On Basing One-Way Functions on NP-Hardness

Adi Akavia[*] Oded Goldreich[†] Shafi Goldwasser[‡] Dana Moshkovitz[§]

February 28, 2006

Abstract

We consider the possibility of basing one-way functions on NP-Hardness; that is, we study possible reductions from a worst-case decision problem to the task of average-case inverting a polynomial-time computable function f (*i.e.*, reductions that are supposed to establish that the latter function is one-way based on a worst-case assumption regarding the decision problem). Our main findings are the following two negative results:

1. If given y one can efficiently compute $|f^{-1}(y)|$ then the existence of a (randomized) reduction of $\mathcal{NP}$ to the task of inverting f implies that co$\mathcal{NP} \subseteq \mathcal{AM}$. Thus, it follows that such reductions cannot exist (unless co$\mathcal{NP} \subseteq \mathcal{AM}$).

   The result extends to functions for which the preimage size is efficiently verifiable via an AM protocol. For example, this includes regular functions with efficiently recognizable range (possibly via an AM protocol).

   We stress that this result holds for any reduction, including *adaptive* ones. We note that the previously known negative results regarding worst-case to average-case reductions were essentially confined to *non-adaptive* reductions, whereas known positive results (regarding computational problems in the geometry of numbers) use adaptive reductions.

2. For any function f, the existence of a (randomized) *non-adaptive* reduction of $\mathcal{NP}$ to the task of average-case inverting f implies that co$\mathcal{NP} \subseteq \mathcal{AM}$.

   This result improves over the previous negative results (for this case) that placed co$\mathcal{NP}$ in non-uniform $\mathcal{AM}$.

Our work builds upon and improves on the previous works of Feigenbaum and Fortnow (*SIAM Journal on Computing*, 1993) and Bogdanov and Trevisan (*44th FOCS*, 2003), while capitalizing on the additional "computational structure" of the search problem associated with the task of inverting polynomial-time computable functions. We believe that our results illustrate the gain of directly studying the context of one-way functions rather than inferring results for it from a the general study of worst-case to average-case reductions.

Area: Cryptography and Complexity.

Keywords: One-Way Functions, Average-Case complexity, Reductions, Adaptive versus Non-adaptive machines, Interactive Proof Systems.

[*]akavia@mit.edu

[†]oded.goldreich@weizmann.ac.il

[‡]shafi@theory.csail.mit.edu

[§]dana.moshkovitz@weizmann.ac.il

Contents

1 Introduction

One-way functions are functions that are easy to compute but hard to invert, where the hardness condition refers to the average-case complexity of the inverting task. The existence of one-way functions is the cornerstone of modern cryptography: almost all cryptographic primitives imply the existence of one-way functions, and most of them can be constructed based either on the existence of one-way functions or on related (but seemingly stronger) versions of this assumption.

As noted above, the hardness condition of one-way functions is an average-case complexity condition. Clearly, this average-case hardness condition implies a worst-case hardness condition; that is, the existence of one-way functions implies that $\mathcal{NP}$ is not contained in $\mathcal{BPP}$. A puzzling question of fundamental nature is whether or not the necessary worst-case condition is a sufficient one; that is, can one base the existence of one-way functions on the assumption that $\mathcal{NP}$ is not contained in $\mathcal{BPP}$.

More than two decades ago, Brassard [Br] observed that the inverting task associated with a one-way *permutation* cannot be $\mathcal{NP}$-hard, unless $\mathcal{NP} = \text{co}\mathcal{NP}$. The question was further addressed in the works of Feigenbaum and Fortnow [FeFo] and Bogdanov and Trevisan [BoTr], which focused on the study of worst-case to average-case *reductions among decision problems*.

1.1 Our Main Results

In this paper we re-visit the aforementioned question, but do so explicitly. We study possible reductions from a worst-case decision problem to the task of average-case inverting a polynomial-time computable function (*i.e.*, reductions that are supposed to establish that the latter function is one-way based on a worst-case assumption regarding the decision problem). Specifically, we consider (randomized) reductions of $\mathcal{NP}$ to the task of average-case inverting a polynomial-time computable function f, and capitalize on the additional "computational structure" of the search problem associated with inverting f. This allows us to strengthen previously known negative results, and obtain the following two main results:

1. If given y one can efficiently compute $|f^{-1}(y)|$ then the existence of a (randomized) reduction of $\mathcal{NP}$ to the task of inverting f implies that $\text{co}\mathcal{NP} \subseteq \mathcal{AM}$.

   The result extends to functions for which the preimage size is efficiently verifiable via an AM protocol. For example, this includes regular functions (cf., *e.g.*, [GKL]) with efficiently recognizable range.

   Recall that $\mathcal{AM}$ is the class of sets having two-round interactive proof systems, and that it is widely believed that $\text{co}\mathcal{NP}$ is not contained in $\mathcal{AM}$. Thus, it follows that such reductions cannot exist (unless $\text{co}\mathcal{NP} \subseteq \mathcal{AM}$).

   We stress that this result holds for any reduction, including *adaptive* ones. We note that the previously known negative results regarding worst-case to average-case reductions were essentially confined to *non-adaptive* reductions (cf. [FeFo, BoTr], where [FeFo] also handles restricted levels of adaptivity). Furthermore, the result holds also for reductions to worst-case inverting f, thus establishing a separation between this restricted type of one-way functions and the general ones (see Remark 7).

2. For any (polynomial-time computable) function f, the existence of a (randomized) *non-adaptive* reduction of $\mathcal{NP}$ to the task of average-case inverting f implies that $\text{co}\mathcal{NP} \subseteq \mathcal{AM}$.

   This result improves over the previous negative results of [FeFo, BoTr] that placed $\text{co}\mathcal{NP}$ in non-uniform $\mathcal{AM}$ (instead of in *uniform* $\mathcal{AM}$).

These negative results can be interpreted in several ways: see discussion in Section 6.

1.2 Relation to Feigenbaum-Fortnow and Bogdanov-Trevisan

Our work is inspired by two previous works. The first work, by Feigenbaum and Fortnow [FeFo], posed the question of whether or not $\mathcal{NP}$-complete problems can be *random self-reducible*. That is, can (worst case) instances of $\mathcal{NP}$-complete problems be reduced to one or more *random instances, where the latter instances are drawn according to a predetermined distribution*. The main result of [FeFo] is that if such (*non-adaptive*) reductions exist, then co$\mathcal{NP}$ is in a non-uniform version of $\mathcal{AM}$, denoted $\mathcal{AM}_{\rm poly}$. Non-uniformity was used in their work to encode statistics about the target distribution of the reduction.

Bogdanov and Trevisan [BoTr] start by viewing the result of [FeFo] as a result about the impossibility of worst-case to average-case reductions for $\mathcal{NP}$-complete problems. They note that even if one cares about the average-case complexity of a problem with respect to a specific distribution (*e.g.*, the uniform one) then it needs not be the case that a worst-case to average-case reduction must make queries according to this distribution. Furthermore, the distribution of queries may depend on the input to the reduction, and so statistics regarding it cannot be given as advice. Nevertheless, combining the ideas of [FeFo] with additional ideas (some borrowed from the study of locally-decodable codes [KaTr]), Bogdanov and Trevisan showed that any *non-adaptive* reduction of (worst-case) $\mathcal{NP}$ to the average-case complexity of $\mathcal{NP}$ (with respect to any sampleable distribution) implies that co$\mathcal{NP} \subseteq \mathcal{AM}_{\rm poly}$.

Although a main motivation of [BoTr] is the question of basing one-way functions on worst-case $\mathcal{NP}$-hardness, its focus (like that of [FeFo]) is on the more general study of *decision problems*. Using known reductions between search and decision problems in the context of distributional problems [BCGL, ImLe], Bogdanov and Trevisan [BoTr] also derive implications on the (im)possibility of basing one-way functions on $\mathcal{NP}$-hardness. In particular, they conclude that if there exists an $\mathcal{NP}$-complete set for which deciding any instance is *non-adaptively* reducible to *inverting a one-way function* (or, more generally, to a search problem with respect to a sampleable distribution), then co$\mathcal{NP} \subseteq \mathcal{AM}_{\rm poly}$.

We emphasize that the *techniques* of [BoTr] refer explicitly only to decision problems, and do not relate to the underlying search problems (*e.g.*, inverting a supposedly one-way function). In doing so, they potentially lose twice: they lose the extra structure of search problems and they lose the additional structure of the task of inverting polynomial-time computable functions. To illustrate the latter aspect, we re-formulate the problem of inverting a *polynomial-time computable function* as follows (or rather spell out what it means in terms of search problems). The problem of (average-case) inverting f on the distribution $f(U_n)$, where U_n denotes the uniform distribution over $\{0,1\}^n$, has the following features:

1. We care about the average-case complexity of the problem; that is, the probability that an efficient algorithm given a random (efficiently sampled) instance y (*i.e.*, $y \leftarrow f(U_n)$) finds $x \in f^{-1}(y)$.

2. The problem is in NP; that is, the solution is relatively short and given an instance of the problem (*i.e.*, y) and a (candidate) solution (*i.e.*, x), it is easy to verify that the solution is correct (*i.e.*, $y = f(x)$).

3. There exists an efficient algorithm that generates random instance-solution pairs (*i.e.*, pairs (y, x) such that $y = f(x)$, for uniformly distributed $x \in \{0,1\}^n$).

Indeed, the first two items are common to all average-case NP-search problems (with respect to sampleable distributions), but the third item is specific to the context of one-way functions (cf. [Go, Sec. 2.1]). In contrast, a generic sampleable distribution of instances is not necessarily coupled with a corresponding sampleable distribution of random instance-solution pairs. Indeed, capitalizing on the third item is the source of our success to obtain stronger (negative) results regarding the possibility of basing one-way functions on $\mathcal{NP}$-hardness.

The results of [BoTr, FeFo] are limited in two ways. First, they only consider *non-adaptive* reductions, whereas the celebrated worst-case to average-case reductions of lattice problems (cf. [Aj, MiRe]) are adaptive. Furthermore, these positive results seem to illustrate the power of adaptive versus non-adaptive reductions.[1] Second, [BoTr, FeFo] reach conclusions involving a *non-uniform* complexity class (*i.e.*, $\mathcal{AM}_{\rm poly}$). Non-uniformity seems an artifact of their techniques, and one may hope to conclude that co$\mathcal{NP} \subseteq \mathcal{AM}$ rather than co$\mathcal{NP} \subseteq \mathcal{AM}_{\rm poly}$. (One consequence of the uniform conclusion is that it implies that the polynomial-time hierarchy collapses to the second level, whereas the non-uniform conclusion only implies a collapse to the third level.) As stated before, working directly with one-way functions allows us to remove the first shortcoming in some cases and remove the second shortcoming in all cases.

1.3 The Benefits of Direct Study of One-Way Functions

The results presented in this paper indicate the gains of studying the question of basing one-way functions on $\mathcal{NP}$-hardness *directly*, rather than as a special case of a more general study. The gains being, getting rid of the non-uniformity altogether, and obtaining a meaningful negative result for the case of general (adaptive) reductions. Specifically, working directly with one-way functions allows us to consider *natural* special cases of potential one-way functions and to establish stronger negative results for them (*i.e.*, results regarding general rather than non-adaptive reductions).

In particular, we consider polynomial-time computable functions f for which, given an image y, one can verify the number of preimages of y under f via a constant-round protocol. We call such functions size-verifiable, and show that the complexity of inverting them resembles the complexity of inverting polynomial-time computable permutations (and is separated from the complexity of inverting general polynomial-time computable functions, see Remark 7).

Indeed, the simplest case of size-verifiable functions is that of a permutation (*i.e.*, a length-preserving 1-1 function). Another interesting special case is that of *regular* functions that have an efficiently recognizable range, where f is regular if each image of f has a number of preimages that is determined by the length of the image. We prove that any reduction (which may be *fully adaptive*) of $\mathcal{NP}$ to inverting such a function f implies co$\mathcal{NP} \subseteq \mathcal{AM}$. Indeed, this is a special case of our result that holds for any size-verifiable function f.

We remark that, in the context of cryptographic constructions, it has been long known that dealing with regular one-way functions is easier than dealing with general one-way functions (see, *e.g.*, [GKL, GIL+, DiIm, HHK+]). For example, constructions of pseudorandom generators were first shown based on one-way permutation [BlMi, Ya], followed by a construction that used regular one-way functions [GKL], and culminated in the complex construction of [HILL] that uses any one-way function. Our work shows that regularity of a function (or, more generally, size-verifiability) is important also for classifying the complexity of inverting f, and not only the ease of using it within cryptographic constructions.

[1] We comment that the power of adaptive versus non-adaptive reductions has been studied in various works (*e.g.*, [FFLS, HNOS, BaLa]). It is known that if $NP \not\subseteq BPE$, then there exists a set in $\mathcal{NP} \setminus \mathcal{BPP}$ that is adaptively random self-reducible but not non-adaptively random self-reducible.

We believe that the study of the possibility of basing one-way functions on worst-case $\mathcal{NP}$-hardness is the most important motivation for the study of worst-case to average-case reductions for $\mathcal{NP}$. In such a case, one should consider the possible gain from studying the former question directly, rather than as a special case of a more general study. We believe that the results presented in this paper indicate such gains. We hope that this framework may lead to resolving the general question of the possibility of basing the existence of *general* one-way functions on worst-case $\mathcal{NP}$-hardness via *general* reductions.

1.4 Techniques

Our results are proved by using the hypothetical existence of corresponding reductions in order to construct constant-round interactive proof systems for co$\mathcal{NP}$ (and using [Ba85, GoSi] to conclude that co$\mathcal{NP} \subseteq \mathcal{AM}$). Towards this end we develop constant-round protocols for verifying the size of various "NP-sets" (or rather to sets of NP-witnesses for given instances in some predetermined $\mathcal{NP}$-sets).[2] Recall that lower-bound protocols for this setting are well-known (cf., *e.g.*, Goldwasser and Sipser [GoSi] and [GVW]), but the known upper-bound protocol of Fortnow [Fo] (see also [AiHa, BoTr]) works only when the verifier knows a "secret" element in the set. The latter condition severely limits the applicability of this upper-bound protocol, and this is the source of all technical difficulties encountered in this work.

To overcome the aforementioned difficulties, we develop two new constant-round protocols for upper bounding the sizes of $\mathcal{NP}$ sets. These protocols suffice for our applications, and may be useful also elsewhere. The two protocols are inspired by the works of Feigenbaum and Fortnow [FeFo] and Bogdanov and Trevisan [BoTr], respectively, and extend the scope of the original ideas.

The first protocol, called *confidence by comparison*, significantly extends the main idea of Feigenbaum and Fortnow [FeFo]. The common setting consists of a verifier that queries a prover such that the following two conditions hold:

1. The prover may cheat (without being detected) only in "one direction": For example, in the decision problem setting of [FeFo], the prover may claim that some yes-instances (of an NP-set) are no-instances (but not the other way around since it must support such claims by NP-witnesses). In our setting (of verifying set sizes) the prover may claim that sets are smaller than their actual size, but cannot significantly overestimate the size of sets (due to the use of a lower-bound protocol).

2. The verifier can obtain (reliable) statistics regarding the distribution of answers to random instances. In [FeFo] the relevant statistics is the frequency of yes-instances in the distribution of instances of a certain size, which in turn is provided as non-uniform advice. In our setting the statistics is the expected logarithm of the size of a random set (drawn from some distribution), and this statistics can be generated by randomly selecting sets such that the upper-bound protocol of [Fo] (and not merely the lower-bound protocols of [GoSi, GVW]) can be applied.

Combining the limited ("one directional") cheating of Type 1 with the statistics of Type 2 yields approximately correct answers for the questions that the verifier cares about. In [FeFo] this means that almost all queried instances are characterized correctly, while in our setting it means that for almost all sets sizes we obtain good approximations.

[2] That is, for a witness relation R that corresponds to some $\mathcal{NP}$-set $S = \{x : \exists y \ (x,y) \in R\}$, we consider the sets $R(x) = \{y : (x,y) \in R\}$ for various $x \in S$.

The second protocol abstracts a central idea of Bogdanov and Trevisan [BoTr], and is based on "hiding" (from the prover) queries of interest among queries drawn from a related distribution. This protocol can be used whenever an "NP-set" is drawn from a distribution D and the verifier can also sample sets from another distribution $\widetilde{D}$ that has the following two properties: (a) There exists an constant-round protocol for proving upper bounds on sets drawn from $\widetilde{D}$, and (b) the distribution $\widetilde{D}$ dominates D in the sense that $\Pr_{S\sim D}[S] \leq \lambda \cdot \Pr_{S\sim\widetilde{D}}[S]$, where λ is polynomial in the relevant efficiency parameter. We stress that the protocol postulated in (a) need not be the upper-bound protocol of [Fo]; it may also be a confidence by comparison protocol as outlined in the previous paragraph.

1.5 Organization

In Section 2, we provide an overview of our proofs as well as a formal statement of our main results. Detailed proofs can be found in the the following sections (*i.e.*, preliminaries are in Section 3, the treatment of adaptive reductions is in Section 4, and the treatment of general functions is in Section 5). In Section 6 we discuss possible interpretations of our negative results.

2 Overview of Results and Proofs

Having observed the potential benefit of working explicitly with the problem of inverting a polynomial-time computable function f, materializing this benefit represents the bulk of the technical challenge and the technical novelty of the current work.

Let us first clarify what we mean by saying that a decision problem L is (efficiently and randomly) reducible to the problem of inverting a (polynomial-time computable) function f. We take the straightforward interpretation (while using several arbitrary choices, like in setting the threshold determining the definition of an inverting oracle):

Definition 1 (inverting oracles and reductions). *A function $\mathcal{O} : \{0,1\}^* \to \{0,1\}^*$ is called an* (average-case) f-inverting oracle *if, for every n, it holds that $\Pr[\mathcal{O}(f(x)) \in f^{-1}(f(x))] \geq 1/2$, where the probability is taken uniformly over $x \in \{0,1\}^n$.*

For a probabilistic oracle machine R, we denote by $R^{\mathcal{O}}(w)$ a random variable representing the output of R on input w and access to oracle $\mathcal{O}$, where the probability space is taken uniformly over the probabilistic choices of machine R (i.e., its randomness*).*

A probabilistic polynomial-time oracle machine R is called a reduction of L to (average-case) inverting f *if, for every $w \in \{0,1\}^*$ and any f-inverting oracle $\mathcal{O}$, it holds that $\Pr[R^{\mathcal{O}}(w) = \chi_L(w)] \geq 2/3$, where $\chi_L(w) = 1$ if $w \in L$ and $\chi_L(w) = 0$ otherwise.*

A reduction as in Definition 1 may only establish that f is a *i.o. and weak*[3] one-way function (*i.e.*, that f cannot be inverted with probability exceeding 1/2 *on every input length*), which makes our impossibility results even stronger. Throughout this work, the function f will always

[3] In contrast, the standard definition of one-way function requires that any efficient inverting algorithm succeeds with negligible probability (*i.e.*, probability that is smaller than $1/\text{poly}(n)$ *on all but finitely many n's*). Here we relax the security requirement in two ways (by requiring more of a successful inverting algorithm): first, we require that the inverting algorithm be successful *on any input length* (hence hardness only occurs i.o.), and second that the success probability exceeds 1/2 rather than an arbitrary small $1/\text{poly}(n)$ (hence the term "weak").

be polynomial-time computable, and for simplicity we will also assume that it is length preserving (*i.e.*, $|f(x)| = |x|$ for all x).

Let us take a closer look at the reduction R. On input w, it may ask polynomially many queries to the inverting oracle. In adaptive reductions, later queries may depend on the oracle answers to earlier queries. In non-adaptive reductions all queries are computed in advance (based solely on the input w and the randomness of the reduction, denoted r). For simplicity, we will assume throughout this section that all queries are of length $|w|$.

Suppose, that there exists a reduction R from deciding membership in L to inverting the function f. We aim to use this reduction to give an constant-round protocol for $\overline{L}$, and conclude that if L is NP-complete (or just NP-hard) then $\text{co}\mathcal{NP} \subseteq \mathcal{AM}$. (We mention that a similar constant-round protocol can be given for L itself, but we have no need for the latter protocol.)

As in [FeFo, BoTr], the main backbone of our constant-round protocol for $\overline{L}$ is an emulation of the reduction R on input w (*i.e.*, the common input of the protocol), which in turn yields an output indicating whether or not $w \in \overline{L}$. Of course, the verifier cannot emulate the reduction on its own, because the reduction requires access to an f-inverting oracle. Instead, the prover will play the role of the inverting oracle, thus enabling the emulation of the reduction. Needless to say, the verifier will check that all answers are actually f-preimages of the corresponding queries (and for starters we will assume that all queries are in the image of f). Since we aim at a constant-round protocol, we send all queries to the prover in one round, which in the case of an adaptive reduction *requires* sending the randomness r of the reduction to the prover. Note that also in the non-adaptive case, we may as well just send r to the prover, because the prover may anyhow be able to determine r from the queries.

The fact that r is given (explicitly or implicitly) to the prover is the source of all difficulties that follow. It means that the prover need not answer the queries obliviously of other queries (or of r), but may answer the queries depending on r. In such a case, the prover's answers (when considering all possible r) are not consistent with any single oracle. Indeed, all these difficulties arise only in case f is not 1-1 (and indeed in case f is 1-1 the correct answer is fully determined by the query). We stress that the entire point of this study is the case in which f is not 1-1. In the special case that f is 1-1 (and length preserving), inverting f cannot be $\mathcal{NP}$-hard for rather obvious reasons (as has been well-known for a couple of decades; cf. [Br]).[4]

To illustrate what may happen in the general case, consider a 2-to-1 function f. Note that an arbitrary reduction of L to inverting f may fail in the rare case that the choice of the f-preimages returned by the oracle (*i.e.*, whether the query y is answered by the first or second element in $f^{-1}(y)$) matches the reduction's internal coin tosses.[5] This event may occur rarely in the actual reduction (no matter which f-inverting oracle it uses), but a cheating prover may always answer

[4]Intuitively, inverting such an f (which is a search problem in which each instance has a unique solution) corresponds to a decision problem in $\mathcal{NP} \cap \text{co}\mathcal{NP}$ (*i.e.*, given (y, i) determine the i-th bit of $f^{-1}(y)$). Thus, the fact that inverting f cannot be $\mathcal{NP}$-hard (unless $\mathcal{NP} = \text{co}\mathcal{NP}$) is analogous to the fact that sets in $\mathcal{NP} \cap \text{co}\mathcal{NP}$ cannot be $\mathcal{NP}$-hard (again, unless $\mathcal{NP} = \text{co}\mathcal{NP}$). In contrast, in case f is not 1-1, the corresponding decision problems are either not known to be in $\mathcal{NP} \cap \text{co}\mathcal{NP}$ or are *promise problems* (cf. [ESY]) in the "promise problem class" analogue of $\mathcal{NP} \cap \text{co}\mathcal{NP}$. Recall that promise problems in the latter class *may be* $\mathcal{NP}$-hard even if $\mathcal{NP} \neq \text{co}\mathcal{NP}$ (see [ESY]).

[5]For example, given an arbitrary reduction of L to inverting f, consider a modified reduction that tosses n additional coins $\rho_1, ..., \rho_n$, issues n additional queries, and halts without output if and only if for $i = 1, ..., n$ the i-th additional query is answered with the $(\rho_i + 1)$-st corresponding preimage (in lexicographic order). This reduction works with probability that is very close to the original one, but a cheating prover can always cause its emulation to halt without output.

in a way that matches the reduction's coins (hence violating the soundness requirement of the protocol).

A different way of looking at things is that the reduction guarantees that, for any adequate (f-inverting) oracle $\mathcal{O}$, with probability $2/3$ over the choices of r, machine R decides correctly when given oracle access to $\mathcal{O}$. However, it is possible that for every r there exists an oracle $\mathcal{O}_r$ such that R, when using coins r, decides incorrectly when given oracle access to $\mathcal{O}_r$. If this is the case (which we cannot rule out) then the prover may cheat by answering like the bad oracle $\mathcal{O}_r$.

In the rest of this section, we provide an outline of how we deal with this difficulty in each of the two cases (*i.e.*, size-verifiable functions and non-adaptive reductions).

2.1 Size-Verifiable Functions (Adaptive Reductions)

Recall that our aim is to present an constant-round protocol for $\overline{L}$, when we are given a general (adaptive) reduction R of the (worst-case) decision problem of L to inverting f. We denote by q the number of queries made by R, by $R(w, r, a_1, ..., a_{i-1})$ the i-th query made by R on input w and randomness r after receiving the oracle answers $a_1, ..., a_{i-1}$, and by $R(w, r, a_1, ..., a_q)$ the corresponding final decision. Recall that for simplicity, we assume that all queries are of length $n \stackrel{\text{def}}{=} |w|$. In the bulk of this subsection we assume that, given y, one can efficiently determine $|f^{-1}(y)|$.

As a warm-up we first assume that $|f^{-1}(y)| \leq \text{poly}(|y|)$, for every y. In this case, on common input w, the parties proceed as follows.

1. The verifier selects uniformly coins r for the reduction, and sends r to the prover.
2. Using r, the prover emulates the reduction as follows. When encountering a query y, the prover uses the lexicographically first element of $f^{-1}(y)$ as the oracle answer (and uses $\perp$ if $f^{-1}(y) = \phi$). Thus, it obtains the corresponding list of queries $y_1, ..., y_q$, which it sends to the verifier along with the corresponding sets $f^{-1}(y_1), ..., f^{-1}(y_q)$.
3. Upon receiving $y_1, ..., y_q$ and $A_1, ..., A_q$, the verifier checks, for every i, that $|A_i| = |f^{-1}(y_i)|$ and that $f(x) = y_i$ for every $x \in A_i$. Letting a_i denote the lexicographically first element of A_i, the verifier checks that $R(w, r, a_1, ..., a_{i-1}) = y_i$ for every i. The verifier accepts w (as a member of $\overline{L}$) if and only if all checks are satisfied and $R(w, r, a_1, ..., a_q) = 0$.

Note that the checks performed by the verifier "force" the prover to emulate a uniquely determined (perfect) inverting oracle (*i.e.*, one that answers each query y with the lexicographically first element of $f^{-1}(y)$). Thus, the correctness of the reduction implies the completeness and soundness of the above constant-round protocol.

In general, however, the size of $f^{-1}(y)$, for y in the range of f may not be bounded by a polynomial in n (where $n = |y| = |w|$). In this case, we cannot afford to have $f^{-1}(y)$ as part of a message in the protocol (because it is too long). The natural solution is to have the verifier send a random hash function $h : \{0,1\}^n \to \{0,1\}^\ell$, where $\ell = \lfloor (\log_2 |f^{-1}(y)|/\text{poly}(n)) \rfloor$, and let the prover answer with $h^{-1}(0^\ell) \cap f^{-1}(y)$ (rather than with $f^{-1}(y)$). The problem is that in this case the verifier cannot check the "completeness" of the list of preimages (because it cannot compute $|h^{-1}(0^\ell) \cap f^{-1}(y)|$), which allows the prover to omit a few members of $h^{-1}(0^\ell) \cap f^{-1}(y)$ at its choice. Recall that this freedom of choice (of the prover) may obliterate the soundness of the protocol.

The solution is that, although we have no way of determining the size of $h^{-1}(0^\ell) \cap f^{-1}(y)$, we do know that its expected size is exactly $|f^{-1}(y)|/2^\ell$, where the expectation is taken over the choice of h (assuming that a random h maps each point in $\{0,1\}^n$ uniformly on $\{0,1\}^\ell$). Furthermore, the prover cannot add elements to $h^{-1}(0^\ell) \cap f^{-1}(y)$ (because the verifier can verify membership in this set), it can only omit elements. But if the prover omits even a single element, it ends-up sending a set that is expected to be noticeably smaller than $|f^{-1}(y)|/2^\ell$ (because the expected size of $h^{-1}(0^\ell) \cap f^{-1}(y)$ is a polynomial in n). Thus, if we repeat the process many times, the prover cannot afford to cheat in most of these repetitions, because in that case the statistics will deviate from the expectation by too much.

Before turning to the specific implementation of this idea, we mention that the above reasoning corresponds to the confidence by comparison paradigm outlined in Section 1.4. Specifically, the prover may cheat (without being detected) only in one direction; that is, the prover may send a proper subset of a set of preimages under f and h (rather than the set itself), but it cannot send elements not in this set because membership in the set is efficiently verifiable by the verifier.

In the following protocol we use families of hash functions of very high quality (*e.g.*, poly(n)-wise independent ones). Specifically, in addition to requiring that a random $h : \{0,1\}^n \to \{0,1\}^\ell$ maps each point uniformly, we require that, for a *suitable polynomial* p and for any $S \subseteq \{0,1\}^n$ of size at least $p(n) \cdot 2^\ell$, with overwhelmingly high probability over the choice of h it is the case that $|h^{-1}(0^\ell) \cap S| < 2|S|/2^\ell$. In particular, the probability that this event does not occur is so small that, when conditioning on this event, the expected size of $|h^{-1}(0^\ell) \cap S|$ is $(1 \pm 2^{-n}) \cdot |S|/2^\ell$. (Thus, under this conditioning and for S as above, the variance of $2^\ell |h^{-1}(0^\ell) \cap S|/|S|$ is smaller than 2.)

1. The verifier selects uniformly $m = n \cdot q^2 p(n)^2 = \text{poly}(n)$ sequences of coins, $r^{(1)}, ..., r^{(m)}$ for the reduction, and sends them to the prover. In addition, for each $k = 1, ..., m$, $i = 1, ..., q$ and $\ell = 1, ..., n$, it selects and sends a random hash function $h_{k,i,\ell} : \{0,1\}^n \to \{0,1\}^\ell$.

   To streamline the following description, for $j \leq 0$, we artificially define $h_{k,i,j}$ such that $h_{k,i,j}^{-1}(0^j) \stackrel{\text{def}}{=} \{0,1\}^n$. In such a case, $S \cap h_{k,i,j}^{-1}(0^j) = S$, and so an instruction to do something with the former set merely means using the latter set.

2. For every $k = 1, ..., m$, the prover uses $r^{(k)}$ to emulate the reduction as follows. When encountering the i-th query, $y_i^{(k)}$, it determines $\ell_i^{(k)} = \lfloor (\log_2 |f^{-1}(y_i^{(k)})|/p(n)) \rfloor$, and uses the lexicographically first element of $f^{-1}(y_i^{(k)}) \cap h_{k,i,\ell_i^{(k)}}^{-1}(0^{\ell_i^{(k)}})$ as the oracle answer (and uses $\perp$ if the latter set is empty).[6] Thus, it obtains the corresponding list of queries $y_1^{(k)}, ..., y_q^{(k)}$, which it sends to the verifier along with the corresponding sets $f^{-1}(y_1^{(k)}) \cap h_{k,1,\ell_1^{(k)}}^{-1}(0^{\ell_1^{(k)}}), ..., f^{-1}(y_q^{(k)}) \cap h_{k,q,\ell_q^{(k)}}^{-1}(0^{\ell_q^{(k)}})$.

   We assume that none of the latter sets has size greater than $4p(n)$. Note that the bad event occurs with negligible probability, and in such a case the prover halts and the verifier rejects. (Otherwise, all mq sets are sent in one message.)

[6]Note that if $|f^{-1}(y_i^{(k)})| = 0$ then the oracle answer is defined as $\perp$. The formally inclined reader may assume that in this case $\log_2 0$ is defined arbitrarily.

3. Upon receiving $y_1^{(1)},...,y_q^{(1)},...,y_1^{(m)},...,y_q^{(m)}$ and $A_1^{(1)},...,A_q^{(1)},...,A_1^{(m)},...,A_q^{(m)}$, the verifier conducts the following checks:

   (a) For every $k = 1,...,m$ and $i = 1,...,q$, the verifier checks that for every $x \in A_i^{(k)}$ it holds that $f(x) = y_i^{(k)}$ and $h_{k,i,\ell_i^{(k)}}(x) = 0^{\ell_i^{(k)}}$, where $\ell_i^{(k)} = \lfloor(\log_2|f^{-1}(y_i^{(k)})|/p(n))\rfloor$ is efficiently computable due to the "size-computation" hypothesis. Letting $a_i^{(k)}$ be the lexicographically first element of $A_i^{(k)}$, it checks that $R(w, r^{(k)}, a_1^{(k)}, ..., a_{i-1}^{(k)}) = y_i^{(k)}$.

   (b) For every $i = 1,...,q$, it checks that

$$\frac{1}{m} \cdot \sum_{k=1}^{m} \frac{2^{\ell_i^{(k)}} \cdot |A_i^{(k)}|}{|f^{-1}(y_i^{(k)})|} > 1 - \frac{1}{100q \cdot p(n)} \tag{1}$$

   where $0/0$ is defined as 1.

   The verifier accepts w if and only if all the foregoing checks are satisfied and it holds that $R(w, r^{(k)}, a_1^{(k)}, ..., a_q^{(k)}) = 0$ for a uniformly selected $k \in \{1,...,m\}$.

We first note that the additional checks added to this protocol have a negligible effect on the *completeness* condition: the probability that either $|f^{-1}(y_i^{(k)}) \cap h^{-1}_{k,i,\ell_i^{(k)}}(0^{\ell_i^{(k)}})| > 4p(n)$ for some i, k or that Eq. (1) is violated for some i is exponentially vanishing.[7] Turning to the soundness condition, we note that the checks performed by the verifier force the prover to use $A_i^{(k)} \subseteq T_i^{(k)} \stackrel{\text{def}}{=} f^{-1}(y_i^{(k)}) \cap h^{-1}_{k,i,\ell_i^{(k)}}(0^{\ell_i^{(k)}})$. Also, with overwhelmingly high probability, for every $i = 1, ..., q$, it holds that

$$\frac{1}{m} \cdot \sum_{k=1}^{m} \frac{2^{\ell_i^{(k)}} \cdot |f^{-1}(y_i^{(k)}) \cap h^{-1}_{k,i,\ell_i^{(k)}}(0^{\ell_i^{(k)}})|}{|f^{-1}(y_i^{(k)})|} < 1 + \frac{1}{100q \cdot p(n)} \tag{2}$$

Combining Eq. (1) and Eq. (2), and recalling that $A_i^{(k)} \subseteq T_i^{(k)}$ (and $|f^{-1}(y_i^{(k)})| < 2p(n) \cdot 2^{\ell_i^{(k)}}$), it follows that $(1/m) \cdot \sum_{k=1}^{m}(|T_i^{(k)} \setminus A_i^{(k)}|/2p(n)) < 2/(100q \cdot p(n))$ for every i. Thus, for each i, the probability over a random k that $A_i^{(k)} \neq T_i^{(k)}$ is at most $1/25q$. It follows that for a random k, the probability that $A_i^{(k)} = T_i^{(k)}$ for all i's is at least $1 - (1/25)$. In this case, the correctness of the reduction implies the soundness of the foregoing constant-round protocol.

The foregoing description presumes that the verifier can determine the size of the set of f-preimages of any string. The analysis can be easily extended to the case that the verifier can only check the correctness of the size claimed and proved by the prover. That is, we refer to the following definition.

[7] Recall that here we refer to the case that $A_i^{(k)} = f^{-1}(y_i^{(k)}) \cap h^{-1}_{k,i,\ell_i^{(k)}}(0^{\ell_i^{(k)}})$. Thus, regarding Eq. (1), we note that the l.h.s is the average of m independent random variables, each having constant variance. Applying Chernoff bound, the probability that Eq. (1) is violated is upper-bounded by $\exp(-\Omega(m/(100q \cdot p(n))^2)) = \exp(-\Omega(n))$.

Definition 2 (Size Verifiable)**.** *We say that a function* $f\colon \{0,1\}^* \to \{0,1\}^*$ *is* size verifiable *if there is a constant-round proof system for the set* $\{(y, |f^{-1}(y)|) : y \in \{0,1\}^*\}$.

A natural example of a function that is size verifiable (for which the relevant set is not known to be in $\mathcal{BPP}$) is the integer multiplication function. That is, we consider the function that maps pairs of integers (which are not necessarily prime or of the same length) to their product. In this case the set $\{(y, |f^{-1}(y)|) : y \in \{0,1\}^*\}$ is in $\mathcal{NP}$ (*i.e.*, the NP-witness is the prime factorization) but is widely believed not to be in $\mathcal{BPP}$ (*e.g.*, it is believed to be infeasible to distinguish product of two $(n/2)$-bit random primes from the product of three $(n/3)$-bit long random primes).

Theorem 3 (Adaptive Reductions)**.** *Unless* co$\mathcal{NP} \subseteq \mathcal{AM}$*, there exists no reduction* (even not an adaptive one) *from deciding an NP-hard language to inverting a* size-verifiable *polynomial-time computable function.*

In other words, it is unlikely that the existence of size-verifiable one-way functions can be based on NP-hardness. We note that the result can be extended to functions that are "approximately size-verifiable" (covering the "approximable preimage-size" function of [HHK+] as a special case). A formal description of these results appears in Section 4.

Remark 4. *The proof of Theorem 3 does not utilize the fact that the oracle accessed by the reduction is allowed to err on some of the queries. Thus, the proof holds also for the case of reductions to the task of inverting* f *in the worst-case* (*i.e.*, inverting f on every image)*. It follows that, unless* co$\mathcal{NP} \subseteq \mathcal{AM}$*, there exist no reductions from* $\mathcal{NP}$ *to inverting in the worst-case a size-verifiable polynomial-time computable function.*

2.2 Non-Adaptive Reductions (General Functions)

We now turn to outline the proof of our second main result. Here we place no restrictions on the function f, but do restrict the reductions.

Theorem 5 (General Functions)**.** *Unless* co$\mathcal{NP} \subseteq \mathcal{AM}$*, there exists no* non-adaptive *reduction from deciding an NP-complete language to inverting a polynomial-time computable function.*

Considering the constant-round protocol used in the adaptive case, we note that in the current case the verifier cannot necessarily compute (or even directly verify claims about) the size of sets of f-preimages of the reduction's queries. Indeed, known lower-bound protocols (cf. [GoSi]) could be applied to these sets, but known upper-bound protocols (cf. [Fo]) cannot be applied because they require that the verifier has a random secret member of these sets. Fortunately, using the techniques described in Section 1.4 allows to overcome this difficulty (for the case of non-adaptive reductions), and to obtain constant-round protocols (rather than merely non-uniform constant-round protocols) for co$\mathcal{NP}$ (thus, implying co$\mathcal{NP} \subseteq \mathcal{AM}$).

Here R is a *non-adaptive* reduction of some set $L \in \mathcal{NP}$ to the average-case inverting of an arbitrary (polynomial-time computable) function f, and our goal again is to show that $\overline{L} \in \mathcal{AM}$. We may assume, without loss of generality, that the queries of $R(w, \cdot)$ are *identically distributed* (but typically *not* independently distributed), and represent this distribution by the random variable

R_w; that is, $\Pr[R_w = y] = |\{r \in \{0,1\}^{n'} : R(w,r) = y\}|/2^{n'}$, where n' denotes the number of coins used by $R(w,\cdot)$.

Actually, our constructions do not rely on the non-adaptivity of the reduction R, but rather on the fact that its queries are distributed accordiong to a single distribution (*i.e.*, R_w) that depends on w. We note that the treatment can be extended to the case that, for every i, the i-th query of R is distributed in a manner that depends only on w and i (but not on the answers to prior queries).

We first consider the (natural) case that R's queries are distributed identically to $F_n \stackrel{\rm def}{=} f(U_n)$, where U_n denotes the uniform distribution over $\{0,1\}^n$. Augmenting the protocol (for the general case) presented in Section 2.1, we require the prover to provide $|f^{-1}(y_i^{(k)})|$ along with each query $y_i^{(k)}$ made in the emulation of $R(w, r^{(k)})$.[8] In order to verify that the claimed set sizes are approximately correct, we require the prover to provide lower-bound proofs (cf., [GoSi]) and employ the confidence by comparison paradigm. Specifically, to prevent the prover from understating these set sizes, we compare the value of $(1/qm) \cdot \sum_{i=1}^{q} \sum_{k=1}^{m} \log_2 |f^{-1}(y_i^{(k)})|$ to the expected value of $\log_2 |f^{-1}(f(U_n))|$, where here and below we define $\log_2 0$ as -1 (in order to account for the case of queries that have no preimages). Analogously to [FeFo], one may suggest that the latter value (*i.e.*, $\mathbf{E}[\log_2 |f^{-1}(F_n)|]$) be given as a non-uniform advice, but we can do better: We require the prover to supply $\mathbf{E}[\log_2 |f^{-1}(f(U_n))|]$ and prove its approximate correctness using the following protocol.

> The verifier uniformly selects $x_1, ..., x_m \in \{0,1\}^n$, computes $y_i = f(x_i)$ for every i, sends $y_1, ..., y_m$ to the prover and asks for $|f^{-1}(y_1)|, ..., |f^{-1}(y_m)|$ along with lower and upper bound constant-round interactive proofs. (As usual, the lower-bound AM-protocol of [GoSi] (or [GVW]) can be applied because membership in the corresponding sets can be easily verified.) The point is that the upper-bound protocol of [Fo] can be applied here, because the verifier has secret random elements of the corresponding sets.

Recall that the lower-bound protocol (of [GoSi] or [GVW]) guarantees that the prover cannot overstate any set size by more than an $\varepsilon = 1/\mathrm{poly}(n)$ factor (without risking detection with overwhelmingly high probability). Thus, we will assume throughout the rest of this section that the prover never overstates set sizes (by more than such a factor). The analysis of understated set sizes is somewhat more delicate, firstly because (as noted) the execution of upper-bound protocols requires the verifier to have a secret random element in the set, and secondly because an understatement by a factor of ε is only detected with probability ε (or so). Still this means that the prover cannot significantly understate many sets sizes and go undetected. Specifically, if the prover understates the size of $f^{-1}(y_i)$ by more than an ε factor for at least n/ε of the y_i's then it gets detected with overwhelmingly high probability. Using a suitable setting of parameters, this establishes the value of $\mathbf{E}[\log_2 |f^{-1}(f(U_n))|]$ up to a sufficiently small additive term, which suffices for our purposes. Specifically, as in Section 2.1, such a good approximation of $\mathbf{E}[\log_2 |f^{-1}(f(U_n))|]$

[8] Actually, a small modification is required for handling the following subtle problem that refers to the possible control of the prover on the hashing function being used in its answer. Recall that the hashing function in use (for query y) is determined by $\ell = \lfloor(\log_2 |f^{-1}(y)|/p(n))\rfloor$, but in the setting of Section 2.1 the verifier knows $|f^{-1}(y)|$ and thus the prover has no control on the value of ℓ. In the current context, the prover may be able to cheat a little about the value of $|f^{-1}(y)|$, without being caught, and this may (sometimes) cause a change of one unit in the value of ℓ (and thus allow for a choice among two hash functions). We resolve this problem by having the verifier "randomize" the value of $|f^{-1}(y)|$ such that, with high probability, cheating a little about this value is unlikely to affect the value of ℓ. Specifically, as a very first step, the verifier selects uniformly $\rho \in [0,1]$ (and sends it to the prover), and the prover is asked to set $\ell = \lfloor(\rho + \log_2 s_y/p(n))\rfloor$ (rather than $\ell = \lfloor(\log_2 s_y/p(n))\rfloor$), where s_y is prover's claim regarding the size of $|f^{-1}(y)|$.

forces the prover not to understate the value of $|f^{-1}(y_i^{(k)})|$ by more than (say) a $1/10p(n)$ factor for more than (say) $m/10$ of the possible pairs (i,k). (Note that, unlike in Section 2.1, here we preferred to consider the sum over all (i,k)'s rather than q sums, each corresponding to a different i.)[9]

We now allow R_w to depend on w, but restrict our attention to the natural case in which the reduction does not ask a query y with probability that exceeds $\Pr[F_n = y]$ by too much. Specifically, suppose that $\Pr[R_w = y] \leq \text{poly}(|y|) \cdot \Pr[F_n = y]$, for every y. In this case, we modify the foregoing protocol as follows.

> Here it makes no sense to compare the claimed value of $(1/qm) \cdot \sum_{i=1}^{q} \sum_{k=1}^{m} \log_2 |f^{-1}(y_i^{(k)})|$ against $\mathbf{E}[\log_2 |f^{-1}(F_n)|]$. Instead we should compare the former (claimed) average to $\mathbf{E}[\log_2 |f^{-1}(R_w)|]$. Thus, the verifier needs to obtain a good approximation to the latter value. This is done by generating many y_i's as before (*i.e.*, $y_i = f(x_i)$ for uniformly selected $x_i \in \{0,1\}^n$) along with fewer but still many y_i's sampled from R_w, and sending all these y_i's (in random order) to the prover. Specifically, for $t \geq \max_{y \in \{0,1\}^*} \{\Pr[R_w = y]/\Pr[F_n = y]\}$, we generate t times more y_i's from F_n, and so each y_i received by the prover is at least as likely to come from F_n than from R_w.
>
> The prover will be asked to provide all $|f^{-1}(y_i)|$'s along with lower-bound proofs, and afterwards (*i.e.*, only after committing to these $|f^{-1}(y_i)|$'s) the verifier will ask for upper-bound proofs for those y_i's generated via F_n (for which the verifier knows a secret and uniformly distributed $x_i \in f^{-1}(y_i)$).

Recall that the prover cannot significantly overstate the size of any $|f^{-1}(y_i)|$ (*i.e.*, overstate it by more than an $\varepsilon = 1/\text{poly}(n)$ factor). If the prover significantly understates the sizes of too many of the $|f^{-1}(y_i)|$'s, then it is likely to similarly overstate also the sizes of many $|f^{-1}(y_i)|$'s that correspond to y_i's that were generated by sampling F_n. But in this case, with overwhelmingly high probability, the prover will fail in at least one of the corresponding upper-bound proofs.

We now allow R_w to depend arbitrarily on w, without any restrictions whatsoever. For a threshold parameter t to be determined later, we say that a query y is t-heavy if $\Pr[R_w = y] > t \cdot \Pr[F_n = y]$. (In the *special case*, we assumed that there are *no* $\text{poly}(n)$-heavy queries.) Observe that the probability that an element sampled according to F_n is t-heavy is at most $1/t$, and thus modifying an inverting oracle such that it answers t-heavy queries by $\perp$ effects the inverting probability of the oracle by at most $1/t$. Thus, for $t \geq 2$, if we answer t-heavy queries by $\perp$ (and answer other f-images with a preimage), then we emulate a legitimate inverting oracle (which inverts f with probability at least $1/2$) and the reduction R is still supposed to work well.[10] Referring to y as t-light if it is not t-heavy, we note that t-light queries can be handled as in the foregoing *special case* (provided $t \leq \text{poly}(n)$), whereas t-heavy queries are accounted for by the previous discussion. The problem is to determine whether a query is t-heavy or t-light, and certainly we have no chance of doing so if many (reduction) queries are very close to the threshold (*e.g.*, if

[9] We stress that in both cases both choices can be made. We note that, when analyzing the completeness condition, one may prefer to analyze the deviation of the individual sums (for each i).

[10] This is the first (and only) place where we use the average-case nature of the reduction R.

$\Pr[R_w = y] = (t \pm n^{-\omega(1)}) \cdot \Pr[F_n = y]$ for all y's). Thus, as in [BoTr], we select the threshold at random (say, uniformly in the interval $[2, 3]$). Next, we augment the foregoing protocol as follows.

- We ask the prover to provide for each query $y_i^{(k)}$, also the value of $\Pr[R_w = y_i^{(k)}]$, or equivalently the size of $\{r : R(w, r) = y_i^{(k)}\}$. In addition, we ask for lower-bound proofs of these set sizes.

- Using lower and upper bound protocols (analogously to the *simple case*)[11], we get an estimate of $\mathbf{E}[\log_2 |\{r : R(w, r) = R_w\}|]$. We let the verifier check that this value is sufficiently close to the claimed value of $(1/qm) \cdot \sum_{i=1}^{q} \sum_{k=1}^{m} \log_2 |\{r : R(w, r) = y_i^{(k)}\}|$, thus preventing an understating of the size of almost all the sets $\{r : R(w, r) = y_i^{(k)}\}$.

  Hence, combining these two items, the verifier gets a good estimate of the size of $\{r : R(w, r) = y_i^{(k)}\}$ for all but few (i, k)'s. That is, the verifier can confirm that for almost all the (i, k)'s the claimed (by prover) size of $\{r : R(w, r) = y_i^{(k)}\}$ is approximately correct.

- Using the claimed (by the prover) values of $\Pr[R_w = y_i^{(k)}]$ and $\Pr[F_n = y_i^{(k)}]$, the verifier makes tentative decisions regarding which of the $y_i^{(k)}$'s is t-light.

  Note that for most (i, k), the prover's claim about $\Pr[R_w = y_i^{(k)}]$ is approximately correct, whereas the claim about $\Pr[F_n = y_i^{(k)}]$ can only be understated (by virtue of the lower-bound protocol employed for the set $f^{-1}(y_i^{(k)})$).

  Using a protocol as in the *special case*, the verifier obtains an estimate of $\mathbf{E}[\log_2 |f^{-1}(R'_w)|]$, where R'_w denotes R_w conditioned on being t-light, and checks that this value is sufficiently close to the claimed average of $\log_2 |f^{-1}(y_i^{(k)})|$, taken only over t-light $y_i^{(k)}$'s. In addition, the verifier checks that the fraction of t-light $y_i^{(k)}$'s (among all $y_i^{(k)}$'s) approximates the probability that R_w is t-light.

  We note that estimating $\mathbf{E}[\log_2 |f^{-1}(R'_w)|]$ is done by generating y_i's as in the *special case*, but with $t \in [2, 3]$ as determined above, and while asking for the value of both $\Pr[R_w = y_i]$ and $\Pr[F_n = y_i]$ for all y_i's, and afterwards requiring upper-bound proofs for one of these values depending on whether y_i was sampled from R_w or F_n. These values will serve as basis for determining whether each y_i is t-heavy or t-light, and will also yield an estimate of the probability that R_w is t-light.

Recall that the verifier accepts w if and only if all the foregoing checks (including the ones stated in the adaptive case) are satisfied.

Ignoring the small probability that we selected a bad threshold t as well as the small probability that we come across a query that is close to the threshold, we analyze the foregoing protocol as follows. We start by analyzing the queries y_i's used in the sub-protocol for estimating $\mathbf{E}[\log_2 |f^{-1}(R'_w)|]$. We first note that, by virtue of the lower and upper bound proofs, for almost all queries y_i's generated by R_w, the sizes of $\{r : R(w, r) = y_i\}$ must be approximately correct. Next, employing a reasoning as in the *special case*, it follows that for almost all t-light queries y_i's we obtain correct estimates of the size of their f-image (*i.e.*, we verify that almost all the sizes

[11] In the simple case we got an estimate of $\mathbf{E}[\log_2 |f^{-1}(F_n)|]$, while relying on our ability to generate samples of F_n along with a uniformly distributed member of $f^{-1}(F_n)$. Here we rely on our ability to generate samples of R_w along with a uniformly distributed member of $\{r : R(w, r) = R_w\}$.

claimed by the prover for the $|f^{-1}(y_i)|$'s are approximately correct). It follows that we correctly characterize almost all the t-light y_i's generated by R_w as such. As for (almost all) t-heavy queries y_i's generated by R_w, we may wrongly consider them t-light only if the prover has significantly overstated the size of their preimage, because we have a good estimate of $\{r : R(w,r) = y_i^{(k)}\}$ for (almost all) these y_i's. Recalling that an overstatement of $|f^{-1}(y_i^{(k)})|$ is detected with overwhelmingly high probability (by the lower-bound protocol), it follows that almost all t-heavy queries y_i's generated by R_w are correctly characterized as such. Thus, the characterization of almost all y_i's (generated by R_w) as t-light or t-heavy is correct, and so is the estimate of the probability that R_w is t-light. Recalling that for almost all the t-light y_i's generated by R_w we have a correct estimate of $|f^{-1}(y_i)|$, we conclude that the estimate of $\mathbf{E}[\log_2 |f^{-1}(R'_w)|]$ is approximately correct.

Next we employ parts of the foregoing reasoning to the $y_i^{(k)}$'s. Recalling that, for almost all queries $y_i^{(k)}$, we obtained correct estimates of the size of $\{r : R(w,r) = y_i^{(k)}\}$ (and that $|f^{-1}(y_i^{(k)})|$ cannot be overstated), we conclude that we correctly characterize almost all t-heavy queries as such. The comparison to the estimated probability that R_w is t-light guarantees that the prover cannot claim too many t-light $y_i^{(k)}$'s as t-heavy, which implies that we have correctly characterized almost all $y_i^{(k)}$'s as t-light or t-heavy. Recalling that $|f^{-1}(y_i^{(k)})|$ can only be understated (due to the lower-bound proofs) and using the estimate of $\mathbf{E}[\log_2 |f^{-1}(R'_w)|]$ as an approximate lower-bound, it follows that the claims made regarding almost all the $|f^{-1}(y_i^{(k)})|$'s are approximately correct. Thus, as in the special case, the correctness of the reduction implies the completeness and soundness of the foregoing constant-round protocol. A formal description of this result appears in Section 5.[12]

Remark 6. *In contrast to Remark 4, dealing with general one-way functions (even in the non-adaptive case) requires referring to the average-case nature of the reduction; that is, we must use the hypothesis that the reduction yields the correct answer even in case that the inverting oracle fails on some inputs (as long as the measure of such inputs is adequately small). This* average-case *hypothesis is required since there exist reductions from $\mathcal{NP}$ to inverting in the* worst-case *some* (general) *polynomial-time computable function* (see [Go, Chap. 2, Exer. 3]).

3 Preliminaries

Omitted. Currently this section is in draft form.

4 Size-Verifiable Functions (Adaptive Reductions)

Omitted. Currently this section is in draft form.

5 Non-Adaptive Reductions (General Functions)

Omitted. Currently this section is in draft form.

[12] We remark that the description in Section 5 differs from the above description on some technicalities.

6 Discussion: interpretations of our negative results

Negative results of the type obtained in this work (as well as in [FeFo, BoTr]) can be interpreted in several ways: The straightforward view is that such results narrow down the means by which one can base one-way functions on $\mathcal{NP}$-hardness. Namely, under the assumption that co$\mathcal{NP}$ is not contained in $\mathcal{AM}$, our results show that (1) *non-adaptive* randomized reductions are not suitable for basing one-way functions on $\mathcal{NP}$-hardness, and (2) that one-way functions based on $\mathcal{NP}$-hardness can not be size verifiable (*e.g.*, cannot be regular with an efficiently recognizable range).

Another interpretation is that these negative results are an indication that (worst-case) complexity assumptions regarding $\mathcal{NP}$ as a whole (*i.e.*, $\mathcal{NP} \not\subseteq \mathcal{BPP}$) are not sufficient to base one-way functions on. But this does not rule out the possibility of basing one-way functions on the worst-case hardness of a subclass of $\mathcal{NP}$ (*e.g.*, the conjecture that $\mathcal{NP} \cap \text{co}\mathcal{NP} \not\subseteq \mathcal{BPP}$). This is the case because our results (as previous ones) actually show that certain reductions of the (worst-case) decision problem of a set S to (average-case) inverting of f imply that $S \in \mathcal{AM} \cap \text{co}\mathcal{AM}$. But no contradiction is obtained if S belongs to $\mathcal{NP} \cap \text{co}\mathcal{NP}$ anyhow. Indeed, the decision problems related to lattices that are currently known to have worst-case to average-case reductions belong to $\mathcal{NP} \cap \text{co}\mathcal{NP}$ (cf. [Aj, MiRe] versus [AhRe]).

Yet another interpretation is that these negative results suggest that we should turn to a more relaxed notion of a reduction, which is uncommon in complexity theory and yet is applicable in the current context. We refer to "non black-box" reductions in which the reduction gets the code (of the program) of a potential probabilistic polynomial-time inverting algorithm (rather than black-box access to an arbitrary inverting oracle). The added power of such (security) reductions was demonstrated a few years ago by Barak [Ba01, Ba02].

Remark 7. *Recall that Remark 4 asserts that, unless* co$\mathcal{NP} \subseteq \mathcal{AM}$, *there exist no reductions from* $\mathcal{NP}$ *to inverting in the worst-case a size-verifiable polynomial-time computable function. In contrast, it is known that reductions do exist from* $\mathcal{NP}$ *to inverting in the worst-case some* (general) *polynomial-time computable function* (see [Go, Chap. 2, Exer. 3]). *This yields a (structural complexity) separation between size-verifiable polynomial-time computable functions on one hand and general polynomial-time computable functions on the other hand,* (assuming as usual co$\mathcal{NP} \not\subseteq \mathcal{AM}$).

Acknowledgments

The research of Adi Akavia was supported in part by NSF grant CCF0514167. The research of Oded Goldreich was partially supported by the Israel Science Foundation (grant No. 460/05). The research of Shafi Goldwasser was supported in part by NSF CNS-0430450, NSF CCF0514167, Sun Microsystems, and the Minerva Foundation. Dana Moshkovitz is grateful to Muli Safra for supporting her visit to MIT, where this research has been initiated.

References

[AhRe] D. Aharonov and O. Regev. Lattice Problems in NP intersect coNP. In *45th IEEE Symposium on Foundations of Computer Science*, 2004.

[AiHa] W. Aiello and J. Hastad. Perfect Zero-Knowledge Languages can be Recognized in Two Rounds. In *28th IEEE Symposium on Foundations of Computer Science*, pages 439–448, 1987.

[Aj] M. Ajtai. Generating hard instances of lattice problems. In *28th ACM Symposium on the Theory of Computing*, pages 99–108, 1996.

[Ba85] L. Babai. Trading Group Theory for Randomness. In *17th ACM Symposium on the Theory of Computing*, pages 421–429, 1985.

[BaMo] L. Babai and S. Moran. Arthur-Merlin Games: A Randomized Proof System and a Hierarchy of Complexity Classes. *Journal of Computer and System Science*, Vol. 36, pp. 254–276, 1988.

[BaLa] L. Babai and S. Laplante. Stronger seperations for random-self-reducability, rounds, and advice. In *IEEE Conference on Computational Complexity 1999*, pages 98–104, 1999.

[Ba01] B. Barak. How to Go Beyond the Black-Box Simulation Barrier. In *42nd IEEE Symposium on Foundations of Computer Science*, pages 106–115, 2001.

[Ba02] B. Barak. Constant-Round Coin-Tossing with a Man in the Middle or Realizing the Shared Random String Model. In *43th IEEE Symposium on Foundations of Computer Science*, pages 345–355, 2002.

[BCGL] S. Ben-David, B. Chor, O. Goldreich, and M. Luby. On the Theory of Average Case Complexity. *Journal of Computer and System Science*, Vol. 44, No. 2, April 1992, pages 193–219.

[BlMi] M. Blum and S. Micali. How to Generate Cryptographically Strong Sequences of Pseudo-Random Bits. *SIAM Journal on Computing*, Vol. 13, pages 850–864, 1984. Preliminary version in *23rd FOCS*, 1982.

[BoTr] A. Bogdanov and L. Trevisan. On worst-case to average-case reductions for NP problems. In *44th IEEE Symposium on Foundations of Computer Science*, pages 308–317, 2003.

[Br] G. Brassard. Relativized Cryptography. In *20th IEEE Symposium on Foundations of Computer Science*, pages 383–391, 1979.

[DiIm] G. Di-Crescenzo and R. Impagliazzo. Security-preserving hardness-amplification for any regular one-way function In *31st ACM Symposium on the Theory of Computing*, pages 169–178, 1999.

[ESY] S. Even, A.L. Selman, and Y. Yacobi. The Complexity of Promise Problems with Applications to Public-Key Cryptography. *Inform. and Control*, Vol. 61, pages 159–173, 1984.

[FeFo] J. Feigenbaum and L. Fortnow. Random self-reducibility of complete sets. *SIAM Journal on Computing*, Vol. 22, pages 994–1005, 1993.

[FFLS] J. Feigenbaum, L. Fortnow, C. Lund, and D. Spielman. The power of adaptiveness and additional queries in random self-reductions. *Computational Complexity*, 4:158–174, 1994.

[Fo] L. Fortnow, The Complexity of Perfect Zero-Knowledge. In [Mi], pages 327–343, 1989. Extended abstract in *19th STOC*, pages 204–209, 1987.

[Go] O. Goldreich. *Foundation of Cryptography – Basic Tools.* Cambridge University Press, 2001.

[GIL+] O. Goldreich, R. Impagliazzo, L.A. Levin, R. Venkatesan, and D. Zuckerman. Security Preserving Amplification of Hardness. In *31st IEEE Symposium on Foundations of Computer Science*, pages 318–326, 1990.

[GKL] O. Goldreich, H. Krawczyk and M. Luby. On the Existence of Pseudorandom Generators. *SIAM Journal on Computing*, Vol. 22, pages 1163–1175, 1993.

[GVW] O. Goldreich, S. Vadhan and A. Wigderson. On interactive proofs with a laconic provers. *Computational Complexity*, Vol. 11, pages 1–53, 2003.

[GoSi] S. Goldwasser and M. Sipser. Private Coins versus Public Coins in Interactive Proof Systems. In [Mi], pages 73–90, 1989. Extended abstract in *18th STOC*, pages 59–68, 1986.

[HHK+] I. Haitner, O. Horvitz, J. Katz, C.Y. Koo, R. Morselli, and R. Shaltiel. Reducing complexity assumptions for statistically-hiding commitment. In *Eurocrypt*, Springer, Lecture Notes in Computer Science (Vol. 3494), pages 58–77, 2005.

[HNOS] E. Hemaspaandra, A.V. Naik, M. Ogiwara, and A.L. Selman. P-Selective Sets, and Reducing Search to Decision vs. Self-reducibility. *Journal of Computer and System Science*, Vol. 53 (2), pages 194–209, 1996.

[HILL] J. Hastad, R. Impagliazzo, L.A. Levin and M. Luby. A Pseudorandom Generator from any One-way Function. *SIAM Journal on Computing*, Vol. 28, pages 1364–1396, 1999.

[ImLe] R. Impagliazzo and L.A. Levin. No Better Ways to Generate Hard NP Instances than Picking Uniformly at Random. In *31st IEEE Symposium on Foundations of Computer Science*, 1990, pages 812–821.

[KaTr] J. Katz and L. Trevisan. On The Efficiency Of Local Decoding Procedures For Error-Correcting Codes. In *32nd ACM Symposium on the Theory of Computing*, pages 80–86, 2000.

[Mi] S. Micali, editor. *Advances in Computing Research: a research annual*, Vol. 5 (Randomness and Computation), 1989.

[MiRe] D. Micciancio and O. Regev. Worst-case to Average-case Reductions Based on Gaussian Measures. In *45th IEEE Symposium on Foundations of Computer Science*, pages 372–381, 2004.

[Ya] A.C. Yao. Theory and Application of Trapdoor Functions. In *23rd IEEE Symposium on Foundations of Computer Science*, pages 80–91, 1982.