Innovations in Computer Science 2010

Hard Instances for Satisfiability and Quasi-one-way Functions

Andrej Bogdanov[1] Kunal Talwar[2] Andrew Wan[3]
[1]The Chinese University of Hong Kong
[2]Microsoft Research SVC
[3]Columbia University
andrejb@cse.cuhk.edu.hk kunal@microsoft.com atw12@columbia.edu

Abstract: We give an efficient algorithm that takes as input any (probabilistic) polynomial time algorithm A which purports to solve SAT and finds, for infinitely many input lengths, SAT formulas ϕ and witnesses w such that A claims ϕ is unsatisfiable, but w is a satisfying assignment for ϕ (assuming NP $\not\subseteq$ BPP). This solves an open problem posed in the work of Gutfreund, Shaltiel, and Ta-Shma (CCC 2005). Following Gutfreund et al., we also extend this to give an efficient sampling algorithm (a "quasi-hard" sampler) which generates hard instance/witness pairs for all algorithms running in some fixed polynomial time.
We ask how our sampling algorithm relates to various cryptographic notions. We show that our sampling algorithm gives a simple construction of *quasi-one-way functions*, a weakened notion of standard one-way functions. We also investigate the possibility of obtaining pseudorandom generators from our quasi-one-way functions and show that a large class of reductions that work in the standard setting must fail.

Keywords: average-case complexity; cryptography; cryptographic complexity

1 Introduction

A fundamental unresolved issue in the complexity theoretic foundations of modern cryptography is whether it is possible to base the security of cryptographic primitives, such as one-way functions, collision resistant hash functions, or zero knowledge, on strong *worst-case* hardness assumptions such as NP $\not\subseteq$ BPP. Despite some remarkable progress in lattice-based cryptography in the last decade, this objective remains elusive. A well-studied obstacle, articulated in a series of works [1–3], is basing the *average-case hardness* of NP-problems on the assumption that NP $\not\subseteq$ BPP. But this is not the only difficulty; Impagliazzo's influential position paper on average-case complexity [4] indicates that the question of equivalence of worst-case and average-case hardness for NP is separate from the question of basing one-way functions (and symmetric key encryption) on average-case hard problems. In particular, one-way functions require the ability to sample not only instances of hard problems but also their solutions.

In this work we aim to illuminate that aspect of the relationship between cryptography and worst-case hardness that is mostly unexplored in [1–3]. In particular, we focus on the following question: if we can sample hard instances of problems, how can we sample their solutions as well?

1.1 Sampling hard instances and their solutions

The basic goal of average-case complexity is to explain the hardness of computational problems where instances of the input are sampled efficiently from some distribution. Currently, we have no methods for arguing the hardness of such problems from worst-case complexity assumptions. Gutfreund, Shaltiel and Ta-Shma [5] proposed studying a substantially relaxed problem: instead of designing a single distribution that is hard for all potential algorithms of an NP-problem — in particular SAT — they ask whether it is possible to obtain a uniform family of hard distributions D_A, one for every potential algorithm A for SAT.

Gutfreund et al. showed how to obtain the distributions D_A: Assuming P $\neq$ NP, they give a sampling procedure that, given a candidate A for SAT, runs in time polynomial in the running time of A and outputs formulas ϕ on which A fails to solve SAT for infinitely many input lengths. Atserias [6] gave a variant of the result for nonuniform algorithms (assuming NP $\not\subseteq$ P/poly).

While the algorithm of Gutfreund et al. produces hard instances for the algorithm A, it does not produce certificates proving that those instances are hard. Indeed, they state it as an open problem (suggested by Adam Smith) to design a "dreambreaker": a procedure that outputs not just a satisfiable formula on which the algorithm fails, but also a satisfy-

ing assignment for that formula (i.e., a witness that the algorithm failed).

A dreambreaker procedure can be used to mechanically produce counterexamples to purported efficient SAT-solving algorithms (assuming P $\neq$ NP). Given a candidate efficient SAT algorithm A and a hardness parameter n, the dreambreaker will output a formula ϕ (of size polynomial in n) and an assignment a such that a satisfies ϕ, but the purported SAT algorithm A claims that ϕ is not satisfiable. Given the wide use of SAT solvers in the practice of software verification, AI, and operations research, a mechanical procedure for obtaining certified hard instances for them may be of interest.

Our first contribution is to construct dreambreakers against candidate search algorithms for SAT, both deterministic and randomized.

Following [5], we then use our dreambreakers to build **quasi-hard instance/solution distributions**: These are distributions produced by samplers that given any time bound $t(n)$ and a parameter n, run in time polynomial in $t(n)$ and output (with noticeable probability) formula/witness pairs that are hard for all candidate randomized search SAT algorithms that run in time $t(n)$, for infinitely many values of n.

1.2 Quasi-one-way functions and pseudorandom generators

The notion of a quasi-hard distribution is substantially weaker than the notion of a "hard distribution" in average-case complexity: samplers for quasi-hard distributions may take more time to run than the adversaries sampled against, while samplers for average-case hard problems have a fixed running time that can be exceeded by their adversaries. Yet, Gutfeund and Ta-Shma [7] showed that the techniques of [5], while falling short of giving average-case hard problems, bypass the worst-case to average-case barriers of [1] and [2].

Given that these methods manage to avoid some of the obstacles between worst-case and average-case complexity, we find it natural to ask whether they can also yield weakened forms of various cryptographic primitives. Roughly speaking, we consider primitives where the honest party may be given more resources than the adversary. (However, we argue that the interesting definitions are not obtained by merely inverting the order of quantifiers in the standard cryptographic definition.) We show that using quasi-hard distributions it is possible to construct *quasi-one-way functions*. These are functions which may take more time to compute than the adversaries they fool (as with standard one-way functions, an adversary is "fooled" if it can't invert), but those adversaries still have enough resources to verify candidate preimages to the function. See Section 2 for a formal definition and discussion.

Next, we consider pseudorandom generators [8, 9]. Relaxing the notion of a cryptographic pseudorandom generator into one where the generator has more time than the adversary *does* have a well motivated application – algorithmic derandomization. Indeed, a "relaxed" pseudorandom generator that takes more time to run than the adversary is nothing more than a pseudorandom generator of the Nisan-Wigderson type [10]. To be consistent with our terminology, for the purpose of this paper we will refer to pseudorandom generators of the Nisan-Wigderson type as *quasi-pseudorandom generators* (quasi-PRGs).

The construction of a polynomial stretch quasi-pseudorandom generator from the assumption P $\neq$ NP would yield a new, non-trivial derandomization of HeurBPP. Such a derandomization would be a 'low-level' analog of the Impagliazzo and Wigderson [11] construction, giving simulations of poly(n) bits with n bits of randomness from a stronger hardness assumption ([11] obtain a sub-exponential simulation, but only assume BPP $\neq$ EXP).

We do not know if a quasi-pseudorandom generator can be obtained from quasi-hard samplers, but we investigate the obstacles to applying the standard cryptographic technology for constructing pseudorandom generators from one-way functions [12, 13] to our setting. In particular, we show that no black-box reductions (we define these formally in Section 2) from distinguishing length-doubling pseudorandom generators to inverting quasi-one-way functions can exist.

2 Our Results

In this section we present formal definitions and state the theorems we prove. We first describe the sampling result.

2.1 Hard Instance/Solution Samplers

We consider search algorithms – it is well-known that an efficient SAT decision algorithm implies an efficient SAT search algorithm. In our proofs, we assume the search algorithm is a *canonical search algorithm*, i.e. that it checks that its output actually satisfies the input formula (so the algorithm only errs by outputting 0). This is without loss of generality, since any formula $\phi \in$ SAT that causes the modified algorithm to reject (incorrectly) also fools the original algorithm. We show how to construct "dream-breakers" for deterministic and randomized algorithms:

Theorem 1. *Assume* P $\neq$ NP *and let A be a polynomial-time search algorithm. There is a polynomial time procedure D, taking as input A, 1^n and $A's$ running time $t(n)$, that for infinitely many n, outputs a formula w_n of length n and witness a_n such that $A(w_n) = 0$ and a_n is a witness for w_n.*

Theorem 2. *Assume* NP $\not\subseteq$ BPP *and let A be a randomized polynomial-time search algorithm. There is a probabilistic polynomial time procedure D, taking as input A, 1^n and $A's$ running time $t(n)$, that for infinitely many n, outputs a formula w_n of length n and witness a_n such that with probability $1 - o(1)$, a_n is a witness for w_n and $\Pr[A(w_n) = 0] > 2/3$.*

We use these samplers use to obtain a "hard instance" generator that produces instances hard for all algorithms running in some fixed time $t(n)$, both in the deterministic and randomized settings.

Theorem 3. *Assume* NP $\not\subseteq$ BPP*. Then for every polynomial $t(n)$ there is a (randomized) polynomial-time function S, $S(1^n) \rightsquigarrow (\phi, w)$ such that (with probability $1 - 1/t(n)$) ϕ is satisfied by w, but $\Pr[A(\phi) = 0] > 2/3$ for any canonical (randomized) search algorithm A running in time $t(n)$.*

2.2 Quasi-one-way Functions

Next, we define quasi-one-way functions. A function is one-way in the standard sense if it is easy to compute $f(x)$ given inputs x but difficult to find preimages $f^{-1}(y)$ given outputs y. Thus, there is a contrast between the easiness of computing f and the hardness of inverting it. The adversary trying to find an inverse for y may have access to more computational resources (here we focus on time) than the honest party that computed $f(x) = y$. In contrast, quasi-one-way functions only require that finding a preimage is hard for adversaries running in some fixed polynomial time, and, furthermore, computing the function may take more time than those adversaries are allowed. In the cryptographic context, this weakening is fundamental, and one may fairly ask if it trivializes the definition, stripping one-way functions of any interesting property. Our next requirement addresses this concern: the weakened adversary still has the ability to verify input/output pairs to the function. The reason for this is explained after the definition.

Definition 1 (Quasi-one-way function)**.** *Fix a polynomial $t_V(n)$ and let $t(n) > t_V(n)$ be any polynomial. A (randomized) polynomial-time computable function $f\colon \{0,1\}^n \to \{0,1\}^n$ is an* infinitely often quasi-one-way function *against (probabilistic) time $t(n)$ with verifier V running in time t_V if for every x,*

$$V(x, f(x)) = 1,$$

but for every algorithm A that runs in time $t(n)$ on inputs of length n,

$$\Pr_x[V(A(x, f(x)), f(x)) = 1] \leq 1/t(n)$$

(for infinitely many n).

Here, both the function f or the adversary A can be either deterministic or randomized. In this work, we mainly consider randomized constructions against randomized adversaries.

Deterministic versus randomized quasi-OWFs

In the standard cryptographic setting, the existence of one-way functions is equivalent to the existence of randomized one-way functions (both against deterministic and against randomized adversaries): if a randomized one-way function exists, it can be made deterministic by making its randomness part of its input, which makes the job of the adversary only more difficult. However, for quasi-one-way functions this may not be possible, since the amount of randomness used by f may exceed the running time of the adversary, in which case it is unfair to ask the adversary to recover this randomness (in particular, making the input of f too long for the verifier V).

We remark, however, that under the derandomization notion introduced by Dubrov and Ishai [14] randomized and deterministic quasi-one-way functions are equivalent. Dubrov and Ishai prove that their notion can be realized assuming the existence of a hard average-case incompressible function in exponential time.[1]

The role of the verifier

To relate the role of the verifier in our definition to cryptographic aims, we refer to an analogy outlined in [4], in which the bitter Professor Grouse hopes to humiliate Gauss by inventing problems Gauss cannot solve. In Minicrypt, where classical one-way functions exist, such a task is possible. If both Gauss and Grouse have access to the one-way function f, Professor Grouse can choose a random input x and send $y = f(x)$ to Gauss. The hardness of inverting f ensures that Gauss will be unable to solve the "problem" $f^{-1}(y)$. Furthermore, the fact that f is easy to compute allows Grouse to humiliate Gauss by presenting him with x, since Gauss may check for himself that x was indeed a pre-image of y.

[1] In fact, a weaker notion of derandomization suffices for our purposes: the definition of Dubrov and Ishai requires statistical indistinguishability between the random and pseudorandom outputs (of the candidate f), while we only need computational indistinguishability.

Now, it is not unreasonable to suppose that Professor Grouse would be willing to spend more time finding the difficult problem than Gauss would be willing to spend solving it. Then a function f satisfying our definition also accomplishes Gauss's humiliation. Professor Grouse would choose a random input x and send $y = f(x)$ (possibly taking a long time to compute y) to Gauss. The hardness of inverting f prevents Gauss from finding $x \in f^{-1}(y)$ in the amount of time he is willing to spend. Triumphant, Professor Grouse could send x to Gauss, who is willing to take the the necessary time to check that $f(x) = y$. Thus, humiliation is tied to verification, not to the ease of computing f or the relative computational powers of Gauss and Grouse (note that in the classical case, Professor Grouse wouldn't be able to find $f^{-1}(y)$ without prior knowledge of x).

Relation to cryptography

One may object that humiliation isn't a goal of cryptography. Indeed, we do not yet know if quasi-one-way functions can be applied to any cryptographic problem. But we believe their definition captures a non-trivial aspect of the easiness-hardness contrast, where easiness here refers to the verification of input/output pairs (as opposed to finding outputs from inputs). The verification requirement implies that quasi-one-way functions (for appropriate choices of t_V and t) can't exist unless P $\neq$ NP (and, it is not too hard to construct functions unconditionally without the verification requirement).

Quasi-one-way functions from NP $\not\subseteq$ BPP

We show how to construct, based on worst-case assumptions, a randomized quasi-one-way function against time $t(n)$ for any polynomial t. In fact, we construct one verifier V running in time t_V while only the function f varies with t.

Theorem 4. *If* NP $\not\subseteq$ BPP*, then there is a polynomial t_V and a verifier V running in time t_V such that for any polynomial t, there exists an i.o. randomized quasi-one-way function f_t with verifier V against probabilistic time $t(n)$.*

We remark that the verifier in our result essentially accepts if and only if its first input is a preimage of its second input (which is stronger than the requirement in Definition 1). The only exceptions are inputs of the form $(x, \mathbf{0})$, which occur when the sampling algorithm of Theorem 3 fails.

2.3 Quasi-one-way Functions and Quasi-pseudorandom Generators

We now argue that "black-box" type constructions of pseudorandom generators from one-way functions do not carry over to the setting of quasi-pseudorandom generators and quasi-one way functions. To motivate our impossibility result, we revisit the constructions of *length-doubling* pseudorandom generators from [15, 16] and [13], and notice that they both conform to the model we describe below. Since our focus is on proving lower bounds, we work with non-uniform definitions and do not impose any unnecessary computational restrictions, which only makes our lower bounds stronger.

The pseudorandomness of cryptographic generators is proven via a reduction from distinguishing to inverting. In the results mentioned above, the inverting algorithm has oracle access to the function it is inverting. For a reduction to inverting quasi-one-way functions, such an inverter will not suffice: in evaluating the function, the reduction will necessarily take more time than the hardness of inverting guarantee.

However it could still be conceivable that access to the quasi-OWF verifier V may be sufficient to extend the argument. After all, the reductions of [12] and [13] work by first coming up with a list of candidate inverses for the function f and then evaluating f to check that the correct one was found. Can the use of f be replaced by V for this purpose? We formalize the reductions of the previous works, replacing oracle access to f with access to V:

Definition 2. *A pair of oracle circuit families (G, I) (where I is polynomial-size) is a* fully black-box quasi-pseudorandom generator construction[2] *with stretch $k(n)$ if the following conditions hold:*

- *There exists a polynomial $m(n)$ such that $G^f(1^n)$ is a function from n bits to $n + k(n)$ bits, where f is a function from $m(n)$ bits to $m(n)$ bits.*
- *Whenever f is a quasi-one-way function with verifier V, for every circuit D such that*

$$\left|\Pr_{x\sim\{0,1\}^n}[D(G^f(x)) = 1] - \Pr_{y\sim\{0,1\}^{n+k}}[D(y) = 1]\right| > \varepsilon,$$

the circuit $I^{D,V}(1^n)$ inverts f with probability $\mathrm{poly}(\varepsilon/n)$ *on at least a* $\mathrm{poly}(\varepsilon/n)$ *fraction of inputs.*

We prove that fully black-box quasi-pseudorandom generator constructions do not exist. Notice that our definition only imposes computational restrictions on I and not on G. (In fact, our proof allows I unbounded access to the distinguishing oracle. We only

[2]We call these reductions "fully black-box" as they treat both the primitive in the construction and the adversary in the proof of security as black boxes. For a detailed explanation of the role of fully black-box reductions in cryptography we refer the reader to [17].

use the fact that I makes a bounded number of queries to the verification oracle.)

Theorem 5. *For $\epsilon = 1/2$ and $m(n) = \omega(\log n)$, there is no fully black-box quasi-pseudorandom generator construction of stretch $k(n) = \omega(\log n)$.*

We argue that these bounds on the parameters $m(n)$ and $k(n)$ are the best possible. For $m(n) = O(\log n)$, f can be inverted with inverse-polynomial probability simply by guessing a random inverse. For $k(n) = O(\log n)$, Blum-Micali and Yao [8, 9] give a fully black-box construction that matches our definition from any one-way *permutation* f. While we do not know if the same stretch can be obtained when f is an arbitrary one-way function, our impossibility result also applies to the setting where f is only assumed to be a one-way permutation.

We remark that Theorem 5 is incomparable to a result of Gennaro and Trevisan [18], which lower bounds the number of queries a pseudorandom generator must make to the one-way function, but imposes no restriction on the inverting algorithm.

2.4 Our Techniques

Sampling algorithms

As mentioned in the introduction, we start with the sampling procedure from [5], which is given a canonical search[3] algorithm A and obtains a distribution that is hard for A.

Roughly speaking, this is achieved through diagonalization: the algorithm A is run on a formula which describes the success of A on smaller input lengths. In short, A is used to find the instances on which it fails. We show that in fact A may be used to find not only the instances on which it fails, but also the satisfying assignments to these instances. This seems counterintuitive; how can A find the solutions to problems on which it will fails? As in [5], the key lies in a diagonalization and running A on formulas which are larger than the instances on which it fails.

More specifically, the sampler in [5] runs A on a formula Φ_n equivalent to the NP statement:

> "There exists a formula w_n of size n such that $A(w_n) = 0$ but SAT(w_n)=1."

Whenever A returns a witness for Φ_n, we can extract a formula w_n and assignment a_n such that $A(w_n) = 0$ but a_n satisfies w_n. But A may not return a witness for Φ_n, either because A makes a mistake or because Φ_n is unsatisfiable. In the first case, we can hope to use Φ_n as a hard instance for A (note that Φ_n is a formula of size $|\Phi_n|$ on which A fails) and obtain a witness for it recursively among the formulas $\Phi_{n'}$ for $n' < n$. But what if Φ_n is unsatisfiable? Since P $\neq$ NP, we know that there are infinitely many satisfiable Φ_n; yet there is no reason to expect that A will succeed on them. Our solution is to modify Φ_n so that when A fails on an instance of size n, the formulae $\Phi_{n'}$ for all $n' > n$ will be satisfiable until either A succeeds in finding a witness, or n' is large enough to construct the witness recursively among the smaller formulae. The details of our sampler and the analysis are given in Section 3.2.

The same principle underlies our sampler for randomized algorithms. Now, the randomized search algorithm A is run on a formula $\Phi_{n,r}$ parameterized by both the input length and the randomness used by A. (The precise definition of $\Phi_{n,r}$ is given in Section 3.2.) We have that when A fails on an instance of size n, the formula $\Phi_{n,r}$ is satisfiable for most choices of r. As in the deterministic case, the we design the formula $\Phi_{n',r}$ to be satisfiable (for most choices of r) for all $n' > n$ until A succeeds in finding a witness or n' is large enough. New challenges arise because there is nothing to prevent the density of satisfiable $\Phi_{n',r'}$ from degrading as n' increases. We show how to overcome this obstacle and present the sampler for randomized algorithms in Section 3.2.

Finally, we use this sampler to build a "quasi-hard" sampler through a simple simulation argument similar in spirit to the one in Section 5 of [5].

Constructing a quasi-one-way function

The quasi-one-way function uses formula/witness pairs that can be obtained from the quasi-hard sampler in the natural way: it outputs the formula, together with some extra information. The information gives an easy way to find the witness given an inverse to the output.

Ruling out quasi-PRGs from quasi-OWFs

The intuition behind the lower bound of Theorem 5 is that in the quasi-cryptographic setting, where the inverter has no access to f, she can only gain very limited information about f by interacting with the distinguishing oracle. In fact, we design the distinguishing oracle in such a way that the inverter will not "know" which is the correct quasi-one-way function f to invert (even if she has unbounded access to the oracle). To do this, we base the distinguisher D not only on the range of G^f, but also the ranges of some other functions $G^{f_1}, \ldots, G^{f_t}$. The purpose is to confuse the inverter, which then has no way of

[3] Gutfreund et al. also handle *decision* algorithms, which presents some additional challenges. However, in our setting it makes more sense to discuss search algrotihms, since we don't expect to be able to verify that formulas are unsatisfiable. Indeed, the problem of building dreambreakers is posed for search algorithms.

knowing if she is supposed to invert f or one of the functions $f_1, \ldots, f_t$. Therefore the inversion is bound to fail with high probability.

One obstacle that must be surmounted is that in addition to f, the inverter is also given access to the verifier V. Concievably, the verifier can be then used to gain some information about the actual f that was used in the construction. We rule out this possibility by showing that in a bounded interaction, the inverter is unlikely to gain much information about f from V. We defer the explanation for this and the full proof to Appendix 5.

3 Sampling Hard Instance Witness Pairs

We first show how to construct the sampler for deterministic algorithms in Section 3.2, which proves Theorem 1. Next, in Section 3.2, we extend our approach to handle randomized algorithms. Finally, we show how to construct quasi-hard distributions in Theorem 3.

3.1 Preliminaries

As in [5], we think of boolean formulae as binary strings which can be padded easily. In other words, if x is an encoding of a formula ϕ then $x \cdot 0^i$ is also an encoding of ϕ. We consider search algorithms instead of decision algorithms – it is well-known that an efficient SAT decision algorithm implies an efficient SAT search algorithm. In our proofs, we assume the search algorithm is a *canonical search algorithm*, i.e. that it checks that its output actually satisfies the input formula (so the algorithm only errs by outputting 0). This is without loss of generality, since any formula $\phi \in$ SAT that causes the modified algorithm to reject (incorrectly) also fools the original algorithm.

As described previously, for a given search algorithm A we define a formula Φ_n which is equivalent to the NP statement:

$$\exists w_N \in \{0,1\}^N : SAT(w_N) = 1 \text{ and } A(w_N) = 0 \text{ and } n^{1/k} < N \leq n.$$

By the Cook-Levin theorem, we may assume for every n that Φ_n is a formula of size $q(n)$ for some polynomial $q(\cdot)$ which depends on A, and that Φ_n can be constructed in polynomial time from A. We choose k so that for large enough x, it holds that $q(x) < (x-1)^k$.

For randomized algorithms, we work with a new formula $\Phi_{n,r}$ which is parameterized by not just the input length, but also the randomness used by the algorithm:

$$\exists w_N \in \{0,1\}^N : SAT(w_N) = 1 \text{ and } A'(w_N, r) = 0 \text{ and } n^{1/k} < N \leq n.$$

Here A is a randomized search algorithm (as before, we assume in the proof that A only errs by answering unsatisfiable), and $A'(w_N, r)$ denotes the result of running $2N$ many trials of A and outputting the first satisfying assignment to w_N found by A (and zero otherwise) using randomness r (which may be truncated as N varies). Note that if A uses randomness of length n^b then A' uses randomness of length $2n^{b+1}$, and that almost all choices of randomness for A' reflect the failure probability of A. More precisely, with probability at least $1 - (2/3)^n$ over the choice of r, for every x it holds that $A'(x, r) = 0$ implies $Pr[A(x) = 0] > 2/3$.

3.2 The Sampling Procedure for Deterministic Algorithms

Let A be a candidate polynomial-time search algorithm for SAT (see the previous section for the assumptions we make about A and the definition of Φ_n). Assuming P $\neq$ NP, A must fail to find a satisfying assignment for infinitely many satisfiable formulas. Our sampler is designed to handle two cases:

1. Either A finds assignments to Φ_n infinitely often, or
2. for every large enough n, $A(\Phi_n) = 0$ (either because $\Phi_n \notin SAT$ or A makes a mistake).

It is clear how to handle the first case: a witness for Φ_n gives a formula and a satisfying assignment that A errs on. We give a sampler (to be used as a subroutine by the overall sampler) to handle the second case in the following lemma:

Lemma 1. *Assume* P $\neq$ NP*, and let A be a polynomial-time search algorithm running in time $t(n)$. Suppose there exists n_0 such that for every $n > n_0$, either Φ_n is unsatisfiable or $A(\Phi_n) = 0$. Then there is an algorithm D^1 (taking inputs 1^n, $t(n)$, and A) and some n_1, such that for every $n > n_1$, $D^1(1^n, A, t(n))$ (outlined below) returns a formula w of length N for $n^{1/k} < N \leq n$ and a witness α that satisfies w, but $A(w) = 0$.*

Proof. Let n' be the smallest input size larger than n_0 such that $\Phi_{n'} \in SAT$ (such n' exists if P $\neq$ NP). We wll give an algorithm D^1 which successfully constructs a witness for Φ_n for any $n > 2^{n'^k}$. First, we claim that for any n'' in $\{n', \ldots, n\}$, the formula $\Phi_{n''}$ is satisfiable. By assumption $\Phi_{n'}, \ldots, \Phi_{(n'-1)^k}$ are satisfiable, so the claim holds for $n' \leq n'' \leq (n'-1)^k$. Otherwise, consider i successive applications of q until

$q^i(n') \leq n'' < q^{i+1}(n')$. We can reason inductively that $\Phi_{q^{i-1}(n')}$ is satisfiable, so by assumption A outputs "0" on $\Phi_{q^{i-1}(n')}$ which is of length $q^i(n')$. But then $\Phi_{q^i(n_0)} \dots \Phi_{(q^i(n_0)-1)^k}$ are all satisfiable, and the choice of k gives $q^i(n') \leq n'' \leq (q^i(n'')-1)^k$ so $\Phi_{n''}$ must be satisfiable as well.

Now a witness for Φ_n should be a formula of size $n^{1/k} < N \leq n$ together with a satisfying assignment. Our choice of k gives the existence of some N so that $q^{-1}(N)$ is an integer. Since $q^{-1}(N) > n'$, we know that $\Phi_{q^{-1}(N)}$ is satisfiable and that $A(\Phi_{q^{-1}(N)}) = 0$. Thus we may use $\Phi_{q^{-1}(N)}$ as a witness for Φ_n if we find a satisfying assignment for $\Phi_{q^{-1}(N)}$. Using the same reasoning, we continue to construct a witness using formulae of decreasing size, until we are left with a formula $\Phi_{n''}$ for $n'' \leq n'^k$ that we may satisfy by exhaustive search.

We describe the algorithm D^1 precisely below. Set $i := n$ and $w := 0$:

1. If $i \leq n'^k$, find a witness for Φ_i by exhaustive search, append to w and stop. Otherwise,
2. run $A(\Phi_i)$. If A returns a witness, output fail.
3. Find an integer j such that $i^{1/k} < q(j) \leq i$.
4. Append Φ_j to w, set $i := j$, and repeat.

Note that the algorithm will not output fail if it is called with any $i > n'$. We also have that in Step 3 a suitable j is always available since $i^k > q(i+1)$. Finally, it is easy to verify that the algorithm runs in time polynomial in n when $n > 2^{n'^k}$.

We may now describe the sampling algorithm D on input 1^n:

1. For $i = n \dots n^k$, run A on Φ_i and output (w, a) returned by A if $|w| = n$.
2. If no such pair is returned, run $D^1(1^i, t(i), A)$ for $i = n, \dots, n^k$ and output the first (w, α) where w is of length n.

Using Lemma 1, it is easy to prove that the sampling procedure D must output hard instance/witness pairs for A infinitely often, which proves Theorem 1.

Proof of Theorem 1. Suppose A finds assignments for Φ_n infinitely often. For each such n, the length of the assignment N may be between $n^{1/k}$ and n. Then Step 1 ensures that the sampler D will succeed on input length N.

On the other hand, assume that the condition of Lemma 1 holds. Then for large enough n, we have that D^1 returns a pair (w, α) where $|w| = N$. The same argument as the previous case gives that D will succeed on input length N.

We now show how to sample hard instance/witness pairs for any randomized search algorithm A by considering two cases. Let A' be the corresponding algorithm described in Section 3.1.

- For inifnitely many n, the a random formula $\Phi_{n,r}$ is likely to be satisfiable and A' will likely find a witness for it.
- There is some n_0 so that for all $n > n_0$, either $\Phi_{n,r}$ is unlikely to be satisfiable or A' will likely make a mistake.

If the first case holds, we can simply find formula/witness pairs by running A'. Our main task is to show how to find witnesses when the second case holds. Because the failure of A' may be because $\Phi_{n,r}$ is mostly unsatisfiable, the task of constructing witnesses for this case is more difficult than for the deterministic case. We give the sampler for this case in Lemma 2 below.

Lemma 2. *Assume* NP $\not\subseteq$ BPP*, and suppose that there exists an n_0 such that for every $n > n_0$,*

$$\Pr_{r,r'}[A'(\Phi_{n,r}, r') = 0 \vee \Phi_{n,r} \notin SAT] > 1 - \frac{1}{q(q(n))^2}.$$

Then for infinitely many $n > n_0$, the algorithm D^1 described below, on inputs A', 1^n, and $t(n)$ returns a randomly chosen $\Phi_{n,r}$ and satisfying assignment with probability at least $1 - o(1)$ over the choice of r and randomness of D^1.

Proof. Let $n' > n_0$ be such that there is a satisfiable $x \in \{0,1\}^{n'}$ but $\Pr[A'(x, r) = 0] > 1 - 1/n'^2$. Since NP $\not\subseteq$ BPP, this occurs infinitely often. We describe the algorithm D^1 below, and show that it returns a satisfying assingment to a random $\Phi_{n,r}$ for $n = \text{poly}(2^{n'^k})$ with high probability. Our algorithm works as in the deterministic case by stringing together smaller and smaller Φ_{i,r_i} until it may find a witness by exhaustive search. Of course, each successive Φ_{i,r_i} must be satisfiable or the entire process fails. The main difficulty comes in the analysis, where we need to to show that all the Φ_{i,r_i} are satisfiable with high (actually increasing) probability over the choices of r and r_i (Claim 1).

We now describe the algorithm D^1 more precisely below. Suppose A' uses n^b bits of randomness. Set $i := n$ and $w := 0$ and choose a random $r_i \in \{0,1\}^{n^b}$.

1. If $i \leq n'^k$, find a witness for Φ_{i,r_i} by exhaustive search, append to w and stop. Otherwise
2. run $A'(\Phi_{i,r_i})$ and output fail if A' returns a witness.
3. Find an integer j such that $i^{1/k} < q(j) \leq i$ and pick a random $r_j \in \{0,1\}^{j^b}$.

4. Append Φ_{j,r_j} to w, set $i := j$, $r_i := r_j$ and repeat.

The idea of the third step is that Φ_{j,r_j} is a formula of size $q(j)$ which satisfies Φ_{i,r_i} with high probability because we assume A' "fails" on most Φ_{j,r_j}. Of course A' may fail because a significant fraction of the Φ_{j,r_j} were unsatisfiable, in which case Φ_{j,r_j} is not a useful witness and D^1 will fail. So we must ensure that the density of satisfiable Φ_{j,r_j} is maintained as j increases. It is plausible that this density decreases; as we've observed, when a fraction of the Φ_{i,r_i} are unsatisfiable, this affects the number of witnesses for Φ_{j,r_j}, which may decrease our bound for the satisfiable instances of Φ_{j,r_j} (which in turn affects the number of witnesses for $\Phi_{q(j),r_{q(j)}}\dots$). We assert that the density is in fact increasing:

Claim 1. *Let $n' \leq i$. Then* $\Pr[\Phi_{i,r_i} \notin SAT] < \frac{2}{i^2}$.

First, observe that we may easily prove Lemma 2 from Claim 1. Note that any consecutive $j < i$ in the sequence of integers chosen by D^1 satisfy $q(j) \leq i$. Thus there are at most $O(\log n)$ random formulas chosen. On the other hand, by Claim 1 any formula chosen fails to be satisfiable with probability at most $2/(\log n)^2$; Thus each chosen Φ_{j,r_j} will satisfy Φ_{i,r_i} except with probability $2/(\log n)^2 + 1/q(q(\log n))^2 = O(1/(\log n)^2)$. A union bound gives that the probability of choosing some unsatisfiable formula is at most $O(1/\log n)$. We proceed to prove Claim 1 which completes the proof of the lemma.

Proof of Claim 1. By assumption, we have that $\Pr[\Phi_{n',r'} \notin SAT] < 1/n'^2$. We need to use our assumption that A' fails to find a witness for Φ_{i,r_i} (either because $\Phi_{i,r_i} \notin SAT$ or because A' outputs 0) to show that $\Phi_{q(i),r_{q(i)}}$ will be mostly satisfiable. The key observation is that if there are not many Φ_{i,r_i} that can be witnesses, they must be witnesses to many $\Phi_{q(i),r_{q(i)}}$. Suppose that for any i we have

- $\Pr_{r,r'}[A'(\Phi_{i,r}, r') = 0 \vee \Phi_{i,r} \notin SAT] > 1 - \frac{1}{h(i)}$, and that
- $\Pr[\Phi_{i,r} \in SAT] > 1 - 1/\ell(i)$.

Then we know that

$$\Pr_{r,r'}[\Phi_{i,r} \in SAT \wedge A'(\Phi_{i,r}, r') = 0] > 1 - \frac{1}{h(i)} - \frac{1}{\ell(i)}.$$

Let S be the set of r such that $\Phi_{i,r} \in SAT$. Then we also have that

$$\Pr_{r,r'}[\Phi_{i,r} \in SAT \wedge A'(\Phi_{i,r}, r') = 0] \leq \Pr_{r}[\Phi_{i,r} \in SAT] \max_{r \in S}\{\Pr[A'(\Phi_{i,r}, r') = 0]\}.$$

Combining the two inequalities, we have that

$$\begin{aligned}\max_{r \in S} \Pr[A'(\Phi_{i,r}, r') = 0] &\geq \frac{1 - \frac{1}{h(i)} - 1/\ell(i)}{1 - 1/\ell(i)} \\ &= 1 - \frac{1}{h(i)(1 - 1/\ell(i))}.\end{aligned}$$

Now, observe that $\Phi_{i,r}$ has length $q(i)$ and that there is some $r \in S$ so that

$$\Pr[A'(\Phi_{i,r}, r') = 0] \geq 1 - \frac{1}{h(i)(1 - 1/\ell(i))}.$$

This implies that for any $j \in \{q(i), \dots, q^2(i)\}$

$$\Pr[\Phi_{j,r'} \in SAT] > 1 - \frac{1}{h(i)(1 - 1/\ell(i))}.$$

Claim 1 follows by letting $h(i) = q(q(i))^2$ and noting that $1 - \ell(i)$ is always greater than $1/2$.

Having proven Lemma 2, we may now give a proof of Theorem 2

Proof of Theorem 2. We describe the behavior of the sampler D on inputs 1^n, A and $t(n)$:

- For $i = n, \dots, n^k$, run A' on $2nq(q(i))^2$ randomly chosen $\Phi_{i,r}$. If there is some (w, a) returned of size n output it.
- If no such pair is returned, run $D^1(1^i, A', t(i))$ for $i = n, \dots, n^k$. If D^1 outputs some $\Phi_{i,r}$ and satisfying assingment (w, α) with $|w| = n$, output (w, a).

The proof follows the one for deterministic samplers. If A' finds a witness to $\Phi_{n,r}$ with probability greater than $1/q(q(n))^2$ for infinitely many n (i.e. the condition of Lemma 2 doesn't hold), the first step amplifies the success of A' so that D will output a formula and witness pair with probability at least $1 - o(1)$ for infinitely many n.

Otherwise, Lemma 2 tells us that for infinitely many n, with probability at least $1 - o(1)$, D^1 will output some pair (w, α) such that $|w| = n$ and (w, α) satisifes a randomly chosen $\Phi_{i,r}$ for some $i \in \{n, \dots, n^k\}$, i.e. $A'(w, r) = 0$ but α satisfies w. By the construction of A' (see Section 3.1), this implies that $\Pr[A(w) = 0] > 2/3$ (except with negligible probability over the choice of r).

3.3 Quasi-hard Samplers

Using a diagonalization argument, we can use the hard instance generators describe above to obtain a single generator that produces instances hard for all algorithms running in some fixed time $t(n)$.

Proof of Theorem 3. We prove the deterministic version of the theorem. The randomized version follows by an analogous argument, using standard randomness amplification (specifically $n\log(t(n))$ independent repetitions) to boost the probabilities.

Consider any standard enumeration of Turing machines $M_1, \cdots$. Let N_i be the machine which on input ϕ simulates all the machines $M_1, \ldots, M_i$ on ϕ for $t(n)$ steps, outputting the first non-zero assignment satisfying ϕ if one exists and '0' otherwise. The sampler S on input 1^n runs $D(1^n, N_i, t(n))$ for $i = 1, \ldots, n$ and outputs the last response (ϕ_i, w) so that $N_i(\phi_i) = 0$.

Now consider any machine running in time $t(n)$ with description M_j. By Lemma 1, the output of $D(1^n, N_j, t(n))$ will succeed infinitely often, and the output of $D(1^n, N_i, t(n))$ for any $i \geq j$ will be a hard instance for M_j.

4 Constructing a quasi-one-way function

The existence of a quasi-sampler suggests a natural construction of a quasi-one-way function: Run the sampler. If one obtains a pair (ϕ, w) output $(\phi, x \oplus w)$, otherwise output "0". The verifier $V(x, (\phi, r))$ outputs 1 whenever $x \oplus r$ satisfies ϕ or the second argument is "0".

Proof of Theorem 4. Fix any polynomial $t(n)$. Let $t'(n) = 2t(n)(t(n) + t_V(n))$ and let S be the sampler for time $4(t(n))^2$. Let $f : \{0,1\}^n \to \{0,1\}^{2n}$ be the randomized function:

$$f(r_w) = (\phi, r_w \oplus w).$$

where (ϕ, w) is the output of $S(1^n)$. The verifier $V(r_w, (\phi, r))$ accepts if its second argument is zero or if ϕ is satisfied by $r_w \oplus r$.

We describe how, given an algorithm A which runs in time $t(n)$ and causes V to accept with probability $1/t(n)$ for almost all n, one may construct an algorithm A' which runs in time $t'(n)$ and solves SAT with high probability under the output distribution of $S(1^n)$:

> **Algorithm A'.** On input ϕ, repeat the following $4t(n)$ times. Choose a random r and run $A(\phi, r)$ to obtain a candidate inverse r_w. If $V(r_w, (\phi, r \oplus r_w))$ accepts for any r, output $r \oplus r_w$, otherwise reject.

If ϕ is distributed according to $S(1^n)$, then (ϕ, r) follows the output distribution of the one-way function, so by assumption the inversion algorithm will succeed with probability $1/t(n)$. Therefore for at least $1/2t(n)$ fraction of the inputs ϕ, the inversion algorithm succeeds for a $1/2t(n)$ fraction of rs. By repetition, for such a ϕ, the corresponding r will be found with probability $1/2$, in which case $r \oplus r_w$ is a witness for ϕ. Therefore A' produces a witness for ϕ with probability $1/4t(n)$, contradicting Theorem 3.

5 Quasi-one-way Functions versus Quasi-pseudorandom Generators

In this section we give a proof of Theorem 5. We prove the theorem by contradiction. In particular, we analyze what happens when we instantiate the construction with a random permutation $f : \{0,1\}^{m(n)} \to \{0,1\}^{m(n)}$. Such a random permutation is one-way with high probability. (While it is not crucial that we use a permutation, it simplifies the analysis somewhat.)

The idea of the proof is that the distinguisher can be chosen in such a way so that the inverter will not "know" which f was used, so she will be unlikely to produce the correct inverse for information-theoretic reasons. One complication is that the inverter also has access to a verifier, from which some information about f may be gained. We will argue that this verification oracle is unlikely to reveal any useful information, so that with little effect on the analysis it can be replaced by an oracle that always answers zero.

Specifically, for any permutation f, we define the following distinguisher D_f and verifier V_f. In our definition, the distinguisher is randomized, but in the end the randomness can be fixed to minimize the distinguishing probability of the inverter.

In what follows $t = t(n)$ is a parameter that is chosen to be super-polynomial in n, but such that $t < 2^{m/2}$ and $t < 2^k/2$. By our assumptions on m and k such a choice is always possible.

Distinguisher $D_f(y)$: Choose $r \in \{1, \ldots, t\}$ uniformly at random. Define $f_i : \{0,1\}^m \to \{0,1\}^m$ to be f if $i = r$, and a uniformly random permutation otherwise. Output 1 if y is in the range of some G^{f_i}, $1 \leq i \leq t$, and 0 otherwise.

Verifier $V_f(x, y)$: Output 1 if $f(x) = y$, and 0 otherwise.

Notice that $D(G^f(x)) = 1$ for every x, but the number of ys for which $D(y) = 1$ is at most $t \cdot 2^n$. As long as $t < 2^k/2$, D distinguishes the output of G^f from the uniform distribution with probability at least $1/2$.

We begin by proving that when t is sufficiently large, the queries made by the inverter to the verifier are always answered by 0.

Claim 2. *Assume $12q \leq t \leq 2^{m/2}$. Fix any $z \in \{0,1\}^m$. The probability that $I^{D_f, V_f}(z)$ ever makes a*

query of the form $(x, f(x))$ to V_f is at most $2^{-m} + 12q/t$, where q is the number of queries that I makes to the V_f oracle.

The intuition behind the claim is this. Suppose we give the inverter a complete description of the truth-tables of the functions $f_1, \ldots, f_t$. In particular this contains all the information provided by the oracle D_f. Now think of the verifier as playing a game where her goal is to make the query $(x, f_r(x))$ for some x. The problem is that she does not know what r is, and she can only gain information about r by making other queries of the same type. What is her best strategy? Intuitively, she should be looking to make queries (x, y) where $y = f_i(x)$ for many is simultaneously. If one of these i's equals r she wins, and otherwise she can rule out several possible values of r. However, since the permutations are random, it is unlikely that there exists a point (x, y) where the graphs of many of them intersect.

Proof. Let us begin by giving an upper bound on the quantity

$$M = \max_{(x,y)} |\{i : y = f_i(x)\}|$$

when $f_1, \ldots, f_t$ are random permutations. For any point (x, y) and any subcollection of 6 of the permutations, the probability that all of them hit (x, y) is at most 2^{-6m}, so by a union bound we have that

$$\Pr[M \geq 6] \leq 2^{2m} \cdot \binom{t}{6} \cdot 2^{-6m} > 2^{-m}$$

using the assumption $t \leq 2^{m/2}$.

Now let $(x_1, y_1), \ldots, (x_q, y_q)$ denote the queries made by the inverter on input z. The probability that the inverter manages to make a query of the type $(x, f_r(x))$ can be upper bounded by

$$\begin{aligned}\Pr[M \geq 6] + \Pr[y_1 = f_r(x_1) \mid M < 6] + \\ \Pr[y_2 = f_r(x_2) \mid y_1 \neq f_r(x_1) \wedge (M < 6)] \\ + \cdots + \\ \Pr[y_q = f_r(x_q) \mid y_1 \neq f_r(x_1) \wedge \ldots \\ \wedge\, y_{q-1} \neq f_r(x_{q-1}) \wedge (M < 6)]\end{aligned}$$

so it remains to upper bound the quantity

$$\begin{aligned}\Pr[y_i = f_r(x_i) \mid y_1 \neq f_r(x_1) \wedge \ldots \\ \wedge\, y_{i-1} \neq f_r(x_{i-1}) \wedge (M < 6)]\end{aligned}$$

for $1 \leq i < q$. To do this, fix any collection of f_is that satisfy the condition $M < 6$, so now the probability is only taken over a random choice of r. By the bound on M, the number of f_j such that $y_i = f_j(x_i)$ for some i is at most $6i \leq 6q$, so r is uniformly distributed over a set of size at least $t - 6q$. It follows that the above probability is upper bounded by $6/(t - 6q)$, so the probability that $I^{D_f, V_f}(z)$ ever makes a positive query to V_f is at most $2^{-m} + 6q/(t - 6q) \leq 2^{-m} + 12q/t$.

We now prove the main theorem of this section.

Proof of Theorem 5. Let p denote the probability that the inverter succeeds on a random y, namely

$$p = \Pr[I^{D_f, V_f}(y) = f^{-1}(y)]$$

where the probability is taken over y, f, and the randomness of the distinguisher. Let 0 be the oracle that always outputs zero. By the inclusion-exclusion principle, we have

$$\begin{aligned}1 \geq \Pr[I^{D_f,0}(y) = f_i^{-1}(y) \text{ for some } i] \geq \\ \sum_{1 \leq i \leq t} \Pr[I^{D_f,0}(y) = f_i^{-1}(y)] - \\ \sum_{1 \leq i < j \leq t} \Pr[I^{D_f,0}(y) = f_i^{-1}(y) \\ \text{and } I^{D_f,0}(y) = f_j^{-1}(y)].\end{aligned}$$

We now lower bound every term of the first type, and upper bound every term of the second type. For terms of the first type, by symmetry of the f_is, we have that

$$\begin{aligned}\Pr[I^{D_f,0}(y) = f_i^{-1}(y)] &= \\ \Pr[I^{D_{f_i},0}(y) = f_i^{-1}(y)] &= \\ \Pr[I^{D_f,0}(y) = f^{-1}(y)].\end{aligned}$$

By Claim 2,

$$\begin{aligned}\Pr[I^{D_f,0}(y) = f^{-1}(y)] \geq \Pr[I^{D_f,V_f}(y) = f^{-1}(y)] \\ - \Pr[V_f \text{ is queried } (x, f(x))] \\ \geq p - (2^{-m} + 12q/t).\end{aligned}$$

For terms of the second type,

$$\begin{aligned}\Pr[I^{D_f,0}(y) = f_i^{-1}(y) \text{ and } I^{D_f,0}(y) = f_j^{-1}(y)] \leq \\ \Pr[f_i^{-1}(y) = f_j^{-1}(y)] = 2^{-m}\end{aligned}$$

so it follows that

$$1 \geq t \cdot (p - (2^{-m} + 12q/t)) - \binom{t}{2} \cdot 2^{-m}$$

From $t = 2^{m/2}$, it follows that $p \leq (12q+3)/t + 2^{-m} = n^{-\omega(1)}$. Since a random permutation f is one-way with overwhelming probability, it follows that there exists a one-way f such that

$$\Pr_{y \sim \{0,1\}^m}[I^{D_f, V_f}(y) = f^{-1}(y)] = n^{-\omega(1)}$$

contradicting the correctness requirement of I.

6 Conclusions

Building on [5], we were able to construct "dream-breakers." Of course, constructing one-way functions that rely on worst-case hardness assumptions such as P $\neq$ NP remains a distant goal. We suggest that the study of quasi-one-way functions and (corresponding notions of other primitives) may enhance our understanding of the relationship between worst-case complexity, average-case complexity and the cryptographic task.

For example, we find it instructive that some methods we take for granted in the standard setting — e.g. that the hardness of one-way functions can be amplified, or that one-way functions and pseudorandom generators are "computationally equivalent" — do not carry over to this new setting. Yet quasi-cryptography offers other tools — like simulations and "non-black-box" methods such as the ones in [5] — that might compensate for this difficulty. It would be interesting to see how much of this can be achieved.

Part of our motivation was to investigate alternatives to [11] in constructing pseudorandom generators under uniform hardness assumptions. Here, we fall short in two aspects: First, our construction of "quasi-one-way functions" is randomized; and second, we find obstacles to turning quasi-one-way functions into pseudorandom generators. Towards bypassing the first obstacle, we observed that randomized quasi-one-way functions can be derandomized by using assumptions considered before (see [14]). Therefore, while complete derandomization (assuming the existence of hard problems in NP) may be out of reach for current techniques, an intermediate goal may be to show that the existence of pseudorandom generators for HeurBPP follows from a hardness assumption plus some generic derandomization assumption.

Acknowledgements

The authors would like to thank Andy Yao for hosting the visit at Tsinghua University where this work was initiated. They would also like to thank Danny Gutfreund and Hoeteck Wee for helpful discussions.

References

[1] J. Feigenbaum and L. Fortnow. Random-self-reducibility of complete sets. *SIAM J COMPUT.*, 22(5):994–1005, 1993.

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

[3] A. Akavia, O. Goldreich, S. Goldwasser, and D. Moshkovitz. On basing one-way functions on NP-hardness. *Proceedings of the thirty-eighth annual ACM symposium on Theory of computing*, pages 701–710, 2006.

[4] Russell Impagliazzo. A personal view of average-case complexity. In *Structure in Complexity Theory Conference 1995*, pages 134–147, 1995.

[5] D. Gutfreund, R. Shaltiel, and A. Ta-Shma. If NP languages are hard on the worst-case then it is easy to find their hard instances. *Computational Complexity, 2005. Proceedings. Twentieth Annual IEEE Conference on*, pages 243–257, 2005.

[6] Albert Atserias. Distinguishing sat from polynomial-size circuits, through black-box queries. In *IEEE Conference on Computational Complexity*, pages 88–95, 2006.

[7] D. Gutfreund and A. Ta-Shma. Worst-Case to Average-Case Reductions Revisited. *LECTURE NOTES IN COMPUTER SCIENCE*, 4627:569, 2007.

[8] A.C. Yao. Theory and application of trapdoor functions. pages 80–91, 1982.

[9] M. Blum and S. Micali. How to Generate Cryptographically Strong Sequences of Pseudorandom Bits. *SIAM Journal on Computing*, 13:850, 1984.

[10] N. Nisan and A. Wigderson. Hardness vs randomness. 49:149–167, 1994.

[11] Russell Impagliazzo and Avi Wigderson. Randomness vs. time: de-randomization under a uniform assumption. In *Proceedings of the 39th FOCS*, pages 734–743, 1998.

[12] Oded Goldreich and Leonid Levin. A hard-core predicate for all one-way functions. In *Proceedings of the Twenty First Annual ACM Symposium*, pages 25–32, 1989.

[13] J. Hastad, R. Impagliazzo, L.A. Levin, and M. Luby. Pseudorandom generator from any one-way function. *SIAM J COMPUT*, 28(4):1364–1396, 1999.

[14] Bella Dubrov and Yuval Ishai. On the randomness complexity of efficient sampling. In *STOC '06: Proceedings of the thirty-eighth annual ACM symposium on Theory of computing*, pages 711–720, New York, NY, USA, 2006. ACM. ISBN 1-59593-134-1. doi: http://doi.acm.org/10.1145/1132516.1132615.

[15] O. Goldreich, H. Krawczyk, and M. Luby. On the existence of pseudorandom generators. In *Foundations of Computer Science, 1988., 29th Annual Symposium on*, pages 12–24, 1988.

[16] L.A. Levin. One way functions and pseudorandom generators. *Combinatorica*, 7(4):357–363, 1987.

[17] O. Reingold, L. Trevisan, and S. Vadhan. Notions of reducibility between cryptographic primitives. *Lecture Notes in Computer Science*, 2951:1–20, 2004.

[18] R. Gennaro and L. Trevisan. Lower bounds on the efficiency of generic cryptographic constructions. In *Proceedings of the 41st Annual Symposium on Foundations of Computer Science*. IEEE Computer Society Washington, DC, USA, 2000.