r} \mathcal{Z}^t}\left[\log\left(\pi_{F^t}(z^t)\right) < t\cdot(k'+\Delta') - \frac{c}{6}\cdot s'(n)\cdot\sqrt{t\cdot\alpha(n)\cdot\log(n)}\right] \leq \mathtt{negl}(n). \tag{4.21}$$

Plugging (4.18) and (4.19) we get that except with negligible probability there is an absolute entropy gap of at least

$$t\cdot\Delta - t\cdot\mathtt{negl}(n) - \frac{c}{3}\cdot s'(n)\cdot\sqrt{t\cdot\alpha(n)\cdot\log(n)} \in \omega(\log(n)) \ . \tag{4.22}$$

4.4.3 A UOWHF in the Non-Uniform Model

To finish the construction we would like to apply the first part of Theorem 33. We use the preimage-entropy of F from Section 4.4.1 (in the form of a non-uniform advice), which equals $k + \Delta$. By what we have shown in Section 4.4.2 it holds that F^t has a strong gap $\omega(\log(n))$ bits. The first part of Theorem 33 with parameters (F^t, τ), where τ is a good estimation of $H_{p,\min}(F^t)$ (given as a non-uniform advice) yields a UOWHF with output length and key length $O(n \cdot {s'}^2(n) \cdot s^2(n)/\log(n))$.

4.4.4 An Efficient Non-Uniform to Uniform Reduction

As explained, the construction obtained requires a non-uniform advice (i.e., the Shannon preimage-entropy of f). Following previous constructions, as in [Rom90] and [HHR+10], we remove the non-uniformity by 'trying all possibilities'. However, as opposed to the case of a general one-way function, where we need to try $O(n^2)$ different values, we show that using the roughness regularity assumption $O(s^2(n))$ tries suffice.

Using the roughly-regularity assumption on f, it follows that the preimage-entropy of F lies in the interval $[r + d\log(n), r + s + d\log(n)]$. Now, for $i \in [\lceil 4 \cdot s(n)/c\log(n) \rceil]$ set $k_i \stackrel{\text{def}}{=} r + d\log(n) + i \cdot \frac{1}{4 \cdot s(n)}$. It holds that one of the k_i is within an additive distance of $\frac{\Delta}{4}$ from the real value $k + \Delta$. Accordingly, set $\tau_i \stackrel{\text{def}}{=} t \cdot (k_i + \Delta) - \frac{c}{4} \cdot s'(n) \cdot \sqrt{t \cdot \alpha(n) \cdot \log(n)}$. It follows that for the same i, (F_t, τ_i) satisfies the premise of the first part of Theorem 33, and thus the second part of the theorem yields a construction of a UOWHF with output length of $O(n \cdot {s'}^2(n) \cdot s^4(n)/\log^3(n))$ and key length of $O(n \cdot {s'}^2(n) \cdot s^4(n)/\log^2(n))$.

4.5 Reflection

We demonstrated how to obtain more efficient constructions of a UOWHF from different assumptions on the structure of the underlying OWF. For the case of known regularity the resulting construction is very efficient and makes an almost logarithmic number of calls to the underlying OWF.

In particular, our construction for a 2^r-regular OWF is very efficient. It requires only $\tilde{O}(\log(n))$ calls to the underlying OWF, whereas the general

construction of Haitner et al. makes polynomially many calls to the OWF. The next plausible extension to our construction, which is probably the "simplest" extension of the class of functions considered here is the following: Let $r_1 = r_1(n)$ and $r_2 = r_2(n)$, and suppose that we know that the underlying OWF is either 2^{r_1}-regular or 2^{r_2}-regular, but we do *not* know which is the case, i.e., we want a single construction to handle both cases. Using standard techniques we observe that one obtains a construction that makes $O(n)$ calls (combining our construction for the regular case with the second part of Theorem 33). That is, already for this case, which is a special case of OWFs of general "unknown" regularity, we observe an exponential gap in the number of calls made to the underlying OWF. The natural question that arises is: "Does there exist a more efficient construction of a UOWHF from a OWF from the aforementioned class?". The next two chapters answer this question negatively.

4.6 Deferred Proofs

4.6.1 Proof of Lemma 38, continued.

Our next goal is to show that the sets $\{T_z \cup L_z\}_{z \in Z}$ satisfy the needed requirements. Claim 4.9 in [HHR$^+$10] shows that any efficient collision-finder cannot (except with negligible probability) output a preimage of $F(z)$ in H_z, as such an algorithm can be used to invert f. Specifically, they show (again, using the constructibility of the three-wise independent hash family as in the second part of Lemma 35) how to efficiently convert any F-collision-finder that outputs a preimage from H_z with probability ϵ to one that inverts a random input of f with probability ϵ/n^{2d}.

As the preimage sets $\{H_z\}_{z \in \mathcal{Z}}$ are inaccessible, it remains to show that they constitute a noticeable part of the preimage sets. In order to complete

the proof, we need to bound:

$$\begin{aligned}
\mathop{\mathbf{E}}_{z \xleftarrow{r} \mathcal{Z}} [\pi_F(z)] - \mathop{\mathbf{E}}_{z \xleftarrow{r} \mathcal{Z}} [\log(|T_z \cup L_z|)] &= \mathop{\mathbf{E}}_{z \xleftarrow{r} \mathcal{Z}} \left[\log \left(\frac{\pi_F(z)}{|T_z \cup L_z|} \right) \right] && (4.23) \\
&= \mathop{\mathbf{E}}_{z \xleftarrow{r} \mathcal{Z}} \left[\log \left(\frac{|T_z| + |L_z| + |H_z|}{|T_z| + |L_z|} \right) \right] && (4.24) \\
&= \mathop{\mathbf{E}}_{z \xleftarrow{r} \mathcal{Z}} \left[\log \left(1 + \frac{|H_z|}{|T_z| + |L_z|} \right) \right] && (4.25) \\
&\geq \frac{1}{2} \mathop{\mathbf{E}}_{z \xleftarrow{r} \mathcal{Z}} \left[\frac{|H_z|}{|T_z| + |L_z| + |H_z|} \right] && (4.26)
\end{aligned}$$

where the second equality is due to the partition in (4.15) and the inequality uses the fact that $\log(1 + x) \geq x/2$ for $x \in [0, 1)$. Thus, it is left to show that indeed $|H_z|$ constitutes a noticeable part of $\pi_F(z)$.

Proposition 39. *Conditioned on $X = x$ and $J = j_i$, define the events:*

$$\begin{aligned}
E^1_{j_i} &\stackrel{\text{def}}{=} \left\{ |H_z| + |L_z| \leq 3 \cdot 2^{n-(j_i - d \log n)} \right\} \\
E^2_{j_i} &\stackrel{\text{def}}{=} \left\{ |H_z| \geq \left(q_i - 4 \cdot \sqrt{1/n^d} \right) \cdot 2^{n-(j_i - d \log n)-1} \right\} \ .
\end{aligned}$$

Then $\Pr_{g \xleftarrow{r} \mathcal{G}} [E^1_{j_i}] > 2/3$ *and* $\Pr_{g \xleftarrow{r} \mathcal{G}} [E^2_{j_i}] > 3/4$ *hold.*

The proof follows the analysis of Claim 4.11 from [HHR$^+$10]. Thus, we

obtain:

$$\mathop{\mathbf{E}}_{z\xleftarrow{r}\mathcal{Z}}[\log(\pi_F(z))] - \mathop{\mathbf{E}}_{z\xleftarrow{r}\mathcal{Z}}[\log(|T_z \cup L_z|)] \tag{4.27}$$

$$\geq \frac{1}{2}\mathop{\mathbf{E}}_{z\xleftarrow{r}\mathcal{Z}}\left[\frac{|H_z|}{|T_z|+|L_z|+|H_z|}\right] \tag{4.28}$$

$$= \frac{1}{2}\sum_{i=0}^{m-1}\Pr_{z\xleftarrow{r}\mathcal{Z}}[J=j_i]\cdot\mathop{\mathbf{E}}_{z\xleftarrow{r}\mathcal{Z}}\left[\frac{|H_z|}{|T_z|+|L_z|+|H_z|}\;\middle|\; J=j_i\right] \tag{4.29}$$

$$\geq \frac{1}{2m}\sum_{i=0}^{m-1}\Pr_{z\xleftarrow{r}\mathcal{Z}}[H_{f(X)}(f(X))\geq j_i]\cdot$$
$$\mathop{\mathbf{E}}_{z\xleftarrow{r}\mathcal{Z}}\left[\frac{|H_z|}{|T_z|+|L_z|+|H_z|}\;\middle|\; H_{f(X)}(f(X))\geq j_i, J=j_i\right] \tag{4.30}$$

$$\geq \frac{1}{2m}\sum_{i=0}^{m-1}(q_i+\cdots+q_m)\cdot\Pr[E^1_{j_i}\wedge E^2_{j_i}\mid J=j_i, H_{f(X)}(f(X))\geq j_i]\cdot$$
$$\mathop{\mathbf{E}}_{z\xleftarrow{r}\mathcal{Z}}\left[\frac{|H_z|}{|T_z|+|L_z|+|H_z|}\;\middle|\; E^1_{j_i}, E^2_{j_i}, H_{f(X)}(f(X))\geq j_i, J=j_i\right] \tag{4.31}$$

$$\geq \frac{1}{2m}\sum_{i=0}^{m-1}(q_i+\cdots+q_m)\cdot(1-\frac{1}{3}-\frac{1}{4})\cdot$$
$$\mathop{\mathbf{E}}_{z\xleftarrow{r}\mathcal{Z}}\left[\frac{|H_z|}{|T_z|+|L_z|+|H_z|}\;\middle|\; E^1_{j_i}, E^2_{j_i}, H_{f(X)}(f(X))\geq j_i, J=j_i\right] \tag{4.32}$$

$$\geq \frac{1}{2m}\sum_{i=0}^{m-1}(q_i+\cdots+q_m)\cdot\frac{5}{12}\cdot\frac{(q_i-4/(n^{d/2}))\cdot 2^{n-(j_i-d\log n)-1}}{2^{n-(j_i-d\log n)+2}} \tag{4.33}$$

$$\geq \left(\frac{5}{2\cdot 8\cdot 12\cdot 2\cdot m}\sum_{0\leq i\leq k\leq m-1}(q_i\cdot q_k)\right) - O\left(\frac{m}{n^{d/2}}\right) \tag{4.34}$$

$$\geq \frac{1}{96m} - O(n^{-d/2+1}) \tag{4.35}$$

$$\geq \frac{1}{200m}\ , \tag{4.36}$$

where we used conditional expectations, the union bound, the fact that $H_{f(X)}(f(x))\geq j_i$ is equivalent to $|T_z|\leq 2^{n-j_i}$ and the roughness-regularity assumption that $\sum_{i=1}^{m} q_i = 1$.

We conclude that the log-size of the set of the accessible inputs to an efficient collision-finder is, on average, bounded away from the point-wise entropy.

Put differently, we obtain a noticeable fraction of $\Omega(1/m) = \Omega(\log(n)/s(n))$ average inaccessible entropy bits.

Chapter 5

A Framework for Black-Box Separations

5.1 Introduction

An important question in complexity-based cryptography is understanding which cryptographic primitives (e.g., one-way functions, pseudo-random generators) are implied by others. In principle, an implication between two primitives can be proved as a logical statement (e.g., the existence of one-way functions implies the existence of pseudo-random generators). However, most proofs of such implications (with very few exceptions, e.g., [Bar01]) are in fact so-called fully black-box constructions.

Informally, a black-box construction of a primitive **Q** from a primitive **P** is a pair of algorithms, called *construction* and *reduction*, such that the construction, using only the functionality of **P**, implements **Q** and the reduction, using only the functionality of **P** and that of a potential breaker algorithm, breaks **P** whenever the breaker algorithm breaks **Q**. As a corollary, such a black-box construction establishes that the existence of **P** implies the existence of **Q**. One of many such examples is the construction of a one-way function from a weak one-way function [Yao82].

After futile attempts to prove that the existence of one-way functions implies that of key agreement, Impagliazzo and Rudich [IR89] proved the first black-box separation result: They showed that there is no fully black-box construction of key agreement protocol from a one-way function. Their seminal work inspired a plethora of similar results and nowadays one identifies two main types of black-box separation results: Black-box separations of a primitive **Q** from a primitive **P** and lower bounds on some complexity parameter (e.g., seed length, number of calls to the underlying primitive, etc.) in the construction of **Q** from **P**. Besides [IR89], the work of Simon

[Sim98], where he shows that there is no fully black-box construction of a collision-resistant hash function from a one-way function, is an example of the former. As an example of the latter, Kim et al. [KST99] established a lower bound of $\Omega(\sqrt{k/\log(n)})$ on the number of queries of any construction of a universal one-way hash function that compresses k bits from a one-way permutation on n bits. This was later improved by Gennaro et al. [GGKT05] to $\Omega(k/\log(n))$.

Reingold et al. [RTV04] were the first to formalize a model for and study the relations between different notions of "black-boxness" of cryptographic constructions.

A key property of a fully black-box construction of **Q** from **P** is the requirement that it constructs **Q** efficiently even when given black-box access to a non-efficient implementation of **P**. A proof technique utilizing this property, which is implicit in many black-box separations, involves an (inefficient) oracle instantiation of the primitive **P** and an appropriate (inefficient) breaker oracle B. The separation is usually proved by showing that B breaks the security of the candidate construction for **Q**, but at the same time no efficient oracle algorithm that has black-box oracle access to both the breaker and the primitive (in particular, the potential reduction) breaks the security property of the underlying instantiation of **P**.

In [HR04], Hsiao and Reyzin introduce the "two-oracle" paradigm, referring to the oracle implementations of **P** and the breaker B. The separation in [MM11] also makes explicit use of this paradigm.

5.1.1 Contributions of this Chapter

In constructions based on one-way functions (or permutations), i.e., when $\mathbf{P} = \mathbf{OWF}$, the oracle that implements **OWF** is usually set to be a random permutation, which is one-way with very high probability even in the presence of a non-uniform algorithm. On the other hand, the proof that the breaker algorithm for the constructed primitive **Q** does not help invert the permutation is repeated in an "ad-hoc" manner in many separation proofs, e.g., in [Sim98, HHRS07] and also in a recent result on lower bounds on the number of calls made by any construction of a pseudo-random generator from a one-way function [HS12].

Thus, while in many separation proofs the task of finding the right breaker oracle is different (this is inherent, as each time it is required to break the security of a different primitive), we observe that the proof that it does not

help in inverting the underlying one-way function can be facilitated and unified to a large extent. To that end, we prove a general theorem that facilitates the proof of black-box separations (Theorem 57). In particular, we show that any circuit with access to an oracle that satisfies two local properties, does not help to invert many functions.

Our framework allows proving separation results that exclude the existence of reductions with very weak security requirements. In this work we focus on the important case where the black-box construction is so-called fixed-parameter. That is, for a security parameter ρ, both the construction algorithm and the reduction access the primitive and breaker of security ρ only. All black-box constructions found in the literature are in fact fixed-parameter constructions. We believe that adapting the approach of [HS12], it is possible to extend our results to the most general case.

Our proof uses the encoding technique from [GGKT05], which was already adapted to the special cases in [HHRS07] and [HS12]. We also use the bending technique that originated in [Sim98] and was subsequently used in [HH09] and [HS12].

5.2 Preliminaries

5.2.1 A Non-Uniform Computational Model

A boolean circuit $A : \{0,1\}^m \to \{0,1\}^{m'}$ is a directed acyclic graph in which every node (called gate) is either an input node of in-degree 0 labeled as one of the m input bits, an output node labeled by one of the m' output bits, an AND gate, an OR or a NOT gate. A circuit A implements the function $f : \{0,1\}^m \to \{0,1\}^{m'}$ that corresponds to its evaluation on its inputs. The converse also holds: For every function $f : \{0,1\}^m \to \{0,1\}^{m'}$ it is always possible to define canonically[1] a circuit A that implements f.

Let $n, n' \in \mathbb{N}^+$. An (n, n')-oracle circuit $C^{(?)}$ is a circuit that additionally has special "oracle" gates, each having in-degree n and out-degree n'. A circuit $C^{(?)}$ is an oracle circuit if it is an (l, l')-oracle circuit for some $l, l' \in \mathbb{N}^+$. Let $C^{(?)}$ be an (n, n')-oracle circuit and let A be a circuit with n input gates and n' output gates (in this case we say that A is compatible with $C^{(?)}$). The circuit $C^{(A)}$ is defined as the circuit $C^{(?)}$ where each oracle gate is substituted

[1] E.g., the circuit that implements the DNF of f.

by a copy A. For a function $f : \{0,1\}^n \to \{0,1\}^{n'}$ define $C^{(f)}$ as the circuit $C^{(?)}$ where each oracle gate is substituted by a copy of the canonical circuit that evaluates f. Similarly, let $n_1, n_1', n_2, n_2' \in \mathbb{N}^+$. An (n_1, n_1', n_2, n_2')-two oracle circuit $C^{(?,?)}$ is a circuit that has two types of oracle gates, where a gate of the first type has n_1 inputs and n_1' outputs and a gate of the second type has n_2 inputs and n_2' outputs. As before, for compatible functions and circuits that evaluate compatible functions $f_1 : \{0,1\}^{n_1} \to \{0,1\}^{n_1'}$ and $f_2 : \{0,1\}^{n_2} \to \{0,1\}^{n_2'}$ the circuit $C^{(f_1,f_2)}$ is defined similarly.

A non-uniform algorithm $A = \{A_\rho\}_{\rho\in\mathbb{N}^+}$ is a parameterized family of circuits A_ρ. A non-uniform algorithm A implements the parametrized functions family $f = \{f_\rho\}_{\rho\in\mathbb{N}^+}$, if each A_ρ implements f_ρ.

A non-uniform oracle algorithm $A^{(?)} = \{A^{(?)}{}_\rho\}_{\rho\in\mathbb{N}^+}$ is a parameterized family of oracle circuits. For an oracle algorithm $A^{(?)}$ and a family of functions f we define $A^{(f)} \stackrel{\text{def}}{=} \{A_\rho^{f_\rho}\}$ (resp., $A^{(B)} \stackrel{\text{def}}{=} \{A_\rho^{B_\rho}\}$) whenever for every ρ the function is compatible with the oracle-circuit.

Uniform generation of oracle algorithms.

The construction and reduction algorithms in fully black-box constructions are assumed to work for any[2] input/output lengths of the primitive and breaker functionalities, and therefore are modeled in the following way: In addition to the security parameter ρ, both the construction and the reduction algorithms take as input information about the input/output lengths of the underlying primitive f_ρ and the breaker algorithm B_ρ.

A uniform oracle algorithm is a machine M that on input $M(1^\rho, n(\rho), n'(\rho))$ outputs an $(n(\rho), n'(\rho))$-oracle circuit $A_\rho^{(?)}$. For a uniform oracle algorithm M and a parameterized family of functions $f = \{f_\rho : \{0,1\}^{n(\rho)} \to \{0,1\}^{n'(\rho)}\}_{\rho\in\mathbb{N}^+}$, define $M^{(f)} \stackrel{\text{def}}{=} \{A_\rho^{(f_\rho)}\}_{\rho\in\mathbb{N}^+}$, where $A_\rho^{(?)} \stackrel{\text{def}}{=} M(1^\rho, n(\rho), n'(\rho))$. For a non-uniform algorithm A, the family $M^{(A)}$ is defined analogously.

Let $s = s(\rho)$ be a security function. An s-non-uniform two oracle algorithm is a machine M such that for every $\rho, n_1, n_1', n_2, n_2' \in \mathbb{N}^+$ and every $a \in \{0,1\}^{s(\rho)}$, it holds that $M(1^\rho, n_1, n_1', n_2, n_2', a)$ outputs an (n_1, n_1', n_2, n_2')-two oracle circuit $A_{\rho,a}^{(?,?)}$ with at most $s(\rho)$ oracle gates. Note that the last requirement

[2] A-priori, for a fixed security parameter ρ there is no bound on the input length the construction is expected to work, as long as the series of the input-output lengths is bounded by *some* polynomial.

is essential and is implicit in the case of an efficient uniform oracle algorithm, where the number of oracle gates is bounded by the polynomial that bounds the running time of the algorithm. For an s-non-uniform two oracle algorithm M, a non-uniform algorithm B and a family of functions f, we formally define $M^{[B,f]} \stackrel{\text{def}}{=} (M, B, f)$.

5.2.2 Modeling Cryptographic Primitives

In order to state our results in their full generality, and in particular to exclude reductions that are allowed to use non-uniformity and are considered successful in inverting the one-way function even if they invert only a negligible fraction of the inputs of the function, the following two definitions are very general, and extend Definitions 2.1 and 2.3 from [RTV04]. The example of modeling a one-way function follows the definition.

Definition 40 (Cryptographic Primitive)**.** *A primitive* $\mathbf{Q}$ *is a pair* $\langle F_{\mathbf{Q}}, R_{\mathbf{Q}} \rangle$*, where* $F_{\mathbf{Q}}$ *is a set of parametrized families of functions* $f = \{f_\rho\}_{\rho \in \mathbb{N}^+}$ *and* $R_{\mathbf{Q}}$ *is a relation over triplets* $\langle f_\rho, C, \epsilon \rangle$ *of a function* $f_\rho \in f$ *(for some* $f \in F_{\mathbf{Q}}$*), a circuit* C *and a number* $\epsilon > 0$*. We define that* C $(\mathbf{Q}, \epsilon)$-breaks f_ρ *if and only if* $\langle f_\rho, C, \epsilon \rangle \in R_{\mathbf{Q}}$*.*

The set $F_{\mathbf{Q}}$ specifies all the correct implementations (not necessarily efficient) of $\mathbf{Q}$ and the relation $R_{\mathbf{Q}}$ captures the security property of $\mathbf{Q}$, that is, it specifies for every concrete security parameter implementation, how well a breaker algorithm performs with respect to the security property of the primitive.

Finally, let $s = s(\rho)$ be a security function, $B = \{B_\rho\}_{\rho \in \mathbb{N}^+}$ be a non-uniform algorithm, and $f \in F_{\mathbf{Q}}$. We say that B $(\mathbf{Q}, \frac{1}{s})$-breaks f if $\langle f_\rho, B_\rho, \frac{1}{s(\rho)} \rangle \in R_{\mathbf{Q}}$ for infinitely many values ρ. Let us fix an s-non-uniform two oracle algorithm R. We say that $R^{[B,f]}$ $(\mathbf{Q}, \frac{1}{s})$-breaks f if for infinitely many values ρ there exists an $a \in \{0,1\}^{s(\rho)}$ (called advice) such that $\langle f_\rho, R_{\rho,a}^{(B_\rho, f_\rho)}, \frac{1}{s(\rho)} \rangle \in R_{\mathbf{Q}}$, where $R_{\rho,a}^{(?,?)} = R(1^\rho, n, n', b, b', a)$.

The usual notion of polynomial security of a primitive is captured by the following definition: B $\mathbf{Q}$-breaks f if there exists a polynomial $p = p(\rho)$ such that B $(\mathbf{Q}, \frac{1}{p})$-breaks f.

A primitive $\mathbf{Q}$ exists if there exists an efficient uniform algorithm M that implements an $f \in F_{\mathbf{Q}}$, and for every efficient uniform algorithm M' that, on input 1^ρ outputs a circuit, it holds that $\{M'(1^\rho)\}_{\rho \in \mathbb{N}^+}$ does not $\mathbf{Q}$-break f.

Observe that the requirement that M' outputs a circuit is made without loss of generality and captures the standard definition of an efficient randomized machine M' that breaks a primitive. Given such an M' that tosses at most $r = r(\rho)$ random coins, there exists[3] a (now deterministic) efficient uniform machine M'' that on input 1^ρ outputs a circuit C_ρ with $m(\rho) + r(\rho)$ input gates and $n(\rho)$ output gates that computes the output of M for all strings of length $m(\rho)$, and therefore **Q**-breaks the primitive.

5.2.3 One-Way Functions

Our model for describing a primitive is very general and captures the security properties of many cryptographic primitives. As an example, we recall the definition of a one-way function and then explain how it can be described in our model.

Definition 41 (One-Way Function)**.** *A* one-way function $f = \{f_\rho\}_{\rho\in\mathbb{N}^+}$ *is an efficiently uniformly computable family of functions* $f_\rho : \{0,1\}^{n(\rho)} \to \{0,1\}^{m(\rho)}$*, such that for every efficient randomized machine* A*, the function that maps* ρ *to*

$$\Pr_{x\xleftarrow{r}\{0,1\}^{m(\rho)},A}\left[A(1^\rho, f_\rho(x)) \in f_\rho^{-1}(f_\rho(v))\right]$$

is negligible.

In order to model a one-way function (OWF), we set $f = \{f_\rho\}_{\rho\in\mathbb{N}^+} \in F_{\mathbf{OWF}}$, where $f_\rho : \{0,1\}^{n(\rho)} \to \{0,1\}^{m(\rho)}$, if and only if $n = n(\rho)$ and $m = m(\rho)$ are polynomial security functions. We say that $F_{\mathbf{OWF}}$ contains a collection of sets of functions $\mathcal{F} = \{\mathcal{F}_\rho\}_{\rho\in\mathbb{N}^+}$, if for every family $f' = \{f'_\rho\}_{\rho\in\mathbb{N}^+}$, where $f'_\rho \in \mathcal{F}_\rho$ for every ρ, it holds that $f' \in F_{\mathbf{OWF}}$.

In this case, for a function $f_\rho \in f \in F_{\mathbf{OWF}}$, a circuit C that inverts f_ρ on an ϵ-fraction of its inputs, and $\epsilon' > 0$, set $\langle f, C, \epsilon'\rangle \in R_{\mathbf{OWF}}$ if and only if $\epsilon \geq \epsilon'$. The definition is general, and allows for the circuit C to implicitly use randomness. In such a case, for f_ρ as before, a circuit with C with $m(\rho) + r(\rho)$ input bits that computes an output $x \in \{0,1\}^{n(\rho)}$, and a value $\epsilon' > 0$, define $\langle f_\rho, C, \epsilon'\rangle \in R_{\mathbf{OWF}}$ if and only if $\epsilon \geq \epsilon'$, where ϵ is the probability over uniform $z \in \{0,1\}^{r(\rho)}$ and $x \in \{0,1\}^{n(\rho)}$ that $C(f_\rho(x), z)$ outputs an $x' \in f_\rho^{-1}(f_\rho(x))$.

[3]For example, by the canonical encoding of an efficient machine as in the Cook-Levin Theorem.

5.2.4 Fully Black-Box Cryptographic Constructions

Finally, we bring the standard definition of a fixed-parameter fully black-box construction of a primitive $\mathbf{Q}$ from a primitive $\mathbf{P}$, which is usually implicit in the literature. The construction algorithm G is an efficient uniform oracle algorithm and the security reduction R is an efficient uniform two-oracle algorithm. For every security parameter ρ and a function $f_\rho : \{0,1\}^{n(\rho)} \to \{0,1\}^{n'(\rho)}$, G's output on $(1^\rho, n, n')$ is an (n, n')-oracle circuit $g_\rho^{(?)}$ such that $\{g_\rho^{(f_\rho)}\}_{\rho\in\mathbb{N}^+}$ implements $\mathbf{Q}$. The reduction algorithm works as follows: For a security parameter ρ and f as before, and additionally a breaker circuit $B : \{0,1\}^{b(\rho)} \to \{0,1\}^{b'(\rho)}$, the reduction R on input $(1^\rho, n, n', b, b')$ outputs an (n, n', b, b')-two-oracle circuit $R_\rho^{(?,?)}$. The security property property requires that indeed the series of circuits $\{R_\rho^{(B_\rho, f_\rho)}\}_{\rho\in\mathbb{N}^+}$ $\mathbf{P}$-breaks f. We emphasize that the vast majority (if not all) of the constructions of primitives from a one-way function found in the literature are in fact fixed-parameter fully black-box constructions. Formally:

Definition 42 (fixed-parameter fully black-box construction of $\mathbf{Q}$ from $\mathbf{P}$)**.** *An efficient uniform oracle algorithm G and an efficient uniform two oracle algorithm R are a* fixed-parameter fully-BB construction *of a primitive $\mathbf{Q} = \langle F_\mathbf{Q}, R_\mathbf{Q}\rangle$ from a primitive $\mathbf{P} = \langle F_\mathbf{P}, R_\mathbf{P}\rangle$ if for every $f \in F_\mathbf{P}$:*

1. **(correctness)** *$G^{(f)}$ implements $f' \in F_\mathbf{Q}$.*
2. **(security)** *For every algorithm B: If B $\mathbf{Q}$-breaks $G^{(f)}$ then $R^{(B,f)}$ $\mathbf{P}$-breaks f.*

For a super-polynomial security function $s = s(\rho)$ (e.g., $s(\rho) = 2^{\sqrt{\rho}}$), the following definition of a fully black-box construction is significantly weaker than the standard one in the following three aspects: First, it requires that reduction only mildly breaks the one-way property of the function f (whenever the breaker breaks the constructed primitive in the standard polynomial sense). Second, the reduction algorithm does not have to be efficient or uniform (but the non-uniformity is limited to an advice of length s). Lastly, it allows the reduction to make s calls to its oracles[4].

Definition 43 (s-weak fixed-parameter fully black-box construction of $\mathbf{Q}$ from $\mathbf{P}$)**.** *A uniform oracle algorithm G and an s-non-uniform two oracle*

[4]In Definition 42 the limitation on the number of queries made to the oracles is implicit as R is an efficient algorithm, and so its output circuit has at most a polynomially number of oracle gates.

*algorithm R are an s-*weak fixed-parameter fully-BB construction *of a primitive $\mathbf{Q} = \langle F_{\mathbf{Q}}, R_{\mathbf{Q}} \rangle$ from a primitive $\mathbf{P} = \langle F_{\mathbf{P}}, R_{\mathbf{P}} \rangle$ if for every $f \in F_{\mathbf{P}}$:*

1. **(correctness)** *$G^{(f)}$ implements an $f' \in F_{\mathbf{Q}}$.*
2. **(security)** *For every non-uniform algorithm B: If B $\mathbf{Q}$-breaks $G^{(f)}$ then $R^{[B,f]}$ $(\mathbf{P}, 1/s)$-breaks f.*

5.2.5 Random Permutations and Regular Functions

Let n and i be two integers such that $0 \leq i \leq n$. We denote the set of all permutations on $\{0,1\}^n$ by $\mathcal{P}_n$. Let $\mathcal{X}, \mathcal{Y}$ be sets. We denote by $(\mathcal{X} \to \mathcal{Y})$ the set of all functions from $\mathcal{X}$ to $\mathcal{Y}$. A function $f : \mathcal{X} \to \mathcal{Y}$ is regular if $|\{x' : f(x) = f(x')\}|$ is constant for all $x \in \mathcal{X}$. A family of functions $f = \{f_\rho\}_{\rho \in \mathbb{N}^+}$ is a regular function if for every ρ the function f_ρ is regular. We denote by $\mathcal{R}_{n,i}$ the set of all regular functions from $\{0,1\}^n$ to itself such that the image of f contains 2^i values. E.g., $\mathcal{R}_{n,n} = \mathcal{P}_n$ is the set of all permutations, and $\mathcal{R}_{n,0}$ is the set of all constant functions.

5.2.6 Bending a Function and Image Adaptation

It will be useful for us to compare the run of a circuit with oracle access to a function f to a run that is identical except that the output of one specific value is altered.

For a fixed function $f : \{0,1\}^n \to \{0,1\}^n$ and $y', y'' \in \{0,1\}^n$, set

$$f_{(y',y'')}(x) \stackrel{\text{def}}{=} \begin{cases} y'' & \text{if } f(x) = y' \\ f(x) & \text{otherwise.} \end{cases}$$

Similarly, for two fixed functions $f, f' : \{0,1\}^n \to \{0,1\}^n$ and a set $S \subset \{0,1\}^n$, we define the image adaptation [5] of f to f' on S to be the function

$$f_{(S,f')}(x) \stackrel{\text{def}}{=} \begin{cases} f'(x) & \text{if } x \in f^{-1}(f(S)) \\ f(x) & \text{otherwise.} \end{cases}$$

[5] We mention that if f is a permutation, the condition $f(x) = y$ can be replaced by $x = f^{-1}(y)$, and similarly for $f_{(S,f')}$ check whether $x \in S$, which is what one may expect initially from such a definition.

5.3 A General Theorem for Proving Strong Black-Box Separations

5.3.1 Deterministic Parametrized Oracles and Local Sets

The following definition allows to model general parameterized oracles, that is, oracles that, for any function f from some set of functions and any q from some query domain, return a value a from some answer set. We observe that many of the oracles used for black-box separations found in the literature could be described in such a way.

Let $\mathcal{X}, \mathcal{Y}, \mathcal{D}$ and $\mathcal{R}$ be sets. A deterministic parametrized oracle for a class of functions $(\mathcal{X} \to \mathcal{Y})$ is an indexed collection $\mathcal{O} = \{\mathcal{O}_f\}_{f \in \mathcal{X} \to \mathcal{Y}}$, where $\mathcal{O}_f : \mathcal{D} \to \mathcal{R}$. We call f, $\mathcal{D}$, and $\mathcal{R}$ the function parameter, the domain, and the range of the oracle, respectively.

Our first example of a deterministic parametrized oracle is the evaluation oracle $\mathcal{E}$ for functions on $\{0,1\}^n$, which on a query q returns the evaluation of f on q. In this case we have that $\mathcal{X} = \mathcal{Y} = \mathcal{D} = \mathcal{R} = \{0,1\}^n$ and $\mathcal{E}_f(q) \stackrel{\text{def}}{=} f(q)$.

The next two definitions capture two important local properties of parametrized oracles. We believe that they are natural and observe that many of the oracles devised for separation results satisfy them.

Intuitively, a determining set is an indexed collection of sets that determine the output of the oracle for every function f and query q in the following sense: If for two functions f and f' it holds that their corresponding oracle outputs differ for some q, then for one of them (f or f') it holds that the local change of an image adaptation of one of the functions to agree with that of the other on its determining set changes the output of the oracle. Formally:

Definition 44. *Let $\mathcal{O}$ be a deterministic parametrized oracle. A* determining set *$\mathcal{I}^{\mathcal{O}}$ for a class of functions $\mathcal{F} \subset (\mathcal{X} \to \mathcal{Y})$ is an indexed collection $\{\mathcal{I}^{\mathcal{O}}_{f,q}\}_{f \in \mathcal{F}, q \in \mathcal{D}}$ of subsets of $\mathcal{X}$, such that for every $f, f' \in \mathcal{F}$ and every query $q \in \mathcal{D}$: If $\mathcal{O}_f(q) \neq \mathcal{O}_{f'}(q)$, then it holds that either the image adaptation of f to f' on $\mathcal{I}^{\mathcal{O}}_{f',q}$ changes $\mathcal{O}_f(q)$ (i.e., $\mathcal{O}_{f_{\left(\mathcal{I}^{\mathcal{O}}_{f',q}, f'\right)}}(q) \neq \mathcal{O}_f(q)$), or the image adaptation of f' to f on $\mathcal{I}^{\mathcal{O}}_{f,q}$ changes $\mathcal{O}_{f'}(q)$. $\mathcal{I}^{\mathcal{O}}$ is a t-*determining set *if for every function $f \in \mathcal{F}$ and query $q \in \mathcal{D}$ it holds that $\left|\mathcal{I}^{\mathcal{O}}_{f,q}\right| \leq t$.*

In the example of the evaluation oracle, we observe that it has a 1-determining set. Indeed, setting $\mathcal{I}^{\mathcal{E}}_{f,q} \stackrel{\text{def}}{=} \{q\}$ satisfies the required definition, since if for any $f, f' \in \{0,1\}^n \to \{0,1\}^n$ and $x \in \{0,1\}^n$ for which $f(x) \neq f'(x)$ it holds that $f_{(\{x\},f')}(x) = f'(x) \neq f(x)$.

Consider an oracle $\mathcal{O}$ with a determining set $\mathcal{I}^{\mathcal{O}}$ for some class of permutations $\mathcal{F}$. Fix $f, f' \in \mathcal{F}$ and $q \in \mathcal{D}$. The following two propositions are immediate from the definition of determining sets:

Proposition 45. *If $\mathcal{O}_f(q) \neq \mathcal{O}_{f'}(q)$ and $f(x) = f'(x)$ for all $x \in \mathcal{I}^{\mathcal{O}}_{f',q}$ (in this case we say that f agrees with f' on $\mathcal{I}^{\mathcal{O}}_{f',q}$), then adapting f' to agree with f on $\mathcal{I}^{\mathcal{O}}_{f,q}$ changes $\mathcal{O}_{f'}(q)$.*

Proposition 46. *If for all $x \in \mathcal{I}^{\mathcal{O}}_{f,q} \cup \mathcal{I}^{\mathcal{O}}_{f',q}$ it holds that $f(x) = f'(x)$ (in this case we say that the functions* agree *on their determining sets), then $\mathcal{O}_f(q) = \mathcal{O}_{f'}(q)$.*

Proposition 46 establishes that determining sets indeed determine the output of the oracle in the following sense: If we know the value $\mathcal{O}_f(q)$ for a query q and a function f, and, moreover, we know that functions f', f agree on their determining sets for q, then this information already determines for us the value $\mathcal{O}_{f'}(q)$.

The next local property of an oracle captures the fact that it is in some sense "stable". For a function f and query q as before, and a value y in the image set of f, a bending set for f, q, and y is a set of all potentially "sensitive" y' values: For any value y which is not in the image of f on its determining set, and for any value y' which is not in the bending set, the oracle's answer to query q does not change for the local adaptations of f from y' to y. That is, it holds that $\mathcal{O}_{f_{(y',y)}}(q) = \mathcal{O}_f(q)$. Formally:

Definition 47. *Let $\mathcal{O}$ be a deterministic parametrized oracle. A* bending set *$\mathcal{B}^{\mathcal{O}}$ for $\mathcal{F}$ is an indexed collection $\{\mathcal{B}^{\mathcal{O}}_{f,q,y}\}_{f\in\mathcal{F},q\in\mathcal{D},y\in\mathcal{Y}}$ of subsets of $\mathcal{Y}$, such that for every function $f \in \mathcal{F}$, query $q \in \mathcal{D}$, for every target image $y \in \mathcal{Y}$, and for every source image $y' \notin \mathcal{B}^{\mathcal{O}}_{f,q,y}$, it holds that $\mathcal{O}_f(q) = \mathcal{O}_{f_{(y',y)}}(q)$. We say that $\mathcal{B}^{\mathcal{O}}$ is a t-*bending set *if for every function $f \in \mathcal{F}$, query $q \in \mathcal{D}$ and $y \in \mathcal{Y}$ it holds that $|\mathcal{B}^{\mathcal{O}}_{f,q,y}| \leq t$.*

For the example of the evaluation oracle, we observe that it also has a 1-bending set. Setting $\mathcal{B}^{\mathcal{E}}_{f,q,y} \stackrel{\text{def}}{=} \{f(q)\}$ (for the relevant f, q and y) satisfies the required definition. Indeed, for any $y' \neq f(q)$ and $y'' \in \mathcal{Y}$, it holds that $\mathcal{E}_{f_{(y',y'')}}(q) = f_{(y',y'')}(q) = f(q) = \mathcal{E}_f(q)$.

Finally, a deterministic parametrized algorithm $\mathcal{O}$ is t-stable for a class of functions $\mathcal{F}$ if there exist $(\mathcal{I}^{\mathcal{O}}, \mathcal{B}^{\mathcal{O}})$ that are a t-determining set and a t-bending set for $\mathcal{F}$, respectively, and at least one of them is not empty.

We note that determining and bending sets always exist unconditionally (just choose the entire domain and range of f, for every determining and bending set, respectively). The challange is finding an oracle that allows to break a primitive and at the same time is t-stable.

Exhaustive-search oracles.

We identify that all the parametrized oracles in the literature are in fact of a special type, which we call exhaustive-search oracles. We say that an oracle is an exhaustive-search oracle if there is an oracle circuit $\Phi^{(?)} : \mathcal{D} \times \mathcal{R} \to \{0,1\}$ (called a predicate) such that for every function f and query q the computation of $\mathcal{O}_f(q)$ can be computed by a loop (according to an understood enumeration) over the values $v \in \mathcal{R}$, where in each step the current value v is checked to satisfy a predicate $\Phi^{(f)}(q, v)$. If the predicate is satisfied, i.e., $\Phi^{(f)}(q, v) = 1$, the algorithm outputs v, and if no such v exists it returns a special bottom value $\perp$.

Formally, a deterministic parameterized oracle $\mathcal{O}$ is an exhaustive-search oracle if there exists Φ (as before) such that following algorithm outputs $\mathcal{O}_f(q)$ for every function parameter f and every query $q \in \mathcal{D}$:

Exhaustive Search Algorithm $\mathcal{O}_f(q)$ (on function $f \in \mathcal{X} \to \mathcal{Y}$ and input $q \in \mathcal{D}$)

---

for all $v \in \mathcal{R}$ **do**
. **if** $\Phi^{(f)}(q, v) = 1$ **then**
. **return** v
return $\perp$

Observe that the range of an exhaustive-search algorithm is the set $\mathcal{R} \cup \{\perp\}$. Of special interest are exhaustive-search oracles that make relatively few queries to their oracles (e.g., a polynomial number of queries). We note that most parameterized oracles in the literature are in fact of this type. The following lemma shows that an exhaustive-search oracle $\mathcal{O} : \mathcal{D} \to \mathcal{R} \cup \{\perp\}$ for which Φ makes t queries has a t-determining set.

Lemma 48. *Let $\mathcal{O}$ be an exhaustive-search oracle with predicate $\Phi^{(?)} : \mathcal{D} \times \mathcal{R} \to \{0,1\}$ that makes t queries to its oracle. Denote by $X_\Phi^{(f)}(q,v)$ the list of queries issued by the predicate during the evaluation of $\Phi^{(f)}(q,v)$. Then*

$$\mathcal{I}_{f,q}^{\mathcal{O}} \stackrel{\text{def}}{=} \begin{cases} X_\Phi^{(f)}(q,v) & v \text{ is the first value for which } \Phi^{(f)}(q,v) = 1 \\ X_\Phi^{(f)}(q,v_{\text{last}}) & \text{if } \mathcal{O}_f(q) = \perp, \text{ where } v_{\text{last}} \text{ is the last value in the enumeration of } \mathcal{R} \end{cases}$$

is a t-determining set for $\mathcal{O}$.

The proof follows by inspection of the algorithm and the definition of determining sets.

Proof: The bound on the size of the set is immediate and so we check that $\mathcal{I}_{f,q}^{\mathcal{O}}$ is indeed a determining set. Suppose that for two functions f, f' and a query q it holds that $\mathcal{O}_f(q) \neq \mathcal{O}_{f'}(q)$. It follows that for at least one of the functions, there is an iteration v for which the predicate $\Phi^{(f)}(q,v)$ holds (otherwise both return $\perp$). W.l.o.g. assume that this happens for f, i.e., that $\mathcal{O}_f(q) = v$, and that v is the minimal of the returned values, i.e., that $\mathcal{O}_{f'}(q) = v'$, where either $v' > v$ or $v' = \perp$. Now, observe that after adapting f' to agree with f on $\mathcal{I}_{f,q}^{\mathcal{O}} = X_\Phi(f,q)$ it holds that with the adapted function oracle the predicate circuit returns v (or some value that appears in the enumeration before v), as the answer of the predicate depends only on the answers of the function to the query set. ∎

The choice of v_{last} (for the case $\mathcal{O}_f(q) = \perp$) is somewhat arbitrary, but it will be useful for us to have the queries involved with q on some arbitrary evaluation of the predicate as part of the bending set. We note that this is not needed for the proof of the lemma.

5.3.2 A t-Stable Oracle $\mathcal{O}_f$ Inverts Only a Few Functions

The next lemma, which first appeared in [GGKT05] and was subsequently adapted to many other separation results, e.g., [HHRS07, RS10, HS12], establishes an information-theoretic bound on the number of functions an oracle-aided algorithm can invert from a set $\mathcal{F}$ if the oracle is t-stable for $\mathcal{F}$. Essentially, it shows that given an oracle circuit $A^{(?)}$ with access to such an oracle $\mathcal{O}$, it is possible to encode a function $f \in \mathcal{F}$ that A inverts well

using significantly fewer bits than $\log(|\mathcal{F}|)$, such that f can still be fully reconstructed, or equivallently, that the encoding is injective.

Lemma 49 (Encoding Lemma)**.** *Let $A^{(?)}$ be an oracle circuit making at most c calls to its oracle, and let $\mathcal{O} = \{\mathcal{O}_f\}_{f\in\{0,1\}^n\to\{0,1\}^n}$ be a deterministic parameterized oracle such that for a class of permutations $\mathcal{F} \subseteq \mathcal{P}_n$ it is t-stable with sets $(\mathcal{I}^{\mathcal{O}}, \mathcal{B}^{\mathcal{O}})$. Then, for at most $d_n = d_n(c,t) = \left(\binom{2^n}{b}\right)^2 \cdot ((2^n - b)!)$, where $b \stackrel{\text{def}}{=} \frac{2^n}{3\cdot c^{2}\cdot t}$, of the permutations f in $\mathcal{F}$, it holds that $\Pr_{x\xleftarrow{r}\{0,1\}^n}\left[A^{\mathcal{O}_f}(f(x)) = x\right] > \frac{1}{c}$.*

The proof is a generalized version of the encoding technique of [GGKT05].

Proof: We describe an injective canonical mapping from the set of all functions $f \in \mathcal{F}$ which $A^{\mathcal{O}_f}$ inverts well (i.e., inverts more than a $\frac{1}{c}$-fraction of the inputs to f) into a small set.

Denote by $\text{Queries}(A^{\mathcal{O}_f}(y))$ the list of at most c of queries made by A to $\mathcal{O}_f$ during its computation on input y. Consider the following algorithm:

Algorithm Buildimage(f):

1. $S := \emptyset$
2. $T := \{y : A^{\mathcal{O}_f}(y) \in f^{-1}(y)\}$
3. **while** $T \neq \emptyset$ **do** :
4. $\quad y' := \min_{y\in T}$
5. $\quad S := S \cup \{y'\}$
6. $\quad R := f\left(\bigcup_{q\in\text{Queries}\left(A^{\mathcal{O}_f}(y')\right)} \mathcal{I}^{\mathcal{O}}_{f,q}\right)$
7. $\quad R := R \cup \bigcup_{q\in\text{Queries}\left(A^{\mathcal{O}_f}(y')\right)} \mathcal{B}^{\mathcal{O}}_{f,q,y'}$
8. $\quad R := R \cup \{y'\}$
9. $\quad T := T \setminus R$
10. **end**

Define the mapping $\mathcal{M}$ that maps every function f which $A^{\mathcal{O}_f}$ inverts well to a list of its action on $f^{-1}(\mathcal{Y} \setminus S)$ as a list of pairs, i.e., $f \mapsto \left(x_1, f(x_1), x_2, f(x_2), \ldots, x_{|\mathcal{Y}\setminus S|}, f(x_{|\mathcal{Y}\setminus S|})\right)$, where $x_1 < x_2 < \cdots < x_{|\mathcal{Y}\setminus S|}$. We show that $\mathcal{M}$ is injective. Suppose that for $f_1, f_2 \in \mathcal{F}$, $A^{\mathcal{O}_{f_1}}$ and $A^{\mathcal{O}_{f_2}}$ invert well f_1 and f_2, respectively.

Claim 50. *If $\mathcal{M}(f_1) = \mathcal{M}(f_2)$ then $f_1 = f_2$.*

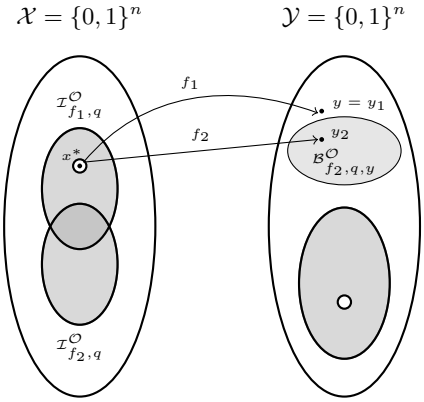

Figure 5.1: An illustration of the proof of Lemma 49. The functions f_1 and f_2 agree on $\mathcal{I}^{\mathcal{O}}_{f_1,q} \setminus \{x^*\}$, $f_1(x^*) = y = y_1 \neq y_2 = f_2(x^*)$ and adapting f_2 to f_1 on $\mathcal{I}^{\mathcal{O}}_{f_1,q}$ changes $\mathcal{O}_{f_2}(q)$.

Proof: Assume towards contradiction that $f_1 \neq f_2$. We first observe that both functions agree on the sets $\{y : A^{\mathcal{O}_{f_1}}(y) \notin f_1^{-1}(y)\}$ and on $\{y : A^{\mathcal{O}_{f_2}}(y) \notin f_2^{-1}(y)\}$. This holds as these images are explicitly given by $\mathcal{M}(\cdot)$ since they are in $\mathcal{Y} \setminus S$. Therefore, there must be a value y, which they both invert, such that

$$A^{\mathcal{O}_{f_1}}(y) = {f_1}^{-1}(y) \neq {f_2}^{-1}(y) = A^{\mathcal{O}_{f_2}}(y) \ . \tag{5.1}$$

Let y be the lexicographically first element for which (5.1) holds. Define $x_1 \stackrel{\text{def}}{=} A^{\mathcal{O}_{f_1}}(y)$ and $x_2 \stackrel{\text{def}}{=} A^{\mathcal{O}_{f_2}}(y)$.

Proposition 51. *For both f_1 and f_2 there is an iteration during the run of the algorithm* Buildimage*(f_1) (resp.,* Buildimage*(f_2)) during which y is added to S.*

Proof: Otherwise, we reach a contradiction to the assumption that $\mathcal{M}(f_1) = \mathcal{M}(f_2)$ as the value of y is explicitly given by $\mathcal{M}$. ∎

Any difference in the computation of $A^{\mathcal{O}_{f_1}}(y)$ and $A^{\mathcal{O}_{f_2}}(y)$ may only stem from different answers to some oracle query. Let q be an oracle query on which

the computations differ, that is, for a query q made in both computations and $a_1 \neq a_2 \in \mathcal{R}$ we have:

$$a_1 = \mathcal{O}_{f_1}(q) \neq \mathcal{O}_{f_2}(q) = a_2 \ . \tag{5.2}$$

By Proposition 46 (for f_1, f_2 and their corresponding determining sets $\mathcal{I}^{\mathcal{O}}$), we have that there exists a value $x^* \in \mathcal{I}^{\mathcal{O}}_{f_1,q} \cup \mathcal{I}^{\mathcal{O}}_{f_2,q}$ for which

$$y_1 = f_1(x^*) \neq f_2(x^*) = y_2 \ , \tag{5.3}$$

as otherwise, (5.2) cannot hold.

The next two propositions yield that for such an $x^* \in \mathcal{I}^{\mathcal{O}}_{f_1,q} \cup \mathcal{I}^{\mathcal{O}}_{f_2,q}$ if $x^* \in \mathcal{I}^{\mathcal{O}}_{f_1,q}$ (resp., $x^* \in \mathcal{I}^{\mathcal{O}}_{f_2,q}$) it holds that $f_1(x^*) = y$ (resp., $f_2(x^*) = y$). In particular, combining this with Equation (5.3) and the fact that f_1 and f_2 are permutations asserts that in each of the sets $\mathcal{I}^{\mathcal{O}}_{f_1,q}$ and $\mathcal{I}^{\mathcal{O}}_{f_2,q}$ there is at most one such x^* . Let us assume that $x^* \in \mathcal{I}^{\mathcal{O}}_{f_1,q}$.

Proposition 52. *It holds that $y_1 \leq y$.*

Proof: Assuming otherwise (that $y_1 > y$), consider the execution of the algorithm Buildimage(f_1). In such a case, recall that by Proposition 51 y is added to the set S, and in line (6) of the algorithm y_1 is added to R (as it is in the image of the determining set for query q). Subsequently, in line (9) y_1 is removed from T. Since at this point of the execution of Bulidimage(f_1) y_1 is not in S, it is never added to S, hence $\mathcal{M}(f_1)$ contains (x^*, y_1), which contradicts $\mathcal{M}(f_1) = \mathcal{M}(f_2)$. ∎

Proposition 53. *It holds that $y_1 \geq y$ and $y_2 \geq y$.*

Proof: We prove that $y_1 \geq y$ and observe that the proof for $y_2 \geq y$ follows from symmetry. Assuming otherwise (that $y_1 < y$), we consider two cases: If (x^*, y_1) appears in $\mathcal{M}(f_1)$ or $(f_2^{-1}(y_1), y_1)$ appears in $\mathcal{M}(f_2)$, then since both are permutations, combining with (5.3) we reach a contradiction for $\mathcal{M}(f_1) = \mathcal{M}(f_2)$. Otherwise, both $A^{\mathcal{O}_{f_1}}(y_1) = x^*$ and $A^{\mathcal{O}_{f_2}}(y_1) = f_2^{-1}(y_1)$ must hold, and moreover y_1 is added to S at each of the corresponding Buildimage runs. Observe now that this contradicts the minimality of y. ∎

It follows that: For any element x in $\mathcal{I}^{\mathcal{O}}_{f_1,q} \cup \mathcal{I}^{\mathcal{O}}_{f_2,q}$ on which f_2 and f_1 do not agree it holds that $f_1(x) = y$ if $x \in \mathcal{I}^{\mathcal{O}}_{f_1,q}$ and that $f_2(x) = y$ if $x \in \mathcal{I}^{\mathcal{O}}_{f_2,q}$.

Moreover, since f_1, f_2 are permutations, we know that there is at most one such element in each of the sets, and in at least one of them such an element exists.

Next, if there exists exactly one such element, then, as before, without loss of generality we assume that there exists $x^* \in \mathcal{I}^{\mathcal{O}}_{f_1,q}$ for which $f_1(x^*) = y$. If there are two elements, i.e., $x^* \in \mathcal{I}^{\mathcal{O}}_{f_1,q}$ and $x^{**} \in \mathcal{I}^{\mathcal{O}}_{f_2,q}$, then by the definition of determining sets we know that adapting one of the functions to agree with the other changes the value of the oracle on it.

In both cases we get that (without loss of generality) we may assume that adapting f_2 to agree with f_1 on $\mathcal{I}^{\mathcal{O}}_{f_1,q}$ changes $\mathcal{O}_{f_2}(q)$. Recall that if there is exactly one such element, this follows from Proposition 45.

Now, note that the image adaptation of f_2 to f_1 on $\mathcal{I}^{\mathcal{O}}_{f_1,q}$ is just $f_{2(y_2,y)}$ and therefore $\mathcal{O}_{f_{2(y_2,y)}}(q) \neq \mathcal{O}_{f_2}(q)$, and by defintion of the bending sets it must hold that $y_2 = f_2(x^*) \in \mathcal{B}^{\mathcal{O}}_{f_2,q,y}$. Observe that at step (4) of the execution of Buildimage(f_2), after y' is set to y from T, in line (7) y_2 is added to R (by the definition of the determining sets and in line (9) y_2 is removed from T. Propositions 52 and 53 establish that $y_2 > y = y_1$. Therefore, at this point of the execution y_2 is not in S and hence never added to S. Finally, this means that (x^*, y_2) appears in $\mathcal{M}(f_2)$ which contradicts $\mathcal{M}(f_1) = \mathcal{M}(f_2)$. ∎

Claim 54. *The set S generated by the algorithm Buildimage(f) contains at least b elements.*

Proof: We start by showing that for all the functions f which the circuit inverts well, the generated set S in algorithm Buildimage(f) is large. Let f be a function for which $\Pr_{x \xleftarrow{r} \mathcal{X}}[A^{O_f} \text{ inverts } f(x)] > \frac{1}{c}$. By the definition of the set T at the begining of the run of Buildimage(f), T is of size at least $\frac{|\mathcal{X}|}{c}$. Observe that during each iteration of the algorithm one element is added to S and at most $(c \cdot t + c \cdot t + 1)$ elements are removed from T, where the first two summands correspond the size of the relevant bending set and determining set for each of the c potential calls to the oracle, and the third to the element y (lines 6 - 9 in the algorithm). It follows that when the algorithm terminates, the set S contains at least $\frac{|\mathcal{X}|}{c^2 \cdot (2 \cdot t + 1)} > \frac{2^n}{3 \cdot c^2 \cdot t} = b$ elements. ∎

We note that it is possible to encode $\mathcal{M}(f)$ by describing the set S, the set $\mathcal{Y} \setminus f(S)$, and an ordering on the set $\mathcal{Y} \setminus f(S)$, which by what we have shown has size at most $2^n - b$. Therefore, the size of the range of the mapping $\mathcal{M}$ is at most $\left(\binom{2^n}{b}\right)^2 \cdot ((2^n - b)!)$. Combining this with the first claim, we

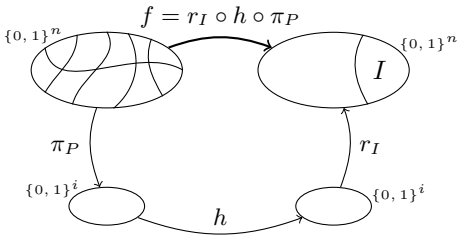

Figure 5.2: Decomposition of a regular function f: π_P is a regular function onto $\{0,1\}^i$ induced by a partition P of $\{0,1\}^n$, h is a permutation on $\{0,1\}^i$, and r_I is injective from $\{0,1\}^i$ into I.

infer that there are at most $\left(\binom{2^n}{b}\right)^2 \cdot ((2^n - b)!)$ such functions. The lemma is proved. ∎

Our next goal is to extend the encoding lemma to the case where $\mathcal{F} \subseteq \mathcal{R}_{n,i}$. We observe that the following process, which consists of 3 independent random choices, samples a uniform random f from from $\mathcal{R}_{n,i}$:

1. **Partition of the domain**: Sample a random uniform partition P of the domain into subsets of size 2^{n-i}. The partition P is chosen in an unordered manner, and then we use some order relation[6] ϕ on all the subsets of size 2^{n-i}. For $x \in P_i$, where $P_i \in P$, we define $\pi_P : \{0,1\}^n \to \{0,1\}^i$ by $\pi_P(x)$ as the order function of P_i in P according to ϕ [7] .
2. **A permutation on** $\{0,1\}^i$: Sample a random permutation h from $\mathcal{P}_i$.
3. **Image determination**: Sample a set $I \subset \{0,1\}^n$ of size 2^i and set $r_I : \{0,1\}^i \to \{0,1\}^n$ as the inverse function of the lexicographical order function[8] of I.
4. Finally, set $f \stackrel{\text{def}}{=} r_I \circ h \circ \pi_P$.

[6] For example, the order relation induced by the minimal element of the sets. That is, $S_1 < S_2$ if and only if $\min_{x \in S_1} < \min_{x \in S_2}$.

[7] Continuing the example, for the relation described in Footnote 6, we always have that for all $x \in S$, where $S \in P$ contains the all zero string 0^n, that $\pi_P(x)$ is the all zero string 0^i.

[8] Of course, any order relation would work just as well.

Indeed, the reader can verify that the mapping $(P,h,I) \mapsto r_I \circ h \circ \pi_P$ is an injective function onto $\mathcal{R}_{n,i}$. In particular, it holds that $|\mathcal{R}_{n,i}| = \left(\frac{(2^n)!}{((2^{n-i})!)^{2^i}\cdot(2^i!)}\right)\cdot((2^i)!)\cdot\binom{2^n}{2^i}$, where the factors in the product correspond to the number of partitions of $\{0,1\}^n$ to sets of size 2^i, the number of permutations on $\{0,1\}^i$ and the number of subsets of size 2^{n-i} from $\{0,1\}^n$, respectively.

For a fixed P and I (of the matching size), we denote by $\mathcal{R}_{n,i}(P,I)$ the subset of $R_{n,i}$ with partition P and image set I. The sampling process establishes a natural bijection $f : \mathcal{P}_i \to \mathcal{R}_{n,i}(P,I)$, where $f[h] \stackrel{\text{def}}{=} r_I \circ h \circ \pi_P$. Similarly, we denote the inverse transformation by $h[f]$. We extend the definition of $f[h]$ to the entire set $\{0,1\}^i \to \{0,1\}^i$ with $f[h] : (\{0,1\}^i \to \{0,1\}^i) \to \mathcal{F}_{n,i}(P,I)$ given by $f[h] \stackrel{\text{def}}{=} r_I \circ h \circ \pi_P$, where $\mathcal{F}_{n,i}(P,I) \stackrel{\text{def}}{=} \{\pi_P \circ h \circ r_I\}_{h\in\{0,1\}^i\to\{0,1\}^i}$. That is, $\mathcal{F}_{n,i}(P,I)$ is the set of all functions f on n bits with image set contained in I, such that if for $x,x' \in \{0,1\}^n$ it holds that $\pi_P(x) = \pi_P(x')$ then $f(x) = f(x')$.

Therefore, it is not surprising that any oracle circuit $A^{(?)}$ with access to a t-stable deterministic parameterized oracle for a class of functions $\mathcal{F} \subset \mathcal{R}_{n,i}(P,I)$ does not invert many functions from $\mathcal{F}$. This is formalized in the next lemma:

Lemma 55. *Let $n \geq i \geq 0$, P be a fixed partition of $\{0,1\}^n$ into subsets of size 2^{n-i}, and $I \subset \{0,1\}^n$ an image set of size 2^i. Let $A^{(?)}$ be an oracle circuit making at most c calls to its oracle, and let $\mathcal{O} = \{\mathcal{O}_f\}_{f\in\mathcal{F}_{n,i}(P,I)}$ be a deterministic parameterized oracle such that for a class of functions $\mathcal{F} \subseteq \mathcal{R}_{n,i}(P,I)$ it is t-stable with sets $(\mathcal{I}^{\mathcal{O}},\mathcal{B}^{\mathcal{O}})$. Then for at most $d_i(c,t)$ of the functions $f \in \mathcal{F}$, where d_i is as*[9] *in Lemma 49, it holds that $\Pr_{x\xleftarrow{r}\{0,1\}^n}\left[A^{\mathcal{O}_f}(f(x)) \in f^{-1}(f(x))\right] > \frac{1}{c}$.*

The proof works by reduction to the setting of Lemma 49. To this end, we define a deterministic parameterized oracle $\tilde{\mathcal{O}}$, a set $\tilde{\mathcal{F}} \subset \mathcal{P}_i$ of permutations with corresponding determining and bending sets, and show how to use $A^{(?)}$ to derive an algorithm $\tilde{A}^{(?)}$ that inverts many permutations from $\tilde{\mathcal{F}}$ which contradicts Lemma 49.

Proof: Towards contradiction, let $\mathcal{O}$ be a deterministic parametrized oracle, where $\mathcal{O}_f : \mathcal{D} \to \mathcal{R}$ defined for all $f \in \{0,1\}^n \to I$, such that for $\mathcal{F} \subset \mathcal{R}_{n,i}(P,I)$ it is t-stable with sets $(\mathcal{I}^{\mathcal{O}},\mathcal{B}^{\mathcal{O}})$, and $A^{(?)}$ an oracle circuit

[9] Note that here we substitute n for i.

that inverts well (as before inverts well f means inverting at least a $\frac{1}{c}$-fraction of the inputs to f) more than d_i functions from $\mathcal{F}$.

We define $\tilde{\mathcal{O}} = \{\tilde{\mathcal{O}}_h\}_{h\in\{0,1\}^i\to\{0,1\}^i}$, a deterministic parameterized oracle, where $\tilde{\mathcal{O}}_h : \mathcal{D} \to \mathcal{R}$ is given by $\tilde{\mathcal{O}}_h(q) \stackrel{\text{def}}{=} \mathcal{O}_{f[h]}(q)$, $\tilde{\mathcal{F}} \stackrel{\text{def}}{=} \{h[f] : f \in \mathcal{F}\}$, and for every $h \in \tilde{\mathcal{F}}$, $q \in \mathcal{D}$ and every $y' \in \{0,1\}^i$ set $\mathcal{I}^{\tilde{\mathcal{O}}}_{h,q} \stackrel{\text{def}}{=} \pi_P\left(\mathcal{I}^{\mathcal{O}}_{f[h],q}\right)$ and $\mathcal{B}^{\tilde{\mathcal{O}}}_{h,q,y'} \stackrel{\text{def}}{=} {r_I}^{-1}\left(\mathcal{B}^{\mathcal{O}}_{f[h],q,r_I(y')}\right)$.

By the construction, the definition of determining sets, bending sets and the definition of image adapting a function to agree with another function on a set, one can check that the new constructed sets are indeed determining sets and bending sets for $\tilde{\mathcal{O}}$. Moreover, as r_I is an injective function it holds that $\mathcal{B}^{\tilde{\mathcal{O}}}$ is a t-bending set for $\tilde{\mathcal{F}}$. For every $h \in \{0,1\}^i$ and $q \in \mathcal{D}$ it holds that

$$\left|\pi_P(\mathcal{I}^{\mathcal{O}}_{f,q})\right| \leq \left|\mathcal{I}^{\mathcal{O}}_{f,q}\right| \leq t,$$

and so $\mathcal{I}^{\tilde{\mathcal{O}}}$ is a t-determining set[10]. Therefore, the oracle is t-stable for $\tilde{\mathcal{F}}$.

We construct a circuit $\tilde{A}^{(?)}$ such that for every function $h \in \tilde{\mathcal{F}}$ and and every image $y' \in \{0,1\}^i$ it holds that $\tilde{A}^{(\tilde{\mathcal{O}}_h)}(y') = h^{-1}(y')$ whenever $A^{(\mathcal{O}_{f[h]})}(r_I(y')) \in (f[h])^{-1}(r_I(y'))$.

The circuit $\tilde{A}$ stores the complete descriptions of π_P and r_I. On input $y' \in \{0,1\}^i$ it computes $y \stackrel{\text{def}}{=} r_I(y')$ and simulates the run of $A^{(\mathcal{O}_f)}(y)$. Whenever in the simulation A issuses a query q to its oracle, $\tilde{A}$ queries its oracle and uses the answer as an answer for the simulation. When the simulation terminates with output $x = A(y)$, $\tilde{A}$ outputs $x' = \pi_P(x)$.

By the construction of the oracle $\tilde{\mathcal{O}}$ it follows that $\tilde{A}$ inverts well exactly the same number of functions as A, and so we reach a contradiction to Lemma 49. ∎

We next show that any two oracle circuit making few queries to two deterministic parametrized oracles, where both oracles are t-stable for some set of functions $\mathcal{F}$, does not invert well many functions from $\mathcal{F}$.

Lemma 56. *Let $n \geq i \geq 0$, P be a fixed partition of $\{0,1\}^n$ into subsets of size 2^{n-i}, and $I \subset \{0,1\}^n$ an image set of size 2^i. Let $A^{(?,?)}$ be an oracle circuit making a total number of at most c calls to its oracles, and let $\mathcal{O}^{(0)} = \{\mathcal{O}^{(0)}_f\}_{f\in\mathcal{F}_{n,i}(P,I)}$ and $\mathcal{O}^{(1)} = \{\mathcal{O}^{(1)}_f\}_{f\in\mathcal{F}_{n,i}(P,I)}$ be deterministic*

[10]Note that in both inequalities an equality may hold.

parameterized oracles such that for a class of functions $\mathcal{F} \subseteq \mathcal{R}_{n,i}(P,I)$ *both oracles are t-stable with sets* $(\mathcal{I}^{\mathcal{O}^{(0)}}, \mathcal{B}^{\mathcal{O}^{(0)}})$ *and* $(\mathcal{I}^{\mathcal{O}^{(1)}}, \mathcal{B}^{\mathcal{O}^{(1)}})$*, respectively. Then for at most* $d_i(c,t)$ *of the functions* $f \in \mathcal{F}$*, where* d_i *is as in Lemma 49, it holds that* $\Pr_{x \xleftarrow{r} \{0,1\}^n}\left[A^{\mathcal{O}_f^{(0)}, \mathcal{O}_f^{(1)}} \in f^{-1}(f(x))\right] > \frac{1}{c}$.

The proof is straight forward and very similar to the proof of Lemma 55. We show how to reduce this case to the setting of Lemma 55. In this case we define a new parametrized oracle $\tilde{\mathcal{O}}_f$ that on query $q' = (b,q) \in (\{0\} \times \mathcal{D}^{(0)}) \cup (\{1\} \times \mathcal{D}^{(1)})$, where $b \in \{0,1\}$ and $\mathcal{O}_f^{(b)} : \mathcal{D}^{(b)} \to \mathcal{R}^{(b)}$, answers $\tilde{\mathcal{O}}_f(q') \stackrel{\text{def}}{=} \mathcal{O}_f^{(b)}(q)$. It follows readily that $\tilde{\mathcal{O}}$ is t-stable. As before, any oracle circuit $A^{(?,?)}$ that inverts well more that d_i functions from $\mathcal{F}$ when plugged with $\mathcal{O}^{(0)}$ and $\mathcal{O}^{(1)}$ can be used to construct a circuit $\tilde{A}^{(?)}$ that inverts well more than d_i functions when plugged with $\tilde{\mathcal{O}}$.

We are now ready to prove the main theorem of this section:

Theorem 57 (Black-Box Separation Factory)**.** *Let* $s = s(\rho)$ *be a security function, and* $p = p(\rho)$ *be a polynomial function. Let* $(G,A) = (G^{(?)}, A^{(?,?)})$ *be a uniform oracle algorithm and an s-non-uniform two-oracle algorithm, respectively. Let* $\mathcal{F} = \{\mathcal{F}_\rho\}_{\rho>0}$*, where* $\mathcal{F}_\rho \subset \mathcal{R}_{n(\rho),i(\rho)}(P_\rho, I_\rho)$*, be contained in* $F_{\mathbf{OWF}}$*, and* $\mathcal{O} = \{\mathcal{O}_\rho\}_{\rho>0}$*, where* $\mathcal{O}_\rho = \{\mathcal{O}_{\rho,f}\}_{f \in \mathcal{F}_{n(\rho),i(\rho)}(P_\rho,I_\rho)}$*, such that for all large enough* ρ*:*

1. $\mathcal{O}_{\rho,f}$ $(\mathbf{Q}, \frac{1}{p(\rho)})$*-breaks* $g_\rho^{(f)}$ *for every* $f \in \mathcal{F}_\rho$*, where* $g_\rho^{(?)} \stackrel{\text{def}}{=} G(1^\rho, n, n')$.
2. $\mathcal{O}_\rho$ *is* t*-stable with sets* $(\mathcal{I}^{\mathcal{O}}, \mathcal{B}^{\mathcal{O}})$ *for* $\mathcal{F}_\rho$ *such that* $2^{s(\rho)} \cdot d_i(s(\rho), t) < |\mathcal{F}_\rho|$ *holds, where* d_i *is as in Lemma 49.*

Then (G,A) *is not an s-weak fixed-parameter fully black-box construction of* $\mathbf{Q}$ *from* $\mathbf{OWF}$*.*

Proof: Towards contradiction, assume that (G,R) is an s-non-uniform fixed-parameter fully black-box construction of $\mathbf{Q}$ from $\mathbf{OWF}$. We explain how to construct a family of functions f' that will be used to contradict the security assumption on our construction. As before, we set $\mathcal{E}_\rho$ is the evaluation oracle for $\mathcal{R}_{n(\rho),i(\rho)}$, and recall that we have shown that $\mathcal{E}_\rho$ is 1-stable for $\mathcal{R}_{n(\rho),i(\rho)}$. Combining this with the first condition and using Lemma 56 we have that for every fixed advice $a \in \{0,1\}^{s(\rho)}$, the two-oracle circuit $R_{\rho,a}^{(?,?)}$ inverts

with probability greater than $\frac{1}{s(\rho)}$ at most $d_i(s(\rho), t)$ of the functions in $\mathcal{F}_\rho$. Therefore, by the union bound there are at most $2^{s(\rho)} \cdot d_i(s(\rho), t)$ functions $f \in \mathcal{F}_\rho$, for which there exists an advice a, such that the circuit inverts more than a $\frac{1}{s}$-fraction of the inputs to f. The condition on the size of $\mathcal{F}_\rho$ gives a a function $f'_\rho \in \mathcal{F}_\rho$, such that for every $a \in \{0,1\}^{s(\rho)}$ it holds that $R_{\rho,a_\rho}^{(\mathcal{O}_{f'_\rho}, \mathcal{E}_{f'_\rho})}$ does not $(\mathbf{OWF}, \frac{1}{s(\rho)})$-break f'_ρ. The first condition, which holds for all the functions in $\mathcal{F}_\rho$, gives that $\mathcal{O}_{\rho,f_\rho}$ $(\mathbf{Q}, \frac{1}{p(\rho)})$-breaks f'_ρ.

Now, set $f' = \{f'_\rho\}_{\rho \in \mathbb{N}^+}$ and $B = \{\mathcal{O}_{\rho,f'_\rho}\}_{\rho \in \mathbb{N}^+}$. By the fact that $\mathcal{F}$ is contained in $F_{\mathbf{OWF}}$, we have that $f' \in F_{\mathbf{OWF}}$. By our assumption on (R, G) it holds that $G^{(f')} \in \mathbf{Q}$, and by what we have shown, it also holds that B $\mathbf{Q}$-breaks $G^{(f')}$. Lastly, (again) by our construction of f', it holds that $R^{[B,f']}$ does not $(\mathbf{OWF}, \frac{1}{s})$-break $G^{(f')}$, which contradicts our assumption on (G, R). ∎

Chapter 6

On the Limits of Black-Box Constructions of a Universal One-Way Hash Function

6.1 Introduction

UOWHFs are a fundamental cryptographic primitive, most notably used for obtaining digital signatures. They were studied extensively since their introduction by Naor and Yung [NY89], who showed a simple construction that makes only one call to the underlying one-way function whenever, additionally, the function is a permutation. Rompel [Rom90] showed a construction based on any one-way function, and the most efficient construction based on general one-way functions is due to Haitner et al. [HHR$^+$10]. Their construction makes $\tilde{O}(n^6)$ calls to a one-way function $f : \{0,1\}^n \to \{0,1\}^n$.

In this chapter we prove our main application of our framework from Chapter 5. We show a lower bound on the number of calls made by the construction algorithm G in any fully black-box construction (G, R) of **UOWHF** from **OWF**. Our bound is achieved by showing a sequence of efficient fixed-parameter fully black-box constructions, where each primitive is constructed from the one that precedes it, and by proving the lower bound on the number of calls a construction makes on the last primitive. A diagram of the reduction sequence is depicted in Figure 6.1.

6.1.1 Contributions of this Chapter

In Section 6.3 we prove a lower bound of $\Omega(n/\log^3(n))$ on the number of calls made by any fully black-box construction of a universal one-way hash function

(UOWHF) from a one-way function $f : \{0,1\}^n \to \{0,1\}^n$. The bound is improved in Section 6.4 to $\Omega(n/\log(n))$. Prior to our work it would have been possible to conjecture that there exists a construction of a UOWHF from a general one-way function that makes only one call to the underlying one-way function. Our bound matches up to a log factor the number of calls made by the constructions of [BM12] and [AGV12], and exactly the number of calls made by the very recent construction of [YGLW14].

Our result can be understood as an analog to that of Holenstein and Sinha, who show a bound of $\Omega(n/\log(n))$ on the number of calls to a one-way function that are made by a construction of a pseudo-random generator. We observe (details are omitted) that the recent result of [HS12] can be explained in our framework.

6.2 Preliminaries

We would make use of the following well-known lemma on probability spaces:

Lemma 58. *Let (X,Y) be jointly distributed random variables taking values from some set $(\mathcal{X},\mathcal{Y})$, and let $A : \mathcal{X} \times \mathcal{Y} \to \{0,1\}$ such that $\Pr_{x \xleftarrow{r} X, y \xleftarrow{r} Y}[A(x,y) = 1] \geq p > 0$. Then for all $\alpha, \beta > 0$ satisfying $\alpha + (1 - \alpha \cdot p) \cdot \beta < 1$ it holds that*

$$\Pr_{x \xleftarrow{r} X}\left[\Pr_{y \xleftarrow{r} Y}\left[A(x,y) = 1\right] \geq \beta \cdot p\right] \geq \alpha \cdot p \ . \tag{6.1}$$

Proof: Assume otherwise, we have that

$$\begin{aligned}
& \Pr_{X,Y}\left[A(X,Y)\right] \\
= \ & \Pr_{X,Y}\left[A(X,Y) \mid \Pr_{Y}[A(X,Y)] < \beta \cdot p\right] \cdot \Pr_{X}\left[\Pr_{Y}[A(X,Y)] < \beta \cdot p\right] \\
+ \ & \Pr_{X,Y}\left[A(X,Y) \mid \Pr_{Y}[A(X,Y)] \geq \beta \cdot p\right] \cdot \Pr_{X}\left[\Pr_{Y}[A(X,Y)] \geq \beta \cdot p\right] \\
< \ & (\beta \cdot p) \cdot (1 - \alpha \cdot p) + 1 \cdot (\alpha \cdot p) < p \ ,
\end{aligned}$$

where we write $A(X,Y)$ for the event that $A(X,Y) = 1$, which contradicts the assumption on (α, β). ∎

6.3 A Lower Bound on the Number of Calls of a Fixed-Parameter Fully Black-Box Construction of UOWHF from OWF

6.3.1 A Characterization of Universal One-Way Hash Functions

Loosely speaking, a universal one-way hash function is a keyed compressing function for which the probability that an adversary wins the following game is very small: First the adversary chooses a preimage v. Then a random key for the UOWHF is chosen. Finally, the adversary "wins" the game he finds a different preimage v' that maps to the same value under the chosen key. Formally:

Definition 59 (UOWHF). *A* universal one-way hash function $h = \{h_\rho\}_{\rho\in\mathbb{N}^+}$ *is a family of uniformly efficiently computable keyed functions* $h_\rho : \{0,1\}^{\kappa(\rho)} \times \{0,1\}^{m(\rho)} \to \{0,1\}^{m'(\rho)}$ *with* $m'(\rho) < m(\rho)$ *such that for any pair of efficient randomized algorithms* (B_1, B_2) *the function mapping* ρ *to*

$$\Pr_{\substack{(v,\sigma)\xleftarrow{r}B_1(\rho)\\ k\xleftarrow{r}\{0,1\}^{\kappa(\rho)}\\ v'\xleftarrow{r}B_2(k,v,\sigma)}}[h_\rho(k,v) = h_\rho(k,v') \wedge v \neq v']$$

is negligible. The family h *is an* ℓ-bit compressing *UOWHF, where* $\ell = \ell(\rho)$*, if* $m(\rho) - m'(\rho) \geq \ell(\rho)$ *for all large enough* ρ*.*

The primitive $\mathbf{UOWHF} = (F_{\mathbf{UOWHF}}, R_{\mathbf{UOWHF}})$ is defined implicitly analogously to the way **OWF** was defined for one-way functions (see Section 5.2.3). In this case, for every security parameter, the breaker (B_1, B_2) can be modeled as one combined circuit that corresponds to the Cook-Levin encoding of their relevant circuits, with an extra input bit that determines which one is actually computed. Similarly, for every $\ell = \ell(\rho)$ the primitive $\ell\text{-}\mathbf{UOWHF} = \langle F_{\ell\text{-}\mathbf{UOWHF}}, R_{\ell\text{-}\mathbf{UOWHF}}\rangle$ corresponds to ℓ-bit compressing universal one-way hash functions.

Domain extension of a UOWHF.

The definition of a UOWHF only guarantees that h_ρ is compressing (i.e., it is possible that $\ell(\rho) = 1$). The first reduction we use is a domain extension

of a UOWHF, that allows to construct an ℓ-bit compressing UOWHF from a UOWHF. Shoup [Sho00] shows a fully-black box construction of a ℓ-bit compressing UOWHF from one that compresses only one bit, which is the minimal requirement from any UOWHF.

Lemma 60 (UOWHF domain extension)**.** *There exists a fixed-parameter fully black-box construction of an ℓ-bit compressing UOWHF $h'_\rho : \{0,1\}^{\log(\ell)\cdot\kappa(\rho)} \times \{0,1\}^{m+\ell} \to \{0,1\}^m$ from a one-bit compressing UOWHF $h_\rho : \{0,1\}^{\kappa(\rho)} \times \{0,1\}^{m+1} \to \{0,1\}^m$. In order to evaluate h'_ρ the construction makes exactly $\ell(\rho)$ calls to h_ρ. The security reduction $R_\rho^{h_\rho,B}$ makes ℓ calls to its h_ρ oracle, and exactly one call to the breaker $B_\rho = (B_1, B_2)_\rho$ oracle. Furthermore, if B_ρ (ℓ-**UOWHF**, ϵ)-breaks h'_ρ, then the reduction (**UOWHF**, $\frac{\epsilon}{\ell}$)-breaks h_ρ.*

We observe that the security definition for UOWHFs involves an interaction, and allows the adversary to save its state using σ. It will be more convenient for us to work with an equivalent non-interactive version. The following definition of collision resistance is tightly related to that of a UOWHF by the lemma that follows it, where we denote by $a\|b$ the concatenation of a and b.

Definition 61 (RP-CRHF)**.** *A* random preimage collision resistance hash function *is an efficiently uniformly computable family of functions $h_\rho : \{0,1\}^{m(\rho)} \to \{0,1\}^{m'(\rho)}$ with $m'(\rho) < m(\rho)$, such that for every efficient randomized machine B the function mapping ρ to*

$$\Pr_{\substack{v \xleftarrow{r} \{0,1\}^{m(\rho)} \\ v' \xleftarrow{r} B(\rho, v)}} [h_\rho(v) = h_\rho(v') \wedge v \neq v']$$

*is negligible. The family h is an ℓ-*bit compressing *RP-CRHF, where $\ell = \ell(\rho)$, if additionally it holds that $m(\rho) - m'(\rho) \geq \ell(\rho)$ for all large enough ρ.*

The primitives **RP-CRHF** and $\log^2(\rho)$-**RP-CRHF** are defined analogously.

Lemma 62 (**UOWHF** to **RP-CRHF**, folklore)**.** *Let $h = \{h_\rho\}_{\rho\in\mathbb{N}^+}$ be a UOWHF. Then the family $h'_\rho : \{0,1\}^{\kappa(\rho)+m(\rho)} \to \{0,1\}^{\kappa(\rho)+m'(\rho)}$ given by $h'_\rho(k\|v) \stackrel{\text{def}}{=} (k\|h_\rho(k,v))$ is an RP-CRHF.*

Proof: The security definition of an RP-CRHF is trivially implied by that of a UOWHF. Suppose that we have an breaker B_ρ that finds a collision for a random preimage of $h'_\rho(k\|v)$. The reduction outputs the algorithms $(B_1, B_2)_\rho$

as follows: B_1 is the algorithm that chooses a uniform random $v \in \{0,1\}^{m(\rho)}$. Upon receiving a random key k, the reduction B_2 returns $B_\rho(k\|v)$. It is immediate that $(B_1, B_2)_\rho$ break the security of h as a UOWHF whenever B_ρ breaks the security of h' as a RP-CRHF. ∎

In particular, we observe that the proof of the lemma implies the existence of a fixed-parameter fully black-box construction of **RP-CRHF** from **UOWHF** that for every security parameter makes exactly one call to the universal one-way hash function. Additionally, the construction is security preserving in the strongest possible sense: If B_ρ $(\mathbf{RP\text{-}CRHF}, \epsilon)$-breaks the constructed h'_ρ (i.e., returns a collision on a uniform input v' with probability ϵ) then it holds that the reduction $(\mathbf{UOWHF}, \epsilon)$-breaks the underlying h_ρ. We mention that both the construction and the security reduction are uniform and efficient, and both make exactly one query to their oracles.

Pseudo-injective Functions.

Our last reduction establishes that padding the output of a $\log^2(\rho)$-**RP-CRHF** yields a primitive that is both a one-way function, and behaves like an injective function. A pseudo-injective function is an efficiently uniformly computable family $g = \{g_\rho\}_{\rho\in\mathbb{N}^+}$ of length preserving functions $g_\rho : \{0,1\}^{m(\rho)} \to \{0,1\}^{m(\rho)}$ such that for a uniformly chosen input $v \in \{0,1\}^{m(\rho)}$ it is impossible to find another input $v' \neq v$ such that both map to the same value under g_ρ. We stress that pseudo-injective functions exists unconditionally: Any permutation is a pseudo-injective function. Formally:

Definition 63 (Pseudo-Injectivity)**.** *A* pseudo-injective function $g = \{g_\rho\}_{\rho\in\mathbb{N}^+}$ *is a uniformly efficiently computable family of functions* $g_\rho : \{0,1\}^{m(\rho)} \to \{0,1\}^{m(\rho)}$*, such that for all uniform efficient algorithms* A *the function mapping* ρ *to*

$$\Pr_{\substack{v\xleftarrow{r}\{0,1\}^{m(\rho)}\\ v'\xleftarrow{r}A(1^\rho,v)}}[g_\rho(v') = g_\rho(v) \wedge v' \neq v]$$

is negligible.

Similarly to before, the primitive $\mathbf{PI} = \langle F_{\mathbf{PI}}, R_{\mathbf{PI}}\rangle$ corresponds to a pseudo-injective function. Next, we consider the primitive $\mathbf{OWF}\wedge\mathbf{PI}$ that corresponds to all functions which are both a one-way function and a pseudo-injective function. Formally, it holds that $f \in F_{\mathbf{OWF}\wedge\mathbf{PI}}$ if and only if $f \in F_{\mathbf{OWF}}$

and $f \in F_{\mathbf{PI}}$. For a breaker circuit C, a function $f_\rho \in f \in F_{\mathbf{OWF}\wedge\mathbf{PI}}$, and a number ϵ it holds that $\langle f_\rho, C, \epsilon\rangle \in R_{\mathbf{OWF}\wedge\mathbf{PI}}$ if and only if $\langle f_\rho, C, \epsilon\rangle \in R_{\mathbf{OWF}}$ *or* $\langle f_\rho, C, \epsilon\rangle \in R_{\mathbf{PI}}$.

It turns out that padding any $\log^2(n)$-**RP-CRHF** to a length-preserving function, yields a function which is both a one-way function and a pseudo-injective function.

Lemma 64 ($\log^2(\rho)$-**RP-CRHF** to **OWF**$\wedge$**PI**). *Let* $h = \{h_\rho\}_{\rho\in\mathbb{N}^+}$ *be an RP-CRHF that compresses* $\ell(\rho) \stackrel{\text{def}}{=} m(\rho) - m'(\rho)$ *bits, where* $\ell(\rho) \geq \log^2(\rho)$. *Then the family* $\{h'_\rho\}_{\rho\in\mathbb{N}^+}$, *where* $h'_\rho(v) \stackrel{\text{def}}{=} h_\rho(v)\|0^{\ell(\rho)}$, *is a one-way function and a pseudo-injective function.*

Proof: By the construction h'_ρ is always length-preserving and so h' implements a pseudo-injective and a one-way function. It is also clear that the construction is uniformly efficiently computable.

As for the security reduction, **OWF**$\wedge$**PI**-breaking h' implies either inverting h' on a uniformly random input, or finding a collision for one, for infinitely many security parameters ρ. This implies that for infinitely many security parameters ρ it is possible to break at least one of the properties.

First observe that any breaker B that breaks the pseudo-injectivity of h' immediately leads to one that **RP-CRHF**-breaks h with the same probability: On input $v' \in \{0,1\}^m$ return the output of $B(h'(v')) = B(h(v)\|0^{\ell(\rho)})$.

On the other hand, if B **OWF**-breaks h', we have that B inverts h' for infinitely many ρ's. We show that in this case, the algorithm, that on input $v \in \{0,1\}^{m(\rho)}$ returns $B(h_\rho(v))$, is the required reduction.

By our assumption B inverts h' on a random input with probability at least $\frac{1}{p}$, where p is a polynomial, for infinitely many security parameters. As h'_ρ compresses at least $\log^2(\rho)$ bits, we have that for a random v, the probability that $h'_\rho(v)$ has only one preimage is at most $2^{-\log^2(\rho)}$. Therefore, with probability at least $\frac{1}{p(\rho)} - \frac{1}{\rho^{\log(\rho)}} > \frac{1}{2p(\rho)}$ (for sufficiently large ρ) we have that B inverts $h'_\rho(v)$ and that $h'_\rho(v)$ has at least two preimages, in which case v is still uniform among $h_\rho^{-1}(h_\rho(v))$. Conitioned on this event it holds that with probability at least $\frac{1}{2}$ B returns a $v' \neq v$ that collides with v. Therefore we conclude that the reduction finds a random collision with probability at least $\frac{1}{4\cdot p(\rho)}$ for all sufficiently large ρ. ∎

The composition of the constructions depicted in Lemmas 60, 62 and 64 establishes a fixed-parameter fully black-box construction of an $h' \in F_{\mathbf{OWF}\wedge\mathbf{PI}}$

from any $h \in F_{\mathbf{UOWHF}}$ that makes $\log^2(\rho)$ calls to the underlying UOWHF. The security reduction makes $\log^2(\rho)$ calls (to both its oracles) in order to break the security of the underlying UOWHF, and if B $(\mathbf{OWF}\wedge\mathbf{PI}, \frac{1}{p})$-breaks the constructed h for some polynomial p and breaker B, then the reduction $(\mathbf{UOWHF}, \frac{1}{p'})$-breaks h, where p' is a different polynomial such that $p'(\rho) > 4 \cdot p(\rho) \cdot \log^2(\rho)$. Thus we obtain:

Corollary 65. *Suppose that (G, R) is an s'-weak fixed-parameter fully black-box construction of* **UOWHF** *from* **OWF** *that makes at most $r' = r'(\rho)$ queries to* **OWF**. *Then there exists an s-weak fixed-parameter fully black-box construction of* **OWF**$\wedge$**PI** *from* **OWF** *that makes $r'(\rho) \cdot \log^2(\rho)$ calls to the underlying one-way function, where $s(\rho) \stackrel{\text{def}}{=} s'(\rho) \cdot \log^2(\rho)$.*

Proof: The composition of G with the constructions described in Lemmas 60, 62 and 64 implements a fixed-parameter fully black-box construction of an $h' \in F_{\mathbf{OWF}\wedge\mathbf{PI}}$ from any $h \in F_{\mathbf{OWF}}$. The composition of the reduction algorithms described in the lemmas with the s'-non-uniform reduction makes at most s calls to both its oracles. It follows that for every B that **OWF**$\wedge$**PI**-breaks the constructed h', the reduction $R^{[B,h]}$ $(\mathbf{OWF}, \frac{1}{s(\rho)})$-breaks h.[1] ∎

Therefore, in order to show that there is no s'-weak fixed-parameter fully black-box construction of **UOWHF** from **OWF**, where the construction makes r' calls to the one-way functions, it is sufficient to show that there is no s-weak fixed-parameter fully black-box construction of **OWF**$\wedge$**PI** from **OWF** that makes r calls, where $s(\rho) \stackrel{\text{def}}{=} s'(\rho) \cdot \log^2(\rho)$ and $r(\rho) \stackrel{\text{def}}{=} r'(\rho) \cdot \log^2(\rho)$. This is the goal of the next section.

6.3.2 A Lower Bound on the Number of Calls for an s-Weak Fixed-Parameter Fully Black-Box Construction of OWF$\wedge$PI from OWF

As explained, a lower bound on a construction of **OWF**$\wedge$**PI** from **OWF** yields a very good (up to a $\log^2$-factor) bound on the construction of **UOWHF**. Our proof utilizes the machinery from Section 5.3. Let us introduce some

[1] Note that for a large enough security parameter it still holds that it inverts h with probability $\frac{1}{s'(\rho)}$, however, as each query in the reduction algorithm of the assumed s-weak construction results in up to $\log^2(\rho)$ queries to the composed one, we get a total number of at most s queries.

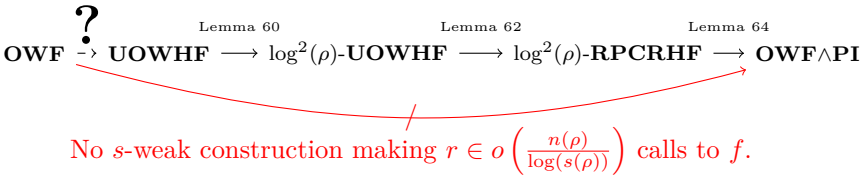

Figure 6.1: Fully Black-Box Constructions Diagram.

notation. For an (n,n)-oracle circuit $g^{(?)} : \{0,1\}^m \to \{0,1\}^m$, a function $f : \{0,1\}^n \to \{0,1\}^n$ and a value $v \in \{0,1\}^m$, denote by $X_g(f,v)$ and $Y_g(f,v)$ the sets of queries and answers made to and received from f during the evaluation of $g^{(f)}(v)$, respectively.

For any potential construction (G,R) denote by $r = r(\rho)$ the number of queries g_ρ makes when instantiated for security parameter ρ with a one-way function $f_\rho : \{0,1\}^\rho \to \{0,1\}^\rho$, that is we set $n(\rho) \stackrel{\text{def}}{=} n'(\rho) \stackrel{\text{def}}{=} \rho$. Additionally, let $s = s(\rho)$ be a super-polynomial security function smaller than $2^{\frac{\rho}{10}}$. I.e., for all polynomials p and all large enough ρ it holds that $p(\rho) < s(\rho) < 2^{\frac{\rho}{10}}$. We prove that if $r(\rho) < \frac{n(\rho)}{2000 \cdot \log(s(\rho))}$ holds for all large enough ρ, then (G,R) is not an s-weak fixed-parameter fully black-box construction of **OWF∧PI** from **OWF**.

Theorem 66. *For all super-polynomial security functions $s = s(\rho) < 2^{\frac{\rho}{10}}$ and $r = r(\rho)$ there is no s-weak fixed-parameter fully black-box construction of* **OWF∧PI** *from* **OWF** *such that $g_\rho^{(?)} : \{0,1\}^{m(\rho)} \to \{0,1\}^{m(\rho)}$ makes at most $r(\rho)$ calls to the underlying one-way function, where $n(\rho) \stackrel{\text{def}}{=} n'(\rho) \stackrel{\text{def}}{=} \rho$ and $g_\rho^{(?)} \stackrel{\text{def}}{=} G(1^\rho, n, n')$, and $r(\rho) \leq \frac{n(\rho)}{2000 \cdot \log(s(\rho))}$ holds for all large enough ρ.*

Proof: Without loss of generality, we assume that the construction g makes exactly $r(\rho) \stackrel{\text{def}}{=} \frac{n(\rho)}{2000 \cdot \log(s(\rho))}$ different queries. Whenever this is not the case, it is always possible to amend G so that it behaves exactly as before, but on input $(1^\rho, n, n')$ it outputs an (n,n')-oracle circuit with $r(\rho)$ oracle gates, and additionally, all queries are different.

Let s and r be a pair of security functions such that s is super-polynomial, that is, for every polynomial p and large enough ρ it holds that $s(\rho) > p(\rho)$,

and that $r(\rho) = \frac{n(\rho)}{2000 \cdot \log(s(\rho))}$ holds for all sufficiently large ρ.

We now explain how to construct the oracle $\mathcal{O} = \{\mathcal{O}_\rho\}_{\rho \in \mathbb{N}^+}$ and the collection of sets of functions $\mathcal{F} = \{\mathcal{F}_\rho\}_{\rho \in \mathbb{N}^+}$. For each security paramter ρ we define the oracle $\mathcal{O}_\rho$ and the set $\mathcal{F}_\rho$ *independently* of the oracles and function sets chosen for other security parameters. It will always hold that $\mathcal{F}_\rho \subset \{0,1\}^n \to \{0,1\}^n$, and so the constructed $\mathcal{F}$ is contained in $F_{\mathbf{OWF}}$.

Therefore, from now on we omit the security parameter in our notation, but formally all our parameters depend on the security parameter ρ. In particular, $g^{(?)}$ is the construction that the uniform construction algorithm G outputs for security parameter $\rho = n = n'$ with a function $f_\rho : \{0,1\}^\rho \to \{0,1\}^\rho$.

Analogously to [HS12], for every security parameter we break either the one-wayness property of the constructed function, or its pseudo-injectivity. For the oracle circuit $g^{(?)} : \{0,1\}^m \to \{0,1\}^m$, we check whether when g is evaluated with a random permutation $f \xleftarrow{r} \mathcal{P}_n$ and a random input $v \xleftarrow{r} \{0,1\}^m$, the output $g^f(v)$ is significantly correlated with any subset of the set of oracle answers returned by f on the calls made to it during the evaluation of $g^f(v)$ (recall that these are denoted by $Y_g(f,v)$). To this end, we bring the procedure STA (for safe to answer), which returns true if and only if there is no such correlation:

Procedure STA(w, Q) (on $w \in \{0,1\}^m$ and $Q \subset \{0,1\}^n$ of size r)

---

for all $B \subseteq Q$ **do**
. **if** $\Pr_{f' \xleftarrow{r} \mathcal{P}_n, v' \xleftarrow{r} \{0,1\}^m} \left[g^{(f')}(v') = w \,\middle|\, B \subseteq Y_g(f', v') \right] \geq 2^{-m + \frac{n}{30}}$
. **return** `false`
return `true`

We set $p(g)$, the probability that for a random permutation f and a random input v, the output $g^f(v)$ is correlated with some subset of the answers $Y_g(f,v)$. Define

$$p(g) \stackrel{\text{def}}{=} \Pr_{f \xleftarrow{r} \mathcal{P}_n, v \xleftarrow{r} \{0,1\}^m} \left[\text{STA}(g^{(f)}(v), Y_g(f,v)) \right] . \tag{6.2}$$

We stress that both the output of STA (for any value y and a set Q), and the value $p(g)$ do not depend on any specific permutation, but rather on a combinatorial property of the construction as a whole, which averages over all permutations.

As explained, we set the oracle $\mathcal{O}$ and the set $\mathcal{F}$ based on the value $p(g)$. In case that $p(g) > \frac{1}{2}$ we set the oracle $\mathcal{O} \stackrel{\text{def}}{=} \text{BreakOW}_g \stackrel{\text{def}}{=} \{\text{BreakOW}_{g,f}\}_{f\in\{0,1\}^n\to\{0,1\}^n}$, where we use the oracle BreakOW_g from [HS12], which is described next. In [HS12] it is implicitly proved that there exists a set $\mathcal{F} \subset \mathcal{P}_n$ of size $|\mathcal{F}| > \frac{|\mathcal{P}_n|}{5}$, such that $\text{BreakOW}_{g,f}$ $(\mathbf{OWF}, \frac{1}{4})$-breaks g^f for all $f \in \mathcal{F}$, and that BreakOW_g is $2^{\frac{n}{5}}$-stable for $\mathcal{F}$, in which case condition (2) in Theorem 57 is satisfied.

Algorithm $\text{BreakOW}_{g,f}(w)$ (on input $w \in \{0,1\}^m$)

for all $v \in \{0,1\}^m$ **do**
. **if** $g^{(f)}(v) = w$ **then**
. **if** $\text{STA}(w, Y_g(f, v))$ **then**
. **return** v
return $\perp$

In the case $p(g) \leq \frac{1}{2}$ we show that when f is chosen uniformly at random from a set of regular degenerate functions, it is often the case that the construction $g^{(f)}$ is not injective, and therefore there exists an oracle which breaks the pseudo-injectivity of $g^{(f)}$. The challenge is to find a breaker oracle that is t-stable. The next lemmas establish that the oracle BreakPI satisfies the required conditions in this case.

Formally, for a construction circuit g we define the oracle $\text{BreakPI}_g = \{\text{BreakPI}_{g,f}\}_{f\in\{0,1\}^n\to\{0,1\}^n}$ that for a function f is given by:

Algorithm $\text{BreakPI}_{g,f}(v)$ (on input $v \in \{0,1\}^m$)

for all $v' \in \{0,1\}^m$ **do**
. **if** $g^{(f)}(v) = g^{(f)}(v')$ **and** $v' \neq v$ **then**
. **if** $Y_g(f, v) = Y_g(f, v')$ **then**
. **return** v'
return $\perp$

Now, we fix $i \stackrel{\text{def}}{=} \frac{n}{200\cdot r} = 10 \cdot \log(s)$. We show that for a $\frac{1}{6}$-fraction of the functions f in $\mathcal{R}_{n,i}$ it holds that $\text{BreakPI}_{g,f}$ breaks the pseudo-injectivity of $g^{(f)}$.

Lemma 67. *Let $g : \{0,1\}^m \to \{0,1\}^m$ be an r-query oracle construction*

with $p(g) \leq \frac{1}{2}$. *Then for a* $\frac{1}{6}$*-fraction of the functions in* $\mathcal{R}_{n,\frac{n}{200\cdot r}}$ *it holds that*

$$\Pr_{v \xleftarrow{r} \{0,1\}^m}\left[\text{BreakPI}_{g,f}(v) \text{ outputs } v' \text{ s.t. } v \neq v' \wedge g^{(f)}(v) = g^{(f)}(v')\right] \geq \frac{1}{24}. \tag{6.3}$$

Proof:

By inspection of the algorithm BreakPI it holds that whenever $\text{BreakPI}_{g,f}(v) \neq \perp$ it outputs a value $v' \neq v$ such that $g^{(f)}(v) = g^{(f)}(v')$, and so it breaks the pseudo-injectivity of $g^{(f)}(\cdot)$. By the assumption, it holds that a random evaluation $\text{STA}(g^{(f)}(v), Y_g(f,v))$ on a random permutation f returns `false` with probability $(1 - p(g)) \geq \frac{1}{2}$. Next, as for every (n,n)-oracle circuit $D^{(?)}$ that does not take an input and has one output bit (called distinguisher) making at most r queries, it holds that

$$\left| \Pr_{f \xleftarrow{r} \mathcal{P}_n}[D^{(f)} = 1] - \Pr_{f \xleftarrow{r} \mathcal{R}_{n,i}}[D^{(f)} = 1] \right| \leq \frac{r^2}{2^i} \ ,$$

we obtain

$$\Pr_{f \xleftarrow{r} \mathcal{R}_{n,\frac{n}{200\cdot r}}, v \xleftarrow{r} \{0,1\}^m}\left[\neg\text{STA}(g^{(f)}(v), Y_g(f,v))\right] \geq (1 - p(g)) \cdot \left(1 - \frac{r^2}{2^{\frac{n}{200\cdot r}}}\right) \ . \tag{6.4}$$

The following counting argument shows that for every $f \in \mathcal{R}_{n,\frac{n}{200\cdot r}}$ there exists a set $\mathcal{W}_f = \mathcal{W}_f(\text{Im}\,(f))$ of size $|W_f| \leq 2^{m-\frac{n}{100}}$ such that

$$\Pr_{f \xleftarrow{r} \mathcal{R}_{n,\frac{n}{200\cdot r}}, v \xleftarrow{r} \{0,1\}^m}\left[g^{(f)}(v) \in \mathcal{W}_f\right] \geq (1 - p(g)) \cdot \left(1 - \frac{r^2}{2^{\frac{n}{200\cdot r}}}\right) > \frac{1}{3} \ , \tag{6.5}$$

for sufficiently large n.

Define $\mathcal{W}_f \stackrel{\text{def}}{=} \left\{g^{(f)}(v) : v \in \{0,1\}^m \wedge \neg\text{STA}(g^f(v), Y_g(f,v))\right\}$. Equation (6.5) follows readily from the definition of $\mathcal{W}_f$ and (6.4). As for $|\mathcal{W}_f|$, we have that there are less than $|\text{Im}\,(f)\,|^r = 2^{\frac{n}{200}}$ different possibilities for $Y_g(f,v)$ sets. For each of these sets there are 2^r subsets, and for each such subset S there are at most $2^{m-\frac{n}{30}}$ values w that satisfy

$\Pr_{f\xleftarrow{r}P_n, v\xleftarrow{r}\{0,1\}^m}\left[g^f(v)=w|S\subseteq Y_g(f,v)\right] > 2^{-m+\frac{n}{30}}$. In total, we have that $|\mathcal{W}_f| \leq 2^{m-\frac{n}{30}}\cdot 2^{\frac{n}{200}}\cdot 2^r < 2^{m-\frac{n}{100}}$. The existence of such a set $\mathcal{W}_f$ was observed in [HS12].

Now, By Lemma 58 (with $p=\frac{1}{3}, \alpha=\frac{1}{2}$ and $\beta=\frac{1}{2}$) it holds that a $\frac{1}{6}$-fraction of the functions $f\in\mathcal{R}_{n,\frac{n}{200\cdot r}}$ satisfy:

$$\Pr_{v\xleftarrow{r}\{0,1\}^m}\left[g^{(f)}(v)\in\mathcal{W}_f\right] > \frac{1}{6}\ . \tag{6.6}$$

Fix a function $f\in\mathcal{R}_{n,\frac{n}{200\cdot r}}$ for which (6.6) holds. We show that in this case $\text{BreakPI}_{g,f}$ returns a collision with probability at least $\frac{1}{30}$ for a randomly chosen v.

Define $\mathcal{W}'_f \stackrel{\text{def}}{=} \left\{w\in\mathcal{W}_f\ :\ |\left(g^{(f)}\right)^{-1}(w)| > \frac{1}{12}\cdot 2^{\frac{n}{100}}\right\}$. We claim that

$$\Pr_{v\xleftarrow{r}\{0,1\}^m}\left[g^{(f)}(v)\in\mathcal{W}'_f \,\middle|\, g^{(f)}(v)\in\mathcal{W}_f\right] \geq \frac{1}{2}\ . \tag{6.7}$$

Equation (6.6) asserts that there are at least $\frac{1}{6}\cdot 2^m$ elements v that are mapped to fewer than $2^{m-\frac{n}{100}}$ images. It follows that there are at most $\frac{1}{12}\cdot 2^{\frac{n}{100}}\cdot 2^{m-\frac{n}{100}} = \frac{1}{12}\cdot 2^m$ elements in $\left(g^{(f)}\right)^{-1}(\mathcal{W}_f)\backslash\left(g^{(f)}\right)^{-1}(\mathcal{W}'_f)$, which asserts the claim. Finally, for any $w\in\mathcal{W}'_f$, we claim that

$$\Pr_{v\xleftarrow{r}\{0,1\}^n}\left[\text{BreakPI}_{g,f}(v)\neq\perp \,\middle|\, g^{(f)}(v)=w\right] > \frac{1}{2}\ . \tag{6.8}$$

Conditioned on $g^{(f)}(v)=w$, we have that v is still uniform among $\left(g^{(f)}\right)^{-1}(w)$. Now, recall that $\text{BreakPI}_{g,f}$ fails to return a collision only when v is such that $Y_g(f,v)$ differs from the sets $Y_g(f,v')$, for all $v'\in\left(g^{(f)}\right)^{-1}(w)\backslash\{v\}$. Since f is degenerate, there are at most $|\text{Im}\,(f)\,|^r = 2^{\frac{n}{200}}$ different $Y_g(f,v)$ sets[2]. Therefore, $\text{BreakPI}_{g,f}(v)$ fails for at most $2^{\frac{n}{200}}$ of the elements of $\left(g^{(f)}\right)^{-1}(w)$, as for at most $2^{\frac{n}{200}}-1$ of them it can be the case that there is no other v' for which $Y_g(f,v)=Y_g(f,v')$. Since $w\in\mathcal{W}'_f$ we have that

[2] In fact, this shows that we could also require that BreakPI returns a value $v'\neq v$ such that the vector (now differentiating two y-answer sets according to the ordering of the answers) of query answers in the computation of $g^{(f)}(v)$ matches that of $g^{(f)}(v')$, but this is not needed in order to prove that BreakPI is t-stable.

$|(g^{(f)})^{-1}(w)| > 2^{\frac{n}{100}}$ and so $\text{BreakPI}_{g,f} \neq \perp$ with probability at least $\frac{1}{2}$. The claim follows. Combining our claims we obtain:

$$\begin{aligned}
&\Pr_{v \xleftarrow{r} \{0,1\}^m}[\text{BreakPI}_{g,f}(v) \neq \perp] \\
&\geq \Pr_v\left[\text{BreakPI}_{g,f}(v) \neq \perp \,\middle|\, g^{(f)}(v) \in \mathcal{W}'_f\right] \\
&\quad \cdot \Pr_v\left[g^{(f)}(v) \in \mathcal{W}'_f \,\middle|\, g^{(f)}(v) \in \mathcal{W}_f\right] \cdot \Pr_v\left[g^{(f)}(v) \in \mathcal{W}_f\right] \\
&\geq \frac{1}{12} \sum_{w \in \mathcal{W}'_f} \Big(\Pr_v\left[\text{BreakPI}_{g,f}(v) \neq \perp \,\middle|\, g^{(f)}(v) = w\right] \\
&\qquad\qquad \cdot \Pr_v\left[g^{(f)}(v) = w \,\middle|\, g^{(f)}(v) \in \mathcal{W}'_f\right] \Big) > \frac{1}{24} \ .
\end{aligned}$$

The lemma follows. ∎

We conclude from Lemma 67 that if $p(g) \leq \frac{1}{2}$, there exists a partition P of $\{0,1\}^n$ to sets of size 2^{n-i} and an image-set I of size 2^i, such that (6.3) holds for at least a $\frac{1}{6}$-fraction of the functions $f \in \mathcal{R}_{n,i}(P,I)$. Set $\mathcal{F} \subset \mathcal{R}_{n,i}(P,I)$ to be the set of all functions for which (6.3) holds. It follows that $|\mathcal{F}| \geq \frac{1}{6} \cdot 2^i$, as $|\mathcal{R}_{n,i}(P,I)| = |\mathcal{P}_i|$.

We next show that for the class of functions $\mathcal{R}_{n,i}(P,I)$ the oracle can be implemented such that it is stable.

Lemma 68. *Let $i \in \mathbb{N}^+$ and $I \subset \{0,1\}^n$ of size 2^i and P a partition of $\{0,1\}^n$ to sets of size 2^{n-i}. Then there exists an implementation of the oracle BreakPI_g that is n-stable for $\mathcal{R}_{n,i}(P,I)$.*

Proof: As we show next, when we limit the oracle to a fixed partition P and a fixed image I, it can be implemented such that it makes only few queries to f on every call to BreakPI and therefore enjoys stable sets.

We begin by describing the parsimonious implementation of the algorithm, i.e., an implementation that makes very few queries to f.

The parsimonious algorithm will have a description of the partition P and the image mapping I. On input v it simulates BreakPI_g as follows: In every iteration the BreakPI_g queries only $g^{(f)}(v)$. The algorithm records action of the f on the inputs of $X_g(f,v)$. That is, it records the values of $h[f]$ on $\pi_P(X_g(f,v))$.

It then tries to emulate the value $g^{(f)}(v')$ while assuming that f is in $\mathcal{R}_{n,i}(P,I)$. Whenever $g^f(v')$ issues a query x, the algorithm checks whether

it is mapped under the partition function π_P to the same value as one of the x queries issued before. I.e., it checks whether $\pi_P(x) = \pi_P(x')$ for some $x' \in X_g(P, I)$. In such a case it uses the recorded value of $f(x')$ and the emulation continues. If the algorithm gets stuck it decides that $Y_g(f, v) \neq Y_g(f, v')$ and continues to the next iteration. We first note that the algorithm is well defined for all $f \in \mathcal{F}_{n,i}(P, I)$.

Next, we observe that for a function $f \in \mathcal{R}_{n,i}(P, I)$ the parsimonious algorithm simulates BreakPI_g perfectly: If for some value v' the evaluation of $g^{(f)}(v')$ gets stuck on some query x, we know that the output would not return the value v' since in this case the function $h[f]$ is a permutation and r_I is injective it holds that $f(x) \notin Y_g(f, v)$. On the other hand, when the simulation of an iteration succeeds it follows that for every x it holds that $f(x)$ is computed correctly (and accordingly $g^{(f)}(v)$ and $Y_g(f, v')$). Again, this follows from the decomposition $f = r_I \circ h \circ \pi_P$.

We note that the algorithm is not guranteed to behave the same as BreakPI_g for an arbitrary $f \in \mathcal{F}_{n,i}(P, I)$, but we only care about its behavior for a regular f. It is now immediate that the implementation is r-stable: In a similar manner to the case of the evaluation oracle we set $\mathcal{I}^{\mathcal{O}}_{f,v} \stackrel{\text{def}}{=} X_g(f, v)$ and $\mathcal{B}^{\mathcal{O}}_{f,q,y} \stackrel{\text{def}}{=} Y_g(f, v)$ as determining and bending sets for BreakPI, respectively.

■

It is left to check that $2^s \cdot d_i(s, n) < |\mathcal{F}|$. We compute

$$\frac{d_i(s,n)}{|\mathcal{F}|} = \frac{\binom{2^i}{b}^2 \cdot (2^i - b)!}{(1/6) \cdot 2^i} = \binom{s^{10}}{b} \frac{1}{(1/6) \cdot b!} \leq \left(\frac{e^2 \cdot s^{10}}{b^2}\right)^b , \tag{6.9}$$

where

$$b = \frac{2^i}{3 \cdot s^2 \cdot n^2} = \frac{2^{\frac{n}{200 \cdot r}}}{3 \cdot s^2 \cdot n^2} = \frac{2^{10 \cdot \log(s)}}{3 \cdot s^2 \cdot n^2} > s^7, \tag{6.10}$$

since $s > 3 \cdot n^2$ for sufficiently large n. Combining (6.9) with (6.10) we conclude that $\frac{2^s \cdot d_i(s,n)}{|\mathcal{F}|} < 1$.

We have shown that the conditions of Theorem 57 hold, and therefore we conclude that there is no s-weak fixed-parameter fully black-box construction of **OWF∧PI** from **OWF**. The theorem is proved. ■

6.3.3 Deriving the Lower Bound

We are now ready to derive our lower bound for constructions of a universal one-way hash function from a one-way function:

Corollary 69. *Let s' be a security function such that $s(n) \stackrel{\text{def}}{=} s'(n) \cdot \log^2(n)$ is a super-polynomial security function for which $s(n) < 2^{\frac{n}{10}}$ holds. Then there is no s-weak fixed-parameter fully black-box construction of* **UOWHF** *from* **OWF**, *where the construction makes at most $r'(n) = \frac{n}{2000 \cdot \log(s(n)) \cdot \log^2(n)}$ calls to a one-way function $f = \{f_n : \{0,1\}^n \to \{0,1\}^n\}_{n \in \mathbb{N}^+}$.*

Proof: We apply Corollary 65 with Theorem 66. ∎

Corollary 70. *There is no fixed-parameter fully black-box construction of* **UOWHF** *from* **OWF**, *where the construction makes at most $r = r(n)$ calls to a one-way function $f = \{f_n : \{0,1\}^n \to \{0,1\}^n\}_{n \in \mathbb{N}^+}$, where $r \in o\left(\frac{n}{\log^3(n)}\right)$.*

Proof: Let $r \in o\left(\frac{n}{\log^3(n)}\right)$. Then there exists a super-constant function $\alpha = \alpha(n)$, such that the function $r'(n)$ given by $r'(n) \stackrel{\text{def}}{=} r(n) \cdot \alpha(n)$ is still in $o\left(\frac{n}{\log^3(n)}\right)$. The bound follows immediately from Corollary 69 applied with $s(n) \stackrel{\text{def}}{=} 2^{\alpha(n) \cdot \log(n)}$. ∎

6.4 A Tight Lower Bound for Fully Black-Box Constructions of UOWHF from OWF.

Corollary 70 establishes a lower bound of $\Omega(n/\log^3(n))$ calls for any fully black-box construction of **UOWHF** from a **OWF**. We now explain how to improve the lower bound by a $\log(n)$ factor.

Recall that the choice of $\log^2(\rho)$ in Lemma 64 was made such that any breaker for **OWF**∧**PI** can be translated to one that breaks the security of the **RP-CRHF**, i.e., finds a collision for a random preimage. We observe that the lemma is still correct if we start with a $\ell(\rho)$-**RP-CRHF**, for any super-logarithmic function $\ell(\rho) \in \omega(\log(\rho))$. That is, a compression by a super-logarithmic number of bits is sufficient for our proof to go through. In this case (as in the proof of Lemma 64) it still follows that at most a negligible fraction

of the inputs have at most one preimage, and therefore the reduction converts a breaker with noticeable success probability for **OWF∧PI** is converted to a breaker with a noticeable success probability for $\ell(\rho)$-**RP-CRHF**. Next, this improves Corollary 65 by a $\log(\rho)$ factor, which consequently improves our bound to $\Omega(n/\log^2(n))$ calls.

6.4.1 A Tight Lower Bound of $\Omega(n/\log(n))$ Calls

In this section we present the most general version of our lower bound. We consider once more the series of reductions used to derive Corollary 65. We show that it is possible to get rid of the last $\log(n)$ factor used in Lemma 64, which in turn results in a lower bound of $\Omega(n/\log(n))$ calls.

We do not know of a reduction that transforms a breaker with an inverse polynomial success probability of a **OWF∧PI** to a breaker with an inverse polynomial success probability for $O(1)$-**RP-CRHF** (i.e., a UOWHF that compresses a constant number of bits). Nevertheless, we can exploit the fact that both breakers presented in Section 6.3 perform better than merely breaking the primitive. That is, in order to derive the lower bound using Corollary 65 it is sufficient that the breaker in Theorem 66 breaks (with some inverse-polynomial probability). However, both BreakPI and BreakOW enjoy a stronger property: They both break the security of the **OWF∧PI** with constant probability (both break the candidate construction with probability at least 1/100) rather than with inverse-polynomial probability.

We observe that in this case the reduction algorithm presented in Lemma 64 gives a breaker with a constant success probability for $\ell(\rho)$-**RP-CRHF**, where $\ell(\rho) = 10$. Following the analysis in Lemma 64 we have that at most a 2^{-10}-fraction of the images have only one preimage, and using the union-bound the reduction breaks the $\ell(\rho)$-**RP-CRHF** with probability at least $1/100 - 1/2^{10} > 1/200$. In a similar manner to Section 6.4 this gives now a version of Corollary 65, that asserts the existence of an s-weak fixed-parameter fully black-box construction of **OWF∧PI** from **OWF** for breakers that are considered successful only if they break the security property of the **OWF∧PI** with constant probability:

Corollary 71. *Suppose that (G, R) is an s'-weak fixed-parameter fully black-box construction of* **UOWHF** *from* **OWF** *that makes at most $r' = r'(\rho)$ queries to* **OWF**. *Then there exists an s-weak fixed-parameter fully black-box construction of* **OWF∧PI** *from* **OWF** *that makes $10 \cdot r'(\rho)$ calls to the underlying one-way function, where $s(\rho) \stackrel{\text{def}}{=} 10 \cdot s'(\rho)$ and additionally, a*

potential breaker for **OWF∧PI** *is successful only if it breaks the security if property of the primitive with probability at least* $1/100$.

We stress that the restriction with respect to the constant success probability of the potential breaker in the corollary does *not* propagate to the constructions we rule out, and our lower bound holds with respect to the 'standard' (polynomial) security requirements. Applying Corollary 71 with Theorem 66 we derive a lower bound of $\Omega(n/\log(n))$ calls for a fully black-box construction of a **UOWHF** from a **OWF** in the conditions of Corollaries 69 and 70.

Part III

Advice in Communication Complexity

Chapter 7

Equality and Divisibility

Все мы, все мы в этом мире тленны
Тихо льется с кленов листьев медь...
Будь же ты вовек благословенно,
Что пришло процвесть и умереть.[1]

из «Не жалею, не зову, не плачу»
– Сергей Есенин

7.1 Introduction

In this chapter, we study two natural problems in the model of communication complexity with advice (CCwA): the equality of two bitstrings, where each of the parties hold a bitstring and the goal is to decide whether they are equal or not, and the problem of divisibility, where each party has a number, and the goal is to decide whether one of them divides the other.

Recall that in the CCwA model, for any function $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$, the problem for k instances with k bits of advice becomes trivial, i.e., $\mathsf{CC}^{\mathsf{k}}_{\mathsf{k}}(f) = 1$. Pătraşcu conjectured that, for all functions f (as before), whenever the number of advice bits m is significantly smaller than k, i.e., for $m \in o(k)$, the communication complexity of f in the CCwA model is linearly related to that in the classical model, i.e., that for all f, $\mathsf{CC}^{\mathsf{k}}_{\mathsf{m}}(f) \in \Omega(\mathsf{CC}(f))$ for $m \in o(k)$.

However, Chattopadhyay et al. [CEEP12] refuted his conjecture, and showed that in the CCwA model $\log k$ bits of advice and communication are

[1] Somewhat freely translates as "In this world of ours we all are mortal \\Copper leaves from maples gently slide... \\Ever blest was I to be accorded\\Time for blossoming before I died.", from a poem by Sergei Yesenin.

sufficient to compute equality (recall that $\mathsf{CC}(\textsc{Eq}) = n+1$) using the following simple protocol: The advice encodes an index j of a y_j such that $x = y_j$, or 0 for the case that $x \neq y_j$ for all j. After the parties are presented an index i, Bob forwards the advice to Charlie, who then answers with the result of the comarison of y_i and y_j (or with *no* if the advice was 0).

An important problem in the CCwA model is set-disjointness, where the inputs $x, y \subset \{0,1\}^n$ are interpreted as characteristic vectors of a subset of $[n]$. The vectors x and y are disjoint, if and only if the sets they describe are disjoint. That is, $\textsc{Disj} : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ is given by $\textsc{Disj}(x,y) = \bigwedge_{i=1}^{n}((x)_i \neq (y)_i)$. Pătraşcu showed that proving a lower bound on set-disjointness in the CCwA model for some specific parameters would imply a polynomial lower bound on many problems of dynamic data-structures. Chattopadhyay et al. studied set-disjointness in the CCwA model, and showed an upper bound of $\tilde{O}(\sqrt{n})$ on its communication complexity, provided the same amount of advice, and a matching lower bound in a more restricted setting.

7.1.1 Contributions of this Chapter

We first study the power of laconic advice for equality. Somewhat surprisingly, we show that, in the CCwA model, a short communication of only a polylogarithmic number of bits already suffices to deterministically compute equality, provided a laconic advice of just one bit. Our result can be understood as a trade-off between the number of advice bits the protocol utilizes and its communication complexity.

Chattopadhyay et al. observed that for every $k, n \in \mathbb{N}^+$:

$$\mathsf{CC}^{\mathsf{k}}_{\log(\mathsf{k})+1}(\textsc{Eq}) \in O(\log(k)) \ ,$$

where we prove that for every $k, n \in \mathbb{N}^+$:

$$\mathsf{CC}^{\mathsf{k}}_{1}(\textsc{Eq}) \in O(\log(k)(\log(k) + \log(n))) \ .$$

Our second main result in this chapter is a protocol for divisibility. We give a protocol that uses roughly $\tilde{O}(\sqrt{n})$ bits of advice and communication, which improves on the performance of the trivial protocol in the classical model, where it is optimal. To see that $\mathsf{CC}(\textsc{Div}) \in \Omega(n)$, note that any protocol for divisibility could be used to compute equality, since $x = y$ if and only if $x|y$ and $y|x$, and by applying the lower bound on equality. Our

protocol here is inspired by that of Chattopadhyay et al. for set-disjointness. Next, we explain how to reduce set-disjointness to divisibility, and employ their lower-bound on set-disjointness to obtain a matching lower-bound for divisibility in a restricted setting.

7.2 Preliminaries

We denote by $\mathbb{P}$ the set of prime numbers. For a natural number n and a prime number p, we denote by $\nu_p(n)$ the multiplicity of p in n, i.e., the largest exponent i such that $p^i|n$. For a natural number $n = \prod_{i=1}^{k} {p_i}^{\nu_{p_i}(n)}$, where the p_i's are the different prime factors of n, we denote by $\|n\|_\pi \stackrel{\text{def}}{=} \sum_{i=1}^{k} \nu_{p_i}(n)$ the total number of its prime factors including repetitions.

We shall use the following fact: Let $a_1, \ldots, a_\ell, b \in \mathbb{N}$. If $b|a_j$ for all $j \in \{1, \ldots, \ell\}$ then $b|\gcd(a_1, \ldots, a_\ell)$.

7.3 Equality with a Laconic Advice

In this section we continue the study of the Equality problem with advice. We show that, when the advice $A(x, y_1, \ldots, y_k)$ answers the question "Is there an index j such that $x = y_j$?", it is possible to compute equality while exchanging only a polylogarithmic number of bits.

We start with a basic version of our protocol that computes equality using $O(k \log(n))$ bits of communication, which already improves on the trivial protocol for the case $k < n/log(n)$.

7.3.1 A Basic Protocol

After receiving the advice bit, Bob forwards it to Charlie, who maintains a set S of inputs that are potentially equal to x. At the beginning, S is just the entire set of inputs. Then, the protocol proceeds in a step-wise manner, where at each step it asserts for at least one of the y_i's that $x \neq y_i$. Eventually, only one y_i remains, and using the advice, it must hold that $x = y$ and the protocol outputs the index i. More precisely, the protocol proceeds as follows:

1. Bob forwards $A(x, y_1, \ldots, y_k)$ to Charlie. Charlie sets $S \leftarrow \{y_1, \ldots, y_k\}$.

While $|S| > 1$, repeat the following two steps:

2. Charlie chooses two strings y and $y' \neq y$ from S, and sends Bob an index q of a bit on which y and y' differ, i.e., $(y)_q \neq (y')_q$.

3. Bob answers with $(x)_q$, the q'th bit of x, and Charlie updates $S \leftarrow \{z \in S : (z)_q = (x)_q\}$.

4. There is only one element y left in S, and Charlie outputs *yes* if $y = y_i$ and *no* otherwise.

Correctness follows immediately, since we rule out only elements for which we are sure that $y_j \neq x$, and using the advice, we know that, at step 4, it must hold that $x = y$. By the choice of q, the size of S reduces by at least one in every iteration of steps 2 and 3, and so the protocol terminates after at most k iterations of steps 2 and 3, and in every round $\log(n) + 1$ bits are communicated, amounting to a total communication complexity of $O(k \log n)$ bits.

7.3.2 A Protocol Using A Polylogarithmic Number of Communication Bits

As in the basic version of the protocol, the protocol now proceeds iteratively, where at each step the protocol asserts for a *constant fraction* of the elements in S that $x \neq y_j$. It follows that after $O(\log k)$ rounds, only a constant number of possible indices remains.

The idea of our protocol is as follows: Think of the messages of Charlie in the basic version of the protocol as describing a predicate from a set of n predicates of the form $p_i : \{0,1\}^n \to \{0,1\}$, where $p_i(s) = 1$ if and only if the i'th bit of s is 1. In each round Charlie chose an appropriate predicate p_i, and Bob's answer was $p_i(x)$.

This allowed in turn to rule out the equality of at least one of the remaining y_i's to x. In contrast, we show next that for every k, there exists a *single* set of $O(n)$ predicates that allows to rule out the equality of x to a *constant fraction* of the remaining y_i's, for any set of $y_1, \ldots, y_k$. This allows to reduce the number of rounds of the protocol to $O(\log(k))$, while essentially maintaining the same number of bits communicated in every round, since describing a particular predicate would require only $\log(n) + O(1)$ bits. The next lemma

establishes the existence of the aformentioned predicate set. We shall make use of the following:

Let $S \subseteq \{0,1\}^n$. A predicate $p : \{0,1\}^n \to \{0,1\}$ is good for S, if $|\{s \in S \,|\, p(s) = 1\}| > \frac{|S|}{4}$ and $|\{s \in S \,|\, p(s) = 0\}| > \frac{|S|}{4}$.

Lemma 72. *Let $k > 17$ and $n > 0$. Then there exists a set $P = P(n,k)$ of $30n$ predicates such that, for any pair-wise different $y_1, \ldots, y_k \in \{0,1\}^n$ there exists a predicate $p \in P$, which is good for $\{y_1, \ldots, y_k\}$.*

Proof: Consider first a fixed set $Y \stackrel{\text{def}}{=} \{y_1, \ldots, y_k\}$ and a random subset $Z \subseteq \{0,1\}^n$, where each string is chosen independently to Z with probability $1/2$. Observe first, that the expected number of elements in $Z \cap \{y_1, \ldots, y_k\}$ is $k/2$.

For a fixed Y as before, let us define the event (over the random choice of Z) that $Z \cap Y$ differs from its expected value by more than $\frac{k}{4}$:

$$B_{Y,Z} \stackrel{\text{def}}{=} \{|Z \cap Y| > \frac{3k}{4} \text{ or } |Z \cap Y| < \frac{k}{4}\} \ .$$

By the Chernoff bound, we have that

$$\Pr\left[|Z \cap Y| > \frac{3k}{4}\right] < e^{-\frac{k}{12}} \ ,$$

and similarly,

$$\Pr\left[|Z \cap Y| < \frac{k}{4}\right] < e^{-\frac{k}{12}} \ .$$

and thus

$$\Pr\left[B_{Y,Z}\right] < 2 \cdot e^{-\frac{k}{12}} < e^{-\frac{k}{24}} \ ,$$

where the second inequality holds for all $k > 17$.

Consider now ℓ independent copies of Z, that is, the random subsets $Z_1, \ldots, Z_\ell$, and the corresponding events

$$B_{Y,Z_i} \stackrel{\text{def}}{=} \{|Z_i \cap Y| > \frac{3k}{4} \text{ or } |Z_i \cap Y| < \frac{k}{4}\} \ .$$

Set $B_Y \stackrel{\text{def}}{=} \cap_{i=1}^{\ell} B_{Y,Z_i}$. That is, B_Y is the event that for every i, B_{Y,Z_i} happens. Now, by the independence of the B_{Y,Z_i}'s we see that for a fixed set Y, $\Pr[B_Y] < e^{-\frac{k\ell}{24}}$.

Finally, applying the union bound, we obtain that the probability that there exists a set $Y \subset \{0,1\}^n$, where $|Y| = k$ for which B_Y happens is at most $2^{nk} \cdot e^{-\frac{k\ell}{24}}$, as there are at most 2^{nk} subsets of $\{0,1\}^n$ of size k. The choice of $\ell \stackrel{\text{def}}{=} 30n$ ensures that this probability is strictly smaller than 1, which in turn implies the existence of a single choice of $30n$ subsets of $\{0,1\}^n$ which satisfy the condition. By what we have shown, setting the predicate set as the corresponding indicator functions completes the proof. ∎

Lemma 72 guarantees the existence of a good predicate set for every n and $k > 17$. Given such n and k both parties can explicitly agree on a specific one without interaction. For example, we can assume that they can agree on the lexicographically first $30n$ predicates and locally exhaustive search them.

We are now ready to describe the protocol:

1. Bob forwards $A(x, y_1, \ldots, y_k)$ to Charlie. Charlie sets $S \leftarrow \{y_1, \ldots, y_k\}$.

   While $|S| > 17$, repeat the following two steps:

2. Charlie sets $k' = |S|$, computes locally a set $P(n, k')$, chooses a good predicate for S, and sends its index q to Bob along with k'.

3. Bob computes locally $P(n, k')$, and answers with $p_q(x)$. Charlie updates $S \leftarrow \{w \in S : p_q(w) = p_q(x)\}$.

4. Continue with the loop of the basic protocol.

Given the lemma, the correctness of the protocol follows readily. As for the communication complexity, note that in every round of iterating steps 2 and 3, at most $(\log(k) + \log(n) + 1)$ bits are communicated, describing $|S|$ and the index of the predicate. Since the chosen predicate is good, it follows that the updated set in Step 4 has size of at most $\frac{3}{4}|S|$, and therefore after $O(\log(k))$ rounds we continue with at most 17 rounds of the basic protocol, resulting in a total communication complexity of $O(\log(k)(\log(k) + \log(n)))$.

Thus we have shown:

Theorem 73. *For all $k \in \mathbb{N}^+$,*

$$\mathsf{CC}_1^{\mathrm{k}}(\mathrm{EQ}) \in O(\log(k)(\log(k) + \log(n))) \ .$$

7.4 Divisibility

Here, Bob and Charlie have inputs x and $y_1, \ldots, y_k \in \{0,1\}^n$, respectively, interpreted as natural numbers in the inteval $[1, 2^n]$, identifying 2^n with the bitstring 0^n. After receiving an advice $A(x, y_1, \ldots, y_k)$ they are presented an index i and required to compute $x|y_i$.

We show a protocol that communicates $O(\log(n)(\log(k) + \log(n))\sqrt{n})$ bits, provided an advice of similar size. First we shall describe our protocol using the following simplifying assumption, and later we explain how to remove it. We assume that the prime factors of all $x, y_1, \ldots, y_k$ are either within the interval $[2^{2^\tau}, 2^{2^{\tau+1}})$ for some $\tau \in \{0, \ldots, \log(n)/2 - 1\}$ or in the interval $[2^{2^\tau}, 2^n)$ for $\tau = \log(n)/2$. Observe that this implies that each of the numbers $x, y_1, \ldots, y_k$ has at most $n/2^\tau$ distinct factors. Moreover, if $\tau \in \{0, \ldots, \log(n)/2 - 1\}$, each factor can be described using at most $2^{\tau+1}$ bits.

The main idea in our algorithm is to reveal information about x using $y_1, \ldots, y_k$. The advice consists in a subset of the inputs of Charlie for which $x|y_i$, and it is built by iterating over the inputs. The inputs that are added are those which x divides, and that additionally, given all the indices added so far, contribute enough additional information about x. Roughly speaking, the index of an input y_i is not added to the advice string for one of two reasons: Either x does not divide it, or, it would not have contributed enough information about x, given the previous positive inputs. If later an input not added due to the latter condition is presented, it could be computed using only $\tilde{O}(\sqrt{n})$ bits. In what follows we set

$$t \stackrel{\text{def}}{=} \sqrt{n}/2^\tau \ .$$

More precisely, the advice sent by Alice is a subset S of the indices $\{1, \ldots, k\}$, which is constructed as follows:

Start with $S = \emptyset$ and add to S the minimum index j_0 such that $x|y_{j_0}$. Then, loop on the elements $y_{j_0+1}, \ldots, y_k$: Let $j_0, \ldots, j_\ell$ be the current indices of S. Add the index j of the current element y_j to S if:

- $x|y_j$ and
- $\|\gcd(y_{j_0}, \ldots, y_{j_\ell})\|_\pi - \|\gcd(y_{j_0}, \ldots, y_{j_\ell}, y_j)\|_\pi \geq t$.

The second condition ensures that the gcd of the elements currently indexed by S contains at least t more prime factors including repetition than the gcd of those elements along with the number y_j.

We claim that during this process, at most $\frac{n}{2^\tau t}$ elements are added to S. Indeed, by our simplifying assumption, y_{j_0} contains at most $\frac{n}{2^\tau}$ different factors and the second condition asserts that the number of prime factors remaining in $\gcd_{j' \in S} a'_j$ after adding an element decreases by at least t.

Substituting for t, we have that $|S| < \sqrt{n}$, and therefore the advice sent by Alice (a description of S) can be encoded using at most $\sqrt{n}\log(k)$ bits.

After Bob receives S from Alice, the parties are presented an index i and need to determine whether $x|y_i$. The protocol between Bob and Charlie continues as follows:

1. If $i \in S$, Bob outputs *yes*.
2. Otherwise, Bob forwards S to Charlie.
3. Charlie computes the set $S' = \{j' \in S \mid j' < i\}$, i.e., the constructed set S as it was just before index i was processed.
4. If $\|\gcd_{j' \in S'}(y_{j'})\|_\pi - \|\gcd(\gcd_{j' \in S'} y_{j'}), y_i)\|_\pi \geq t$, he outputs *no*. Otherwise, there are at most $t-1$ distinct prime factors appearing in y_i but not in $\gcd_{j' \in S'}(y_{j'})$. For each such factor p, Charlie sends $(p, \nu_p(y_i))$ to Bob.
5. Bob outputs *yes* if, for every received pair $(p, \nu_p(y_i))$, it holds that $\nu_p(x) \leq \nu_p(y_i)$ and otherwise outputs *no*.

Let us now see that the protocol always outputs a correct answer. If it outputs *yes* at step 1, by construction, for all indices in S, it holds that $x|y_j$. If the protocol outputs *no* at step 4, it cannot be the case that $x|y_i$, as otherwise both conditions during the construction of S had been satisfied and i would have been added to S. Lastly, if *no* is output at step 5, it is because the protocol witnesses a factor with higher multiplicity in x than in y_i. When this is not the case, we claim that $x|y_i$. We show that for each prime factor in x, it holds that $\nu_p(x) \leq \nu_p(y_i)$. Let us distinguish two cases: If p is one of the up to $t-1$ factors not appearing in $\gcd(\gcd_{j' \in S}(y_{j'}), y_i)$, then the inequality is asserted by Bob. For any other such factor p, we have (1) $\nu_p(\gcd_{j' \in S'}(y_{j'})) \leq \nu_p(y_i)$, and additionally, by construction of S it holds that $x|y_{j'}$ for all $j' \in S$ and therefore $x|\gcd_{j' \in S}(y_{j'})$, and, in particular, (2) $\nu_p(x) \leq \nu_p(\gcd_{j' \in S}(y_{j'}))$. The correctness in this case follows from (1) and (2).

Next, we analyze the communication complexity of our protocol. At step 1, S is forwarded and at most $\sqrt{n}\log(k)$ bits are sent, and at step 4 at most $t-1$ pairs are sent. This implies that, for the case that $\tau = \log(n)/2$, it holds that $t = 1$ and therefore no pairs are sent. For $\tau \in \{0, \ldots, \log(n)/2 - 1\}$, using the second part of the assumption, each prime can be described using at most $2^{\tau+1}$ bits. The multiplicity of every prime number is at most n, which can be described using $\log(n)$ bits, and therefore at most $t(2^{\tau+1} + \log(n)) \in O(\sqrt{n}\log(n))$ bits are communicated at this step. Thus, the total number of bits communicated is $O(\sqrt{n}(\log(k) + \log(n)))$.

Finally, in order to get rid of the assumption, note that $x|y_i$ if and only if, $x^{(\tau)}|y_i^{(\tau)}$ for all $\tau \in \{0, \ldots, \log(n)/2\}$, where for $m \in \mathbb{N}$ we define $m^{(\tau)} \stackrel{\text{def}}{=} \prod_{p \in [2^{2^\tau}, 2^{2^{\tau+1}}) \cap \mathbb{P}} p^{\nu_p(m)}$ for $\tau \in \{0, \ldots, \log(n)/2 - 1\}$ and $m^{(\tau)} \stackrel{\text{def}}{=} \prod_{p \in [2^{2^\tau}, 2^n) \cap \mathbb{P}} p^{\nu_p(m)}$ for $\tau = \log(n)/2$.

Thus, for the final protocol we run the protocol in parallel for each of the $\log(n)/2 + 1$ possible values of τ, and output *yes* in case all runs output *yes*, and *no* otherwise. This results in an overhead of an $O(\log(n))$ factor to the original protocol. We summarize: Utilizing $O(\sqrt{n}\log(k)\log(n))$ advice bits our protocol communicates $O(\sqrt{n}\log(n)(\log(k) + \log(n)))$ bits. Thus we have shown:

Theorem 74. *For all $k \in \mathbb{N}^+$,*

$$\mathsf{CC}^{\mathsf{k}}_{\log(\mathsf{n})\log(\mathsf{k})\sqrt{\mathsf{n}}}(\textsc{Div}) \in O(\sqrt{n}\log(n)(\log(k) + \log(n))) \ .$$

7.4.1 On the Asymmetry of Divisibility

We note that the communication complexity with advice model is inherently asymmetric, and therefore a protocol for $\textsc{Div}(x, y) = x|y$ does not yield a protocol for $\textsc{Div}'(x, y) = y|x$, as is the case in the classical model of communication complexity. In this section, we explain the changes needed in our protocol to obtain a protocol for $\textsc{Div}'$. In our protocol from the previous section, we used the advice to reveal information about x using the y_i's. In particular, the number encoded by the gcd of the y_i's chosen to the advice could be understood as a relatively tight upper bound on the prime powers of x. The analogous advice information for $y|x$ consists in a lower bound on the prime powers of x. Analogously to before, observe that if $a|x$ and $b|x$ then also $\text{lcm}(a, b)|x$. The advice is generated similarly to before, where the set S first contains the minimal index j_0 such that $y_{j_0}|x$. Then,

looping over the indices $j_0 + 1, \ldots, k$, index j is added if and only if $y_j | x$ and $\|\operatorname{lcm}(y_{j_0}, \ldots, y_{j_\ell}, y_j)\|_\pi - \|\operatorname{lcm}(y_{j_0}, \ldots, y_{j_\ell})\|_\pi \geq t$, where $j_0, \ldots, j_\ell$ are the current elements of S. Steps (4) and (5) of the protocol now become:

4. If $\|\operatorname{lcm}(y_{j_0}, \ldots, y_{j_\ell}, y_j)\|_\pi - \|\operatorname{lcm}(y_{j_0}, \ldots, y_{j_\ell})\|_\pi \geq t$ it outputs *no*. Otherwise, there are at most $t - 1$ distinct prime factors appearing in $\operatorname{lcm}_{j' \in S'}(y_{j'})$ but not in y_i. For each such factor p, Charlie sends $(p, \nu_p(y_i))$ to Bob.
5. Bob outputs *yes* if, for every received pair $(p, \nu_p(y_i))$, it holds that $\nu_p(x) \geq \nu_p(y_i)$, and otherwise outputs *no*.

The correctness and analysis of the protocol follow analogously to before.

7.4.2 An Almost Matching Lower Bound in Restricted Settings

The problem of set-disjointness consists in two n-bit inputs x and y, where each is interpreted as the characteristic vector of a subset of a set of n elements. Inputs $x, y \in \{0,1\}^n$ are disjoint if $(y)_i = 0$ whenever $(x)_i = 1$. Chattopadhyay et al. [CEEP12] studied the problem of set-disjointness in the CCwA model, and showed that for $k \geq \sqrt{n}$, any protocol with advice of size $m \leq \alpha\sqrt{n}$ communicates at least $\beta\sqrt{n}$ bits for some constants $0 < \alpha, \beta < 1$. In what follows, we describe a reduction from set-disjointness to divisibility, establishing an analogous lower bound for divisibility.

Given inputs a and b (characteristic vectors of some sets A and B) to set-disjointness, we first observe that A and B are disjoint if and only if $A \subseteq \overline{B}$. Now, let $p_1, \ldots, p_n$ be the first n prime numbers, and set $N_A \stackrel{\text{def}}{=} \prod_{j=1}^{n} {p_i}^{a_i}$ and $N_{\overline{B}} \stackrel{\text{def}}{=} \prod_{j=1}^{n} {p_i}^{1-b_i}$. It follows that $A \cap B = \emptyset$ if and only if $N_A \mid N_{\overline{B}}$. By the prime number theorem, it holds that for all large enough n, the first n prime numbers lie in the interval $[1, 3n\log(n)]$, and therefore both N_A and $N_{\overline{B}}$ are described using at most $n \cdot \log(3n\log(n)) < 2n\log(n)$ bits. Therefore, any protocol in the CCwA model for k inputs of size $2n\log(n)$ and m bits of advice yields a protocol (with the same k and m values) for inputs of size n for divisibility; the parties compute N_x and $N_{\overline{y_1}}, \ldots, N_{\overline{y_k}}$ and run the protocol for divisibility on these inputs. Setting $f(n) = 2n\log(n)$, the lower bound of Chattopadhyay et al. (Theorem 5.2 in [CEEP12]) establishes that a protocol for $k \geq \sqrt{f^{-1}(n)}$ inputs of size n with advice of size at most $\alpha\sqrt{f^{-1}(n)}$

communicates at least $\beta\sqrt{f^{-1}(n)}$ bits. In view of our protocol from Section 7.4, it follows that this is best possible (up to a logarithmic factor) with advice of size $\tilde{O}(\sqrt{n})$.

Bibliography

[AA92] Noga Alon and Yossi Azar. On-line steiner trees in the euclidean plane. In *Symposium on Computational Geometry*, pages 337–343, 1992.

[AAB04] Baruch Awerbuch, Yossi Azar, and Yair Bartal. On-line generalized steiner problem. *Theor. Comput. Sci.*, 324(2-3):313–324, 2004.

[AGV12] Scott Ames, Rosario Gennaro, and Muthuramakrishnan Venkitasubramaniam. The generalized randomized iterate and its application to new efficient constructions of uowhfs from regular one-way functions. In Xiaoyun Wang and Kazue Sako, editors, *ASIACRYPT*, volume 7658 of *Lecture Notes in Computer Science*, pages 154–171. Springer, 2012.

[Bar01] Boaz Barak. How to go beyond the black-box simulation barrier. In *FOCS*, pages 106–115. IEEE Computer Society, 2001.

[Bar14] Kfir Barhum. Tight bounds for the advice complexity of the online minimum steiner tree problem. In Geffert et al. [GPR+14], pages 77–88.

[BBF+14] Kfir Barhum, Hans-Joachim Böckenhauer, Michal Forisek, Heidi Gebauer, Juraj Hromkovič, Sacha Krug, Jasmin Smula, and Björn Steffen. On the power of advice and randomization for the disjoint path allocation problem. In Geffert et al. [GPR+14], pages 89–101.

[BBH+13] Maria Paola Bianchi, Hans-Joachim Böckenhauer, Juraj Hromkovič, Sacha Krug, and Björn Steffen. On the advice complexity of the online $L(2, 1)$-coloring problem on paths and cycles. In Ding-Zhu Du and Guochuan Zhang, editors, *Computing and Combinatorics, 19th International Conference, COCOON 2013, Hangzhou, China, June 21-23, 2013. Proceedings*, volume 7936 of *Lecture Notes in Computer Science*, pages 53–64. Springer, 2013.

[BBHK12] Maria Paola Bianchi, Hans-Joachim Böckenhauer, Juraj Hromkovič, and Lucia Keller. Online coloring of bipartite graphs with and without advice. In Joachim Gudmundsson, Julián Mestre, and Taso Viglas, editors, *Computing and Combinatorics - 18th Annual International Conference, COCOON 2012, Sydney, Australia, August 20-22, 2012. Proceedings*, volume 7434 of *Lecture Notes in Computer Science*, pages 519–530. Springer, 2012.

[BC97] Piotr Berman and Chris Coulston. On-line algorithms for steiner tree problems (extended abstract). In Frank Thomson Leighton and Peter W. Shor, editors, *STOC*, pages 344–353. ACM, 1997.

[BEY98] Allan Borodin and Ran El-Yaniv. *Online computation and competitive analysis.* Cambridge University Press, 1998.

[BH12] Kfir Barhum and Thomas Holenstein. A cookbook for black-box separations and a recipe for uowhfs. *Electronic Colloquium on Computational Complexity (ECCC)*, 19:173, 2012.

[BH13] Kfir Barhum and Thomas Holenstein. A cookbook for black-box separations and a recipe for uowhfs. In *TCC*, pages 662–679, 2013.

[BHK+14] Hans-Joachim Böckenhauer, Juraj Hromkovič, Dennis Komm, Sacha Krug, Jasmin Smula, and Andreas Sprock. The string guessing problem as a method to prove lower bounds on the advice complexity. In *Theor. Comput. Sci.*, pages 95–108, 2014.

[BKK+09] Hans-Joachim Böckenhauer, Dennis Komm, Rastislav Královič, Richard Královič, and Tobias Mömke. On the advice complexity of online problems. In Yingfei Dong, Ding-Zhu Du, and Oscar H. Ibarra, editors, *ISAAC*, volume 5878 of *Lecture Notes in Computer Science*, pages 331–340. Springer, 2009.

[BKKK11] Hans-Joachim Böckenhauer, Dennis Komm, Rastislav Královic, and Richard Královic. On the advice complexity of the k-server problem. In *ICALP (1)*, pages 207–218, 2011.

[BKKR12] Hans-Joachim Böckenhauer, Dennis Komm, Richard Královič, and Peter Rossmanith. On the advice complexity of the knapsack problem. In David Fernández-Baca, editor, *LATIN*, volume 7256 of *Lecture Notes in Computer Science*, pages 61–72. Springer, 2012.

[BM12] Kfir Barhum and Ueli Maurer. UOWHFs from OWFs: Trading regularity for efficiency. In Alejandro Hevia and Gregory Neven, editors, *LATINCRYPT*, volume 7533 of *Lecture Notes in Computer Science*, pages 234–253. Springer, 2012.

[CEEP12] Arkadev Chattopadhyay, Jeff Edmonds, Faith Ellen, and Toniann Pitassi. A little advice can be very helpful. In Yuval Rabani, editor, *SODA*, pages 615–625. SIAM, 2012.

[DHZ12] Reza Dorrigiv, Meng He, and Norbert Zeh. On the advice complexity of buffer management. In *ISAAC*, pages 136–145, 2012.

[DKK12] Stefan Dobrev, Rastislav Královic, and Richard Královic. Independent set with advice: The impact of graph knowledge - (extended abstract). In Thomas Erlebach and Giuseppe Persiano, editors, *Approximation and Online Algorithms - 10th International Workshop, WAOA 2012, Ljubljana, Slovenia, September 13-14, 2012, Revised Selected Papers*, volume 7846 of *Lecture Notes in Computer Science*, pages 2–15. Springer, 2012.

[DKM12] Stefan Dobrev, Rastislav Královic, and Euripides Markou. Online graph exploration with advice. In Guy Even and Magnús M. Halldórsson, editors, *Structural Information and Communication Complexity - 19th International Colloquium, SIROCCO 2012, Reykjavik, Iceland, June 30-July 2, 2012, Revised Selected Papers*, volume 7355 of *Lecture Notes in Computer Science*, pages 267–278. Springer, 2012.

[DKP08] Stefan Dobrev, Rastislav Královič, and Dana Pardubská. How much information about the future is needed? In Viliam Geffert, Juhani Karhumäki, Alberto Bertoni, Bart Preneel, Pavol Návrat, and Mária Bieliková, editors, *SOFSEM*, volume 4910 of *Lecture Notes in Computer Science*, pages 247–258. Springer, 2008.

[EFKR11] Yuval Emek, Pierre Fraigniaud, Amos Korman, and Adi Rosén. Online computation with advice. *Theor. Comput. Sci.*, 412(24):2642–2656, 2011.

[FKS12] Michal Forisek, Lucia Keller, and Monika Steinová. Advice complexity of online coloring for paths. In *LATA*, pages 228–239, 2012.

[GGKT05] Rosario Gennaro, Yael Gertner, Jonathan Katz, and Luca Trevisan. Bounds on the efficiency of generic cryptographic constructions. *SIAM J. Comput.*, 35(1):217–246, 2005.

[GGLS08] Naveen Garg, Anupam Gupta, Stefano Leonardi, and Piotr Sankowski. Stochastic analyses for online combinatorial optimization problems. In Shang-Hua Teng, editor, *SODA*, pages 942–951. SIAM, 2008.

[Gol01] Oded Goldreich. *The Foundations of Cryptography - Volume 1, Basic Techniques*. Cambridge University Press, 2001.

[Gol08] Oded Goldreich. *Computational complexity - a conceptual perspective*. Cambridge University Press, 2008.

[GPR+14] Viliam Geffert, Bart Preneel, Branislav Rovan, Julius Stuller, and A Min Tjoa, editors. *SOFSEM 2014: Theory and Practice of Computer Science - 40th International Conference on Current Trends in Theory and Practice of Computer Science, Nový Smokovec, Slovakia, January 26-29, 2014, Proceedings*, volume 8327 of *Lecture Notes in Computer Science*. Springer, 2014.

[HH09] Iftach Haitner and Thomas Holenstein. On the (im)possibility of key dependent encryption. In Omer Reingold, editor, *TCC*, volume 5444 of *Lecture Notes in Computer Science*, pages 202–219. Springer, 2009.

[HHR+10] Iftach Haitner, Thomas Holenstein, Omer Reingold, Salil P. Vadhan, and Hoeteck Wee. Universal one-way hash functions via inaccessible entropy. In Henri Gilbert, editor, *EUROCRYPT*, volume 6110 of *Lecture Notes in Computer Science*, pages 616–637. Springer, 2010.

[HHRS07] Iftach Haitner, Jonathan J. Hoch, Omer Reingold, and Gil Segev. Finding collisions in interactive protocols - a tight lower bound on the round complexity of statistically-hiding commitments. In *FOCS*, pages 669–679. IEEE Computer Society, 2007.

[HKK10] Juraj Hromkovič, Rastislav Královič, and Richard Královič. Information complexity of online problems. In Petr Hlinený and Antonín Kucera, editors, *MFCS*, volume 6281 of *Lecture Notes in Computer Science*, pages 24–36. Springer, 2010.

[HNO+09] Iftach Haitner, Minh-Huyen Nguyen, Shien Jin Ong, Omer Reingold, and Salil P. Vadhan. Statistically hiding commitments and statistical zero-knowledge arguments from any one-way function. *SIAM J. Comput.*, 39(3):1153–1218, 2009.

[Hoe63] Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, 1963.

[HR04] Chun-Yuan Hsiao and Leonid Reyzin. Finding collisions on a public road, or do secure hash functions need secret coins? In Matthew K. Franklin, editor, *CRYPTO*, volume 3152 of *Lecture Notes in Computer Science*, pages 92–105. Springer, 2004.

[Hro97] Juraj Hromkovič. *Communication Complexity and Parallel Computing*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1997.

[HRV10] Iftach Haitner, Omer Reingold, and Salil P. Vadhan. Efficiency improvements in constructing pseudorandom generators from one-way functions. In Schulman [Sch10], pages 437–446.

[HRVW09] Iftach Haitner, Omer Reingold, Salil P. Vadhan, and Hoeteck Wee. Inaccessible entropy. In Michael Mitzenmacher, editor, *STOC*, pages 611–620. ACM, 2009.

[HS12] Thomas Holenstein and Makrand Sinha. Constructing a pseudorandom generator requires an almost linear number of calls. *CoRR*, abs/1205.4576. Extended Abstract to appear in FOCS 2012, 2012.

[IR89] Russell Impagliazzo and Steven Rudich. Limits on the provable consequences of one-way permutations. In Johnson [Joh89], pages 44–61.

[IW91] Makoto Imase and Bernard M. Waxman. Dynamic steiner tree problem. *SIAM J. Discrete Math.*, 4(3):369–384, 1991.

[Joh89] David S. Johnson, editor. *Proceedings of the 21st Annual ACM Symposium on Theory of Computing, May 14-17, 1989, Seattle, Washigton, USA*. ACM, 1989.

[KK11] Dennis Komm and Richard Královic. Advice complexity and barely random algorithms. *RAIRO - Theor. Inf. and Applic.*, 45(2):249–267, 2011.

[KKM12] Dennis Komm, Richard Královic, and Tobias Mömke. On the advice complexity of the set cover problem. In *CSR*, pages 241–252, 2012.

[KN97] Eyal Kushilevitz and Noam Nisan. *Communication Complexity*. Cambridge University Press, New York, NY, USA, 1997.

[KST99] Jeong Han Kim, Daniel R. Simon, and Prasad Tetali. Limits on the efficiency of one-way permutation-based hash functions. In *FOCS*, pages 535–542, 1999.

[MM11] Takahiro Matsuda and Kanta Matsuura. On black-box separations among injective one-way functions. In Yuval Ishai, editor, *TCC*, volume 6597 of *Lecture Notes in Computer Science*, pages 597–614. Springer, 2011.

[NY89] Moni Naor and Moti Yung. Universal one-way hash functions and their cryptographic applications. In Johnson [Joh89], pages 33–43.

[Pat10] Mihai Patrascu. Towards polynomial lower bounds for dynamic problems. In Schulman [Sch10], pages 603–610.

[Rom90] John Rompel. One-way functions are necessary and sufficient for secure signatures. In *STOC*, pages 387–394. ACM, 1990.

[Ros10] S.M. Ross. *A First Course in Probability*. Pearson Prentice Hall, 2010.

[RS10] Alon Rosen and Gil Segev. Chosen-ciphertext security via correlated products. *SIAM J. Comput.*, 39(7):3058–3088, 2010.

[RTV04] Omer Reingold, Luca Trevisan, and Salil P. Vadhan. Notions of reducibility between cryptographic primitives. In Moni Naor, editor, *TCC*, volume 2951 of *Lecture Notes in Computer Science*, pages 1–20. Springer, 2004.

[Sch10] Leonard J. Schulman, editor. *Proceedings of the 42nd ACM Symposium on Theory of Computing, STOC 2010, Cambridge, Massachusetts, USA, 5-8 June 2010*. ACM, 2010.

[Sho00] Victor Shoup. A composition theorem for universal one-way hash functions. In Bart Preneel, editor, *EUROCRYPT*, volume 1807 of *Lecture Notes in Computer Science*, pages 445–452. Springer, 2000.

[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.

[SSU13] Sebastian Seibert, Andreas Sprock, and Walter Unger. Advice complexity of the online coloring problem. In *CIAC*, pages 345–357, 2013.

[ST85] Daniel Dominic Sleator and Robert Endre Tarjan. Amortized efficiency of list update and paging rules. *Commun. ACM*, 28(2):202–208, 1985.

[SY90] Alfredo De Santis and Moti Yung. On the design of provably secure cryptographic hash functions. In *EUROCRYPT*, pages 412–431, 1990.

[Yao82] Andrew Chi-Chih Yao. Theory and applications of trapdoor functions (extended abstract). In *FOCS*, pages 80–91. IEEE Computer Society, 1982.

[YGLW14] Yu Yu, Dawu Gu, Xiangxue Li, and Jian Weng. (almost) optimal constructions of uowhfs from 1-to-1, regular one-way functions and beyond. Cryptology ePrint Archive, Report 2014/393, 2014. `http://eprint.iacr.org/`.