Electronic Colloquium on Computational Complexity, Report No. 199 (2016)

The Journey from NP to TFNP Hardness

Pavel Hubáček* Moni Naor* Eylon Yogev*

Abstract

The class TFNP is the search analog of NP with the additional guarantee that any instance has a solution. TFNP has attracted extensive attention due to its natural syntactic subclasses that capture the computational complexity of important search problems from algorithmic game theory, combinatorial optimization and computational topology. Thus, one of the main research objectives in the context of TFNP is to search for efficient algorithms for its subclasses, and at the same time proving hardness results where efficient algorithms cannot exist.

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

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

In this work, we answer this question and show that hard-on-average TFNP problems can be based on the weak assumption that there exists a *hard-on-average* language in NP. In particular, this includes the assumption of the existence of one-way functions. In terms of techniques, we show an interesting interplay between problems in TFNP, derandomization techniques, and zero-knowledge proofs.

*Weizmann Institute of Science, Israel. Email: {pavel.hubacek,moni.naor,eylon.yogev}@weizmann.ac.il.
Supported in part by a grant from the I-CORE Program of the Planning and Budgeting Committee, the Israel Science Foundation and BSF. Moni Naor is the incumbent of the Judith Kleeman Professorial Chair.

ISSN 1433-8092

Contents

1 Introduction

I rarely end up where I was intending to go,
but often I end up somewhere I needed to be.

Douglas Adams

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

Beyond its natural theoretical appeal, TFNP attracted extensive attention due to its important syntactic subclasses. The common way of defining such subclasses is via various non-constructive arguments used for proving totality of search problems. For example, the *parity argument for directed graphs*: "If a directed graph has an unbalanced node (a vertex with unequal in-degree and out-degree), then it must have another unbalanced node," gives rise to the class PPAD (for Polynomial Parity Argument on Directed graphs [Pap94]). This might be the most famous subclass of TFNP due to the fact that one of its complete problems is finding Nash equilibria in strategic games [DGP09, CDT09]. Other known subclasses are PPP (for Polynomial Pigeonhole Principle [Pap94]), PLS (for Polynomial Local Search [JPY88]), and CLS (for Continuous Local Search [DP11]).

It is easy to see that if $\mathsf{P} = \mathsf{NP}$ then all search problems in TFNP can be solved efficiently. Therefore, the study of the hardness of TFNP classes must rely on some hardness assumptions (until the $\mathsf{P} \overset{?}{=} \mathsf{NP}$ question is resolved). A related issue is to establish hard distributions for problems in TFNP. Here it is natural to use hardness of cryptographic assumptions, and therefore the goal is to base TFNP hardness on different cryptographic primitives. For instance, the (average-case) hardness of the subclass PPP has been based on the existence of either one-way permutations[1] [Pap94] or collision-resistant hash[2] [Jeř16]. For other subclasses even less standard assumptions have been used. The hardness of PPAD, PLS and even CLS (a subclass of their intersection) has been based on strong cryptographic assumptions, e.g., indistinguishability obfuscation and functional encryption [BPR15, GPS16, HY16].

To understand the hierarchy of cryptographic assumptions it is best to turn to Impagliazzo's worlds [Imp95]. He described five possible worlds: `Algorithmica` (where $\mathsf{P} = \mathsf{NP}$), `Heuristica` (where NP is hard in the worst case but easy on average, i.e., one simply does not encounter hard problems in NP), `Pessiland` (where hard-on-average problems in NP exist, but one-way

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

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

functions do not exist), `Minicrypt` (where one-way functions exist[3]), and `Cryptomania` (where Oblivious Transfer exists[4]). Nowadays, it is possible to add a sixth world, `Obfustopia` where indistinguishability obfuscation for all of P is possible [BGI+01, GGH+13, SW14]. Our goal is to connect these worlds to the world of TFNP.

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

Barriers for proving TFNP hardness. Given the lack of success in establishing hardness of TFNP under general assumptions, there has been some effort in trying to explain this phenomenon. From the early papers on PLS and TFNP by Johnson et al. [JPY88] and Megiddo and Papadimitiriou [MP91] we know that TFNP hardness cannot be based on worst-case NP hardness, unless $\mathsf{NP} = \mathsf{coNP}$[5]. Mahmoody and Xiao [MX10] showed that even a *randomized* reduction from TFNP to worse-case NP would imply that SAT is "checkable" (which is a major open question [BK95]). Buhrman et al. [BFK+10] exhibited an oracle under which all TFNP problems are easy but the polynomial hierarchy is infinite, thus, explaining the failure to prove TFNP hardness based on worst-case problems in the polynomial hierarchy. In a recent work, Rosen et al. [RSS16] showed that any attempt to base TFNP hardness on (trapdoor) one-way functions (in a black-box manner) must result in a problem with exponentially many solutions. This leads us to ask the following natural question:

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

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

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

In this work, we give an (almost) tight answer to this question. Given the negative results discussed above, our results are quite unexpected. While the barriers imply that it is very unlikely to show TFNP hardness under *worst-case* NP hardness, we are able to show that *hard-on-average* TFNP can be based on any *hard-on-average* language in NP (and in particular on the existence

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

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

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

of one-way functions). In the terminology of the worlds of Impagliazzo, our results show that hard-on-average TFNP problems exist in `Pessiland` (and beyond), a world in which none of the assumptions previously used to show TFNP hardness exist (not even one-way functions). On the other hand, the barriers discussed above indicate that it would be unlikely to prove worse-case TFNP hardness in `Heuristica`, as in that world the only available assumption is $\mathsf{P} \neq \mathsf{NP}$.

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

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

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

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

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

1.1 Our results

Some of our results use a derandomization assumption in the form "assume that there exists a function with deterministic (uniform) time complexity $2^{O(n)}$, and Π_2-circuit complexity $2^{\Omega(n)}$". In the description of our results below, we simply call this the *fooling assumption*. Alternatively, instead of this assumption we can consider the non-uniform setting, and get the same results for TFNP/poly (see Definition 3.8). Some of our results are summarized in Figure 1.

Theorem 1.1 (Informal)**.** *Any hard-on-average* NP *language (e.g., random SAT, hidden clique, or any one-way function) implies a non-uniform* TFNP *problem which is hard-on-average.*

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

Corollary 1.2 (Informal)**.** *Under the fooling assumption, any hard-on-average* NP *language implies a (uniform)* TFNP *problem which is hard-on-average.*

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

Theorem 1.3 (Informal)**.** *Assume the existence of Zaps and injective one-way functions. Then, there exists a hard-on-average problem (either non-uniform, or uniform under the fooling assumption) in* TFNP *such that any instance has at most two solutions.*

1.2 Related work (search vs. decision)

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

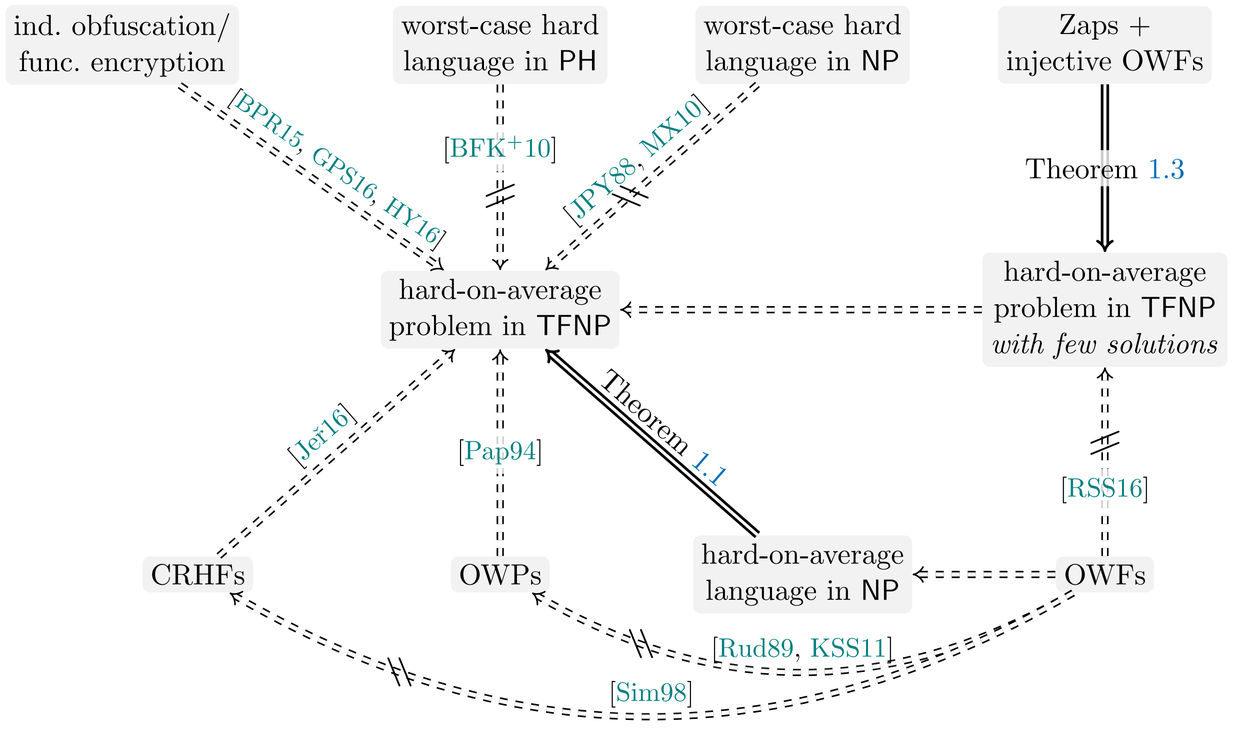
ind. obfuscation/ func. encryption
worst-case hard language in PH
worst-case hard language in NP
Zaps + injective OWFs
[BPR15, GPS16, HY16]
[BFK+10]
[JPY88, MX10]
Theorem 1.3
hard-on-average problem in TFNP
hard-on-average problem in TFNP with few solutions
[Jeř16]
[Pap94]
Theorem 1.1
[RSS16]
CRHFs
OWPs
hard-on-average language in NP
OWFs
[Rud89, KSS11]
[Sim98]

Figure 1: An illustration of our results (solid implications) in the context of previously known positive results (dashed implications) and negative results (crossed out dashed implications).

2 Our Techniques

2.1 TFNP hardness based on average-case NP hardness

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

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

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

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

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

At this point, we have an (efficiently) samplable distribution $\mathcal{D}'$ such that its support provably contains only instances in L', and finding any valid witness is still hard for any polynomial-time algorithm. However, how can we use $\mathcal{D}'$ to create a hard TFNP problem? Indeed, we can use $(L', \mathcal{D}')$ to construct a hard search problem. Since any instance in the support of $\mathcal{D}'$ is satisfiable we would claim that the problem is indeed in TFNP. However, this is actually not the case, since it is infeasible to verify that an instance has actually been sampled from $\mathcal{D}'$! Therefore, the resulting search problem is not in TFNP. To solve this, we need a way to prove that an instance x is in the support of $\mathcal{D}'$ without hurting the hardness of the distribution.

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

If the original distribution $\mathcal{D}$ was public-coin to begin with, then we show that our modifications maintain this property. Thus, assuming that $\mathcal{D}$ is public-coin we get a problem that is provably in TFNP and is as hard as $\mathcal{D}$ (see Section 4.1 for the full proof). Then, we show that in fact any hard distribution can be modified to construct a public-coin distribution while maintaining its hardness, with no additional assumptions. This lets us construct a hard TFNP problem using any hard-on-average language L, even one with a distribution that is not public-coin (see Section 4.2 for discussion of the transformation).

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

2.2 Using zero-knowledge for TFNP hardness

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

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

To solve this issue, we employ as our main tool a Zap: a two-message witness indistinguishable proof system for NP (discussed in Section 1). We construct a problem where an instance contains an image y and a Zap proof π of the statement "y has an inverse under f". The first message of the proof is either non-uniformly hardwired in the problem description, or uniformly derandomized (see discussion in the introduction). This way, any instance has a solution: if π is a valid proof then by the perfect soundness of the Zap we know that y indeed has an inverse, and otherwise, we consider the all-zero string to be a solution. Our goal is to show that this problem is still hard. However, the Zap proof only guarantees that for any two x, x' the proof π is indistinguishable, which does not suffice. In fact, if f is injective, then y has only a single inverse and π might simply be the inverse x without violating the witness indistinguishability property of the Zap.

Therefore, an instance of our problem will consist of two images y, y' and a Zap proof π that *one* of the images has an inverse under f. The goal is to find an inverse of y or y', where one is guaranteed to exist as long as the proof π is valid. This way, we are able to ensure that the proof π does not reveal any useful information about the inverse x. Finally, by randomly embedding an instance y of a challenge to f we show that any adversary that solves this problem can be used to invert the one-way function f. Thus, we get a total problem that is hard assuming one-way functions, and Zaps. Moreover, notice that when the underlying one-way function is *injective* the problem we constructed has *exactly one or two solutions* for any instance. See Theorem 5.1 and Section 5 for details of the full proof.

3 Preliminaries

We present the basic definitions and notation used in this work. For a distribution X we denote by $x \leftarrow X$ the process of sampling a value x from the distribution X. For an integer $n \in \mathbb{N}$ we denote by $[n]$ the set $\{1, \ldots, n\}$. A function $\mathsf{neg}\colon \mathbb{N} \to \mathbb{R}$ is negligible if for every constant $c > 0$ there exists an integer N_c such that $\mathsf{neg}(n) < n^{-c}$ for all $n > N_c$.

Definition 3.1 (Computational Indistinguishability). *Two sequences of random variables $X = \{X_n\}_{n\in\mathbb{N}}$ and $Y = \{Y_n\}_{n\in\mathbb{N}}$ such that X_n's and Y_n's lengths are polynomially bounded are computationally indistinguishable if for every probabilistic polynomial time algorithm A there exists an integer N such that for all $n \geq N$,*

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

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

Definition 3.2 (One-Way Functions). *A polynomial time computable function $f : \{0,1\}^* \to \{0,1\}^*$ is called* one-way *if for any PPT inverter A (the non-uniform version is polysize circuit) there exists a negligible function $\mu(\cdot)$, such that*

$$\Pr\left[A(f(x)) \in f^{-1}(f(x)) : x \leftarrow \{0,1\}^n\right] \leq \mu(n) \ .$$

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

3.1 Average-case complexity

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

Definition 3.3 (Probability distributions). *A probabilistic randomized algorithm $\mathcal{D}$ is said to be a probability distribution if on input 1^n it outputs a string of length n. We denote by $\mathcal{D}(1^n; r)$ the evaluation of $\mathcal{D}$ on input 1^n using the random coins r. We say that $\mathcal{D}$ is efficient if it runs in polynomial time.*

Definition 3.4 (Hard distributional problem). *Let $L \in \mathsf{NP}$ and let $\mathcal{D}$ be an efficient probability distribution. We say that $(L, \mathcal{D})$ is a hard distributional problem if for any probabilistic polynomial time-algorithm A there exist a negligible function $\mathsf{neg}(\cdot)$ such that for all large enough n it holds that:*

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

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

Remark 3.5. *We say that a language $L \in \mathsf{NP}$ is hard-on-average if there exists an efficient probability distribution $\mathcal{D}$ such that $(L, \mathcal{D})$ is a hard distributional problem.*

Definition 3.6 (Hard public-coin distributional problem). *Let $L \in \mathsf{NP}$ and let $\mathcal{D}$ be an efficient probability distribution. We say that $(L, \mathcal{D})$ is a hard public-coin distributional problem if for any probabilistic polynomial time-algorithm A there exists a negligible function $\mathsf{neg}(\cdot)$ such that for all large enough n it holds that:*

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

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

3.2 Total search problems

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

Definition 3.7 (TFNP)**.** *A* total NP search problem *is a relation $\mathcal{S}(x, y)$ such that it is (i) computable in polynomial (in $|x|$ and $|y|$) time (ii)* total*, i.e. there is a polynomial q such that for every x there exists a y such that $\mathcal{S}(x, y)$ and $|y| \leq q(|x|)$.*
The set of all total NP search problems is denoted by TFNP.

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

Definition 3.8 ($\mathsf{TFNP/poly}$)**.** *A* total $\mathsf{NP/poly}$ search problem *is a relation $\mathcal{S}(x, y)$ such that it is (i) computable polynomial (in $|x|$ and $|y|$) time with polynomial advice or equivalently there exists a family of polynomial sized circuits that computes $\mathcal{S}$ (ii)* total*, i.e. there is a polynomial q such that for every x there exists a y such that $\mathcal{S}(x, y)$ and $|y| \leq q(|x|)$.*
The set of all total $\mathsf{NP/poly}$ search problems is denoted by $\mathsf{TFNP/poly}$.

3.3 Witness indistinguishable proof systems

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

Definition 3.9. *Let L be an NP language with relation R. A scheme $(\mathsf{Prove}, \mathsf{Verify})$ is a witness indistinguishable proof system between a verifier and a prover if:*

1. **Completeness***: for every $(x, w) \in R$ we have that:*
$$\Pr\left[\mathsf{Verify}(x, \pi) = 1 \mid \pi \leftarrow \mathsf{Prove}(x, w)\right] = 1 \ .$$
2. **Perfect Soundness***: for every $x \notin L$ and for every π it holds that:*
$$\Pr\left[\mathsf{Verify}(x, \pi) = 1\right] = 0 \ .$$
3. **Witness Indistinguishability***: for any sequence of $\{x, w, w'\}$ such that $(x, w) \in R$ and $(x, w') \in R$ it holds that:*
$$\{\mathsf{Prove}(x, w)\} \approx_c \{\mathsf{Prove}(x, w')\} \ .$$

3.4 Nisan-Wigderson type pseudorandom generators

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

Definition 3.10 (NW-type PRGs.)**.** *A function $\mathsf{G}\colon \{0,1\}^{d(n)} \to \{0,1\}^n$ is an* NW-type PRG against circuits of size $t(n)$ *if it is (i) computable in time $2^{O(d(n))}$ and (ii) any circuit C of size at most $t(n)$ distinguishes $U \leftarrow \{0,1\}^n$ from $\mathsf{G}(s)$, where $s \leftarrow \{0,1\}^{d(n)}$, with advantage at most $1/t(n)$.*

Theorem 3.11 ([IW97]). *Assume there exists a function in $\mathsf{E} = \mathsf{DTIME}(2^{O(n)})$ with circuit complexity $2^{\Omega(n)}$. Then, for any polynomial $t(\cdot)$, there exists a NW-type generator $\mathsf{G}\colon \{0,1\}^{d(n)} \to \{0,1\}^n$ against circuits of size $t(n)$, where $d(n) = O(\log n)$.*

Note that one can find a specific function f satisfying the above condition. In general, any function that is E-complete under linear-time reductions will suffice, and in particular, one can take the bounded halting function.[6]

The above theorem was used in derandomization to fool polynomial sized circuits. It was observed in [AIKS16] that Theorem 3.11 can be extended to more powerful circuits such as non-uniform circuits. In particular, they gave the following theorem that is used in this work.

Definition 3.12 (oracle circuits and Σ_i/Π_i-circuits). *Given a boolean function $f(\cdot)$, an f-circuit is a circuit that is allowed to use f gates (in addition to the standard gates). A Σ_i-circuit (resp., Π_i-circuit) is an f-circuit where f is the canonical Σ_i^p-complete (resp., Π_i^p-complete) language. The size of all circuits is the total number of wires and gates (see [AB09, Chapter 5] for a formal definition of the classes Σ_i^p and Π_i^p).*

Theorem 3.13 (cf., [AIKS16, Theorem 1.7]). *For every $i \geq 0$, the statement of Theorem 3.11 also holds if we replace every occurrence of the word "circuits" by "Σ_i-circuits" or alternatively by "Π_i-circuits".*

The assumption underlying the above theorem is a worst-case assumption and it can be seen as a natural generalization of the assumption that $\mathsf{E} \subsetneq \mathsf{NP}$. For a further discussion about this type of assumptions see [AIKS16, AASY16].

4 TFNP Hardness from Average-Case NP Hardness

We show that there exists a search problem in TFNP that is hard-on-average under the assumption of existence of a hard-on-average NP language and a Nisan-Wigderson type complexity assumption. An overview of the proof is given in Section 2.1. Formally, we prove:

Theorem 4.1. *If there exists a hard-on-average language in NP then there exists a hard-on-average problem in non-uniform TFNP.*

Under an additional (worst-case) complexity assumption (as discussed in Section 1.1), we also give a uniform version of this result.

Corollary 4.2. *Assume that there exist functions with deterministic (uniform) time complexity $2^{O(n)}$, and Π_2-circuit complexity $2^{\Omega(n)}$. If there exists a hard-on-average language in NP then there exists a hard-on-average problem in TFNP.*

We split the proof of Theorem 4.1 and Corollary 4.2 into two parts. First, we prove the results under the assumption that the distribution $\mathcal{D}$ has a special property we call *public-coin*, i.e., when the distribution remains hard even given the random coins used to sample from it (see Definition 3.6). Second, we show that this assumption does not hurt the generality of the statement, since the existence of hard distributional decision problems implies the existence of hard *public-coin* distributional decision problems.

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

4.1 Hardness based on public-coin distributions

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

Proof (Theorem 4.1). Fix an input size n. For simplicity of presentation (and without loss of generality), assume that to sample an instance $x \in \{0,1\}^n$ the number of random coins needed is n (otherwise, one can pad the instances to get the same effect). We begin by showing that any hard distributional decision problem $(L, \mathcal{D})$ implies the existence of a hard distributional search problem. In general, this problem will not be a total one but we will be able to use its structure to devise a total problem.

Let $\mathcal{D}$ be a hard distribution for some NP language L with associated relation R_L. The distributional search problem associated to $(L, \mathcal{D})$, i.e., given $x \leftarrow \mathcal{D}(1^n)$ to find a w such that $R_L(x, w) = 1$, is also hard. Moreover, there exists a polynomial $p(\cdot)$ such that $\Pr_{x \leftarrow \mathcal{D}}[L(x) = 1] \geq p(n)$, as otherwise, the "constant no" algorithm would contradict $\mathcal{D}$ being a hard distribution for L. For any set of $k = n^2 \cdot p(n)$ strings $s_1, \ldots, s_k \in \{0,1\}^n$, we define a distributional problem $(L_{s_1,\ldots,s_k}, \mathcal{D}')$, where the language $L_{s_1,\ldots,s_k}$ is defined by the relation

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

and $\mathcal{D}'$ is the uniform distribution on $\{0,1\}^n$.

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

Lemma 4.3. *Let $(L, \mathcal{D})$ be a hard distributional search problem. Let $(L', \mathcal{D}')$ be a distributional search problem related to $(L, \mathcal{D})$ that satisfies the following conditions:*

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

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

The proof of Lemma 4.3 follows by a reduction to solving the original distributional search problem $(L, \mathcal{D})$, and is deferred to Appendix A.1. Notice that $(L_{s_1,\ldots,s_k}, \mathcal{D}')$ satisfies the three conditions of Lemma 4.3 with respect to $(L, \mathcal{D})$: (1) the instances are in the "OR" form (where $x_i = f_i(r) = r \oplus s_i$), (2) for any $i \in [k]$ it holds that x_i is distributed as $\mathcal{D}$ since $r \oplus s_i$ is uniformly random, and (3) for any $x^* = \mathcal{D}(1^n; r^*)$, sampling from $\mathcal{D}'$ conditioned on $x_i = x^*$ can be done by setting $r = s_i \oplus r^*$. We say that $L_{s_1,\ldots,s_k}$ *is good* if it is total, i.e., if it holds that

$$\forall r \in \{0,1\}^n : L_{s_1,\ldots,s_k}(r) = 1 \ .$$

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

Claim 4.4. $\Pr_{s_1,\ldots,s_k \leftarrow \{0,1\}^{kn}}[L_{s_1,\ldots,s_k}$ *is good* $] \geq 3/4$.

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

$$\Pr_{s \leftarrow \{0,1\}^n}[L(x) = 1 \mid x \leftarrow \mathcal{D}(1^n; r \oplus s)] \geq 1/p(n) \ .$$

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

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

Thus, taking a union bound over all possible $r \in \{0,1\}^n$ we get that

$$\Pr_{s_1,\ldots,s_k \leftarrow \{0,1\}^{kn}}[L_{s_1,\ldots,s_k} \text{ is not good }] \leq 2^n \cdot 2^{-n^2} \leq 1/4 \ ,$$

and thus the statement of the claim follows. □

Thus, for any n we fix a set of k strings non-uniformly to get a good $L_{s_1,\ldots,s_k}$, and get a language L^*, that is a hard-on-average search problem under the uniform distribution $\mathcal{D}^*$. Moreover, we get that the problem is total: for every n, every instance $r \in \{0,1\}^n$ must have a solution since we choose a good $L_{s_1,\ldots,s_k}$ for every n.[7] □

Getting a uniform version. In the above proof of Theorem 4.1 we constructed a problem in non-uniform TFNP, i.e., a problem in $\mathsf{TFNP/poly}$ (see Definition 3.8). To get a uniform version of the problem (i.e., a problem in TFNP) we show how to employ derandomization techniques to find $s_1, \ldots, s_k$.

Proof (Corollary 4.2). Recall the definition of the language $L_{s_1,\ldots,s_k}$ from the proof of Theorem 4.1. Consider a circuit C that on input $s_1, \ldots, s_k$ outputs 1 if and only if $L_{s_1,\ldots,s_k}$ is good. Notice that C can be implemented by a polynomial-sized Π_2-circuit, since $s_1, \ldots, s_k$ is good if for all $r \in \{0,1\}^n$ there exists a witness (s_i, w) such that $R_L(r \oplus s_i, w) = 1$. Let G be a Nisan-Wigderson type PRG (see Definition 3.10) against Π_2-circuits of size $|C|$ with seed length $m = O(\log n)$. For any seed $j \in [2^m]$, let $(s_1^j, \ldots, s_k^j) = \mathsf{G}(j)$ and write $L^j = L_{s_1^j \ldots, s_k^j}$. By Claim 4.4 and by pseudorandomness of G we get that

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

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

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

where every r_j is of length n. Accordingly, define $\mathcal{D}^*$ to sample $\bar{r} = r_1, \ldots, r_{2^m}$ uniformly at random where $r_j \in \{0,1\}^n$. Notice that $(L^*, \mathcal{D}^*)$ satisfies the following:

1. $\Pr_{\bar{r} \leftarrow \mathcal{D}^*}[L^*(\bar{r}) = 1] = 1$.

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

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

The first item follows by the derandomization. The second item follows, since for all $j \in [2^m]$, the search problem $(L^j, \mathcal{D}')$ is hard (which follows from Lemma 4.3). In particular, any adversary A solving the search problem associated to $(L^*, \mathcal{D}^*)$ with noticeable probability $1/p(n)$ must solve one of the $(L^j, \mathcal{D}')$ with probability at least $1/(2^m p(n)) = 1/\mathsf{poly}(n)$, a contradiction. We get that $(L^*, \mathcal{D}^*)$ is a hard-on-average search problem, which is total, and thus in TFNP. □

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

4.2 From private coins to public coins

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

Theorem 4.5. *If hard-on-average* NP *languages exist then public-coin hard-on-average* NP *languages exist.*

Here we discuss two alternative versions of the proof. Though not discussed explicitly, one can see that Impagliazzo anad Levin [IL90] actually proved a similar statement. Their work in the context of average-case complexity showed that problems with natural hard distributions give rise to problems with *simple* hard distributions. Specifically, they showed that any problem with efficiently samplable hard distribution can be reduced to a problem with an efficiently computable hard distribution, i.e., where the cumulative distribution function is efficiently computable. An alternative presentation of this result by Goldreich [Gol08, Sec. 10.2.2.2] provides a step towards reducing simple distributions to hard public-coin distribution. In particular, any problem with a simple distribution can be reduced to a problem with a distribution samplable via a monotone mapping, as shown in [Gol08, Exercise 10.14]. Given the monotonicity property, we can turn any simple distribution into a public-coin distribution. Thus, we get that decision problems with samplable distributions imply decision problems with public-coin distributions.

We provide also a self-contained proof that does not rely on the notions of samplable and simple distributions. Our approach is to use the construction of universal one-way hash functions (UOWHFs) from one-way functions [NY89, Rom90, KK05]. A UOWHF is a weaker primitive than a collision resistant hash function, where an adversary chooses an input x, then it is given a hash function h sampled from the UOWHF family, and its task is to find an input $x' \neq x$ such that $h(x) = h(x')$. Therefore, any UOWHF gives rise to a hard distributional search problem (in fact, a distributional decision problem by [BCGL92]) which is public-coin: given random sample x, h find a colliding x'. We conclude the proof by showing that for any input length, a hard distribution is either already public-coin or it gives rise to a one-way function for this length. See Appendix A.2 for the complete proof.

5 Using Zero-Knowledge for TFNP Hardness

In the previous section, we have shown how to get TFNP hardness from average-case NP hardness. We either settled for a non-uniform version with no additional assumptions or used a NW-type

PRG to get a uniform version. In this section we show a different approach to proving hardness of problems in TFNP. Our main technique uses Zap, a two-message witness indistinguishable proof for NP. As discussed in the Section 1, the Zap can be further reduced to a single message by either fixing the first message non-uniformly or by using a NW-type PRG. In both cases, we get non-interactive witness indistinguishable proof system for NP (see Definition 3.9), and thus we write the proof in terms of this primitive.

The advantage of this technique is twofold. First, it is a general technique that might be used to "push" other problems inside TFNP. Second, it allows us to construct a hard problem in TFNP from injective one-way functions (see Definition 3.2) with at most two solutions. Formally, we prove the following theorem:

Theorem 5.1. *If injective one-way functions and non-interactive witness-indistinguishable proof systems for* NP *exist, then there exists a hard-on-average problem in* TFNP *such that any instance has at most two solutions.*

Proof. We define a new total search problem and call it INVERT-EITHER.

Definition 5.2 (INVERT-EITHER)**.** *Let $f\colon \{0,1\}^m \to \{0,1\}^n$ be an efficiently computable function. Let L be an* NP *language defined by the following relation: $R((y,y'),x) = 1$ if and only if $f(x) \in \{y,y'\}$. Let $(\mathsf{Prove},\mathsf{Verify})$ be a witness indistinguishable proof system for L. The input to the* INVERT-EITHER *problem is a tuple (y,y',π), where $y,y' \in \{0,1\}^n$, and $\pi \in \{0,1\}^{\mathsf{poly}(n)}$. We ask to find a string $x \in \{0,1\}^m$ satisfying one of the following:*

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

We now show that INVERT-EITHER is a total search problem.

Claim 5.3. *The* INVERT-EITHER *problem is in* TFNP*.*

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

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

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

1. choose $x' \leftarrow \{0,1\}^n$ at random and compute $y' = f(x')$.
2. choose $b \leftarrow \{0,1\}$ at random and set $y_b = y$ and $y_{1-b} = y'$.
3. compute $\pi \leftarrow \mathsf{Prove}((y_0,y_1),x')$.

4. compute $w \leftarrow A(y_0, y_1, \pi)$.
5. output w.

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

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

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

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

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

Remark 5.4. *We note that we get hardness of* Invert-Either *from any one-way function, however, the number of solutions is guaranteed to be at most two only when the one-way function is injective.*

6 Open Problems

The most immediate open problem is whether it is possible to base the hardness of any of the known subclasses of TFNP on the assumption that one-way functions exist[8]. Perhaps the most plausible one is PPP: its canonical problem is given by a circuit $\mathsf{C}\colon \{0,1\}^n \to \{0,1\}^n$ where the goal is to find a collision under C or an input x such that $\mathsf{C}(x) = 0$. Notice that by the pigeonhole principle we get that this problem is total. The hardness of PPP was shown from one-way permutations or collision-resistant hash functions. Thus, it is a prime candidate for showing hardness from one-way functions.

Recall that Bellare and Goldwasser [BG94] considered the issue of self reducibility, i.e. are there languages in NP where given a decision oracle it is hard to solve *any* corresponding FNP search problem. They showed that such languages exist under the assumption that $\mathsf{EE} \neq \mathsf{NEE}$ (double

[8]"Provable TFNP" is a subset of TFNP containing the structured classes of it like PPP, PPAD and PLS that was defined by Goldberg and Papadimitriou [GP16]. It is a natural candidate for hardness proofs under as general assumption as possible.

exponential time is not equal the non-deterministic version of it). In particular the language they constructed is in P/Poly. Can standard cryptographic type assumptions and techniques be used to show such separation results?

We have shown how to use derandomization and also zero-knowledge to prove that some (hard) problems are in TFNP. Another interesting direction is to use our techniques to push into TFNP (variants of) other natural search problems which are not known to be total.

Acknowledgements

We wish to thank Oded Goldreich and Roei Tell for their invaluable help and comments.

References

[AASY16] Benny Applebaum, Sergei Artemenko, Ronen Shaltiel, and Guang Yang. Incompressible functions, relative-error extractors, and the power of nondeterministic reductions. *Computational Complexity*, 25(2):349–418, 2016.

[AB09] Sanjeev Arora and Boaz Barak. *Computational Complexity - A Modern Approach.* Cambridge University Press, 2009.

[AIKS16] Sergei Artemenko, Russell Impagliazzo, Valentine Kabanets, and Ronen Shaltiel. Pseudorandomness when the odds are against you. In *31st Conference on Computational Complexity, CCC 2016, May 29 to June 1, 2016, Tokyo, Japan*, pages 9:1–9:35, 2016.

[AJN$^+$16] Prabhanjan Ananth, Aayush Jain, Moni Naor, Amit Sahai, and Eylon Yogev. Universal obfuscation and witness encryption: Boosting correctness and combining security. *IACR Cryptology ePrint Archive*, 2016:281, 2016.

[AS15] Gilad Asharov and Gil Segev. Limits on the power of indistinguishability obfuscation and functional encryption. In *IEEE 56th Annual Symposium on Foundations of Computer Science, FOCS 2015, Berkeley, CA, USA, 17-20 October, 2015*, pages 191–209, 2015.

[AS16] Gilad Asharov and Gil Segev. On constructing one-way permutations from indistinguishability obfuscation. In *Theory of Cryptography - 13th International Conference, TCC 2016-A, Tel Aviv, Israel, January 10-13, 2016, Proceedings, Part II*, pages 512–541, 2016.

[BCGL92] Shai Ben-David, Benny Chor, Oded Goldreich, and Michael Luby. On the theory of average case complexity. *J. Comput. Syst. Sci.*, 44(2):193–219, 1992.

[BDV16] Nir Bitansky, Akshay Degwekar, and Vinod Vaikuntanathan. Structure vs hardness through the obfuscation lens. *Electronic Colloquium on Computational Complexity (ECCC)*, 23:91, 2016.

[BFK$^+$10] Harry Buhrman, Lance Fortnow, Michal Koucký, John D. Rogers, and Nikolai K. Vereshchagin. Does the polynomial hierarchy collapse if onto functions are invertible? *Theory Comput. Syst.*, 46(1):143–156, 2010.

[BG94] Mihir Bellare and Shafi Goldwasser. The complexity of decision versus search. *SIAM J. Comput.*, 23(1):97–119, 1994.

[BGI$^+$01] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. In *CRYPTO*, volume 2139 of *Lecture Notes in Computer Science*, pages 1–18. Springer, 2001.

[BK95] Manuel Blum and Sampath Kannan. Designing programs that check their work. *J. ACM*, 42(1):269–291, 1995.

[BLV06] Boaz Barak, Yehuda Lindell, and Salil P. Vadhan. Lower bounds for non-black-box zero knowledge. *J. Comput. Syst. Sci.*, 72(2):321–391, 2006.

[BOV07] Boaz Barak, Shien Jin Ong, and Salil P. Vadhan. Derandomization in cryptography. *SIAM J. Comput.*, 37(2):380–400, 2007.

[BPR15] Nir Bitansky, Omer Paneth, and Alon Rosen. On the cryptographic hardness of finding a Nash equilibrium. In *56th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2015, Berkeley, CA, USA, October 18-20, 2015*, pages 1480–1498, 2015.

[Bra83] Gilles Brassard. Relativized cryptography. *IEEE Trans. Information Theory*, 29(6):877–893, 1983.

[BV15] Nir Bitansky and Vinod Vaikuntanathan. A note on perfect correctness by derandomization. *Electronic Colloquium on Computational Complexity (ECCC)*, 22:187, 2015.

[CDT09] Xi Chen, Xiaotie Deng, and Shang-Hua Teng. Settling the complexity of computing two-player Nash equilibria. *J. ACM*, 56(3), 2009.

[DGP09] Constantinos Daskalakis, Paul W. Goldberg, and Christos H. Papadimitriou. The complexity of computing a Nash equilibrium. *SIAM J. Comput.*, 39(1):195–259, 2009.

[DN07] Cynthia Dwork and Moni Naor. Zaps and their applications. *SIAM J. Comput.*, 36(6):1513–1543, 2007.

[DNR04] Cynthia Dwork, Moni Naor, and Omer Reingold. Immunizing encryption schemes from decryption errors. In *Advances in Cryptology - EUROCRYPT 2004, International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004, Proceedings*, pages 342–360, 2004.

[DP11] Constantinos Daskalakis and Christos H. Papadimitriou. Continuous local search. In *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2011, San Francisco, California, USA, January 23-25, 2011*, pages 790–804, 2011.

[ESY84] Shimon Even, Alan L. Selman, and Yacov Yacobi. The complexity of promise problems with applications to public-key cryptography. *Information and Control*, 61(2):159–173, 1984.

[FS90] Uriel Feige and Adi Shamir. Witness indistinguishable and witness hiding protocols. In *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing, May 13-17, 1990, Baltimore, Maryland, USA*, pages 416–426, 1990.

[GGH$^+$13] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *FOCS*, pages 40–49, 2013.

[Gol08] Oded Goldreich. *Computational complexity - a conceptual perspective.* Cambridge University Press, 2008.

[GP16] Paul W. Goldberg and Christos H. Papadimitriou. Towards a unified complexity theory of total functions. http://www.cs.ox.ac.uk/people/paul.goldberg/papers/paper-2.pdf, 2016. Unpublished manuscript.

[GPS16] Sanjam Garg, Omkant Pandey, and Akshayaram Srinivasan. Revisiting the cryptographic hardness of finding a Nash equilibrium. In *Advances in Cryptology - CRYPTO 2016 - 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part II*, pages 579–604, 2016.

[HY16] Pavel Hubáček and Eylon Yogev. Hardness of continuous local search: Query complexity and cryptographic lower bounds. *Electronic Colloquium on Computational Complexity (ECCC)*, 23:63, 2016.

[IL89] Russell Impagliazzo and Michael Luby. One-way functions are essential for complexity based cryptography (extended abstract). In *FOCS*, pages 230–235. IEEE Computer Society, 1989.

[IL90] Russell Impagliazzo and Leonid A. Levin. No better ways to generate hard NP instances than picking uniformly at random. In *31st Annual Symposium on Foundations of Computer Science, St. Louis, Missouri, USA, October 22-24, 1990, Volume II*, pages 812–821, 1990.

[Imp95] Russell Impagliazzo. A personal view of average-case complexity. In *Structure in Complexity Theory Conference*, pages 134–147. IEEE Computer Society, 1995.

[IN88] Russell Impagliazzo and Moni Naor. Decision trees and downward closures. In *Proceedings: Third Annual Structure in Complexity Theory Conference, Georgetown University, Washington, D. C., USA, June 14-17, 1988*, pages 29–38, 1988.

[IW97] Russell Impagliazzo and Avi Wigderson. $P = BPP$ if E requires exponential circuits: Derandomizing the XOR lemma. In *Proceedings of the Twenty-Ninth Annual ACM Symposium on the Theory of Computing, El Paso, Texas, USA, May 4-6, 1997*, pages 220–229, 1997.

[Jeř16] Emil Jeřábek. Integer factoring and modular square roots. *J. Comput. Syst. Sci.*, 82(2):380–394, 2016.

[JPY88] David S. Johnson, Christos H. Papadimitriou, and Mihalis Yannakakis. How easy is local search? *J. Comput. Syst. Sci.*, 37(1):79–100, 1988.

[KK05] Jonathan Katz and Chiu-Yuen Koo. On constructing universal one-way hash functions from arbitrary one-way functions. *IACR Cryptology ePrint Archive*, 2005:328, 2005.

[KMN$^+$14] Ilan Komargodski, Tal Moran, Moni Naor, Rafael Pass, Alon Rosen, and Eylon Yogev. One-way functions and (im)perfect obfuscation. In *55th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2014, Philadelphia, PA, USA, October 18-21, 2014*, pages 374–383, 2014.

[KSS11] Jeff Kahn, Michael E. Saks, and Clifford D. Smyth. The dual BKR inequality and Rudich's conjecture. *Combinatorics, Probability & Computing*, 20(2):257–266, 2011.

[Lau83] Clemens Lautemann. BPP and the polynomial hierarchy. *Inf. Process. Lett.*, 17(4):215–217, 1983.

[LNNW95] László Lovász, Moni Naor, Ilan Newman, and Avi Wigderson. Search problems in the decision tree model. *SIAM J. Discrete Math.*, 8(1):119–132, 1995.

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

[MX10] Mohammad Mahmoody and David Xiao. On the power of randomized reductions and the checkability of SAT. In *Proceedings of the 25th Annual IEEE Conference on Computational Complexity, CCC 2010, Cambridge, Massachusetts, June 9-12, 2010*, pages 64–75, 2010.

[Nao91] Moni Naor. Bit commitment using pseudorandomness. *Journal of Cryptology*, 4(2):151–158, 1991.

[NW94] Noam Nisan and Avi Wigderson. Hardness vs randomness. *J. Comput. Syst. Sci.*, 49(2):149–167, 1994.

[NY89] Moni Naor and Moti Yung. Universal one-way hash functions and their cryptographic applications. In *Proceedings of the 21st Annual ACM Symposium on Theory of Computing, May 14-17, 1989, Seattle, Washigton, USA*, pages 33–43, 1989.

[Pap94] Christos H. Papadimitriou. On the complexity of the parity argument and other inefficient proofs of existence. *J. Comput. Syst. Sci.*, 48(3):498–532, 1994.

[Rom90] John Rompel. One-way functions are necessary and sufficient for secure signatures. In *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing, May 13-17, 1990, Baltimore, Maryland, USA*, pages 387–394, 1990.

[RSS16] Alon Rosen, Gil Segev, and Ido Shahaf. Can PPAD hardness be based on standard cryptographic assumptions? *Electronic Colloquium on Computational Complexity (ECCC)*, 23:59, 2016.

[Rud89] Steven Rudich. *Limits on the Provable Consequences of One-way Functions*. PhD thesis, University of California at Berkeley, 1989.

[Sim98] Daniel R. Simon. Finding collisions on a one-way street: Can secure hash functions be based on general assumptions? In *Advances in Cryptology - EUROCRYPT '98, International Conference on the Theory and Application of Cryptographic Techniques, Espoo, Finland, May 31 - June 4, 1998, Proceeding*, pages 334–345, 1998.

[SW14] Amit Sahai and Brent Waters. How to use indistinguishability obfuscation: deniable encryption, and more. In *STOC*, pages 475–484. ACM, 2014.

A Detailed Proofs

A.1 Proof of Lemma 4.3

We give the full proof of Lemma 4.3 that we use in Section 4.1. The proof follows by a reduction to the original distributional problem $(L, \mathcal{D})$. Given a challenge x^* for $(L, \mathcal{D})$ we can create an instance σ of the search problem associated with $(L', \mathcal{D}')$ such that the challenge x^* is embedded in a random location among the "ORed" instances in σ. This can be done efficiently by the first and the third item. Then, by the second item we get that if $x^* \in L$ then a solution for σ will contain a solution for x^* with probability at least $1/k$. Overall, we gain a polynomial advantage for solving x^* which contradicts the hardness of $(L, \mathcal{D})$.

Lemma 4.3. *Let $(L, \mathcal{D})$ be a hard distributional search problem. Let $(L', \mathcal{D}')$ be a distributional search problem related to $(L, \mathcal{D})$ that satisfies the following conditions:*

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

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

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

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

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

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

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

1. Choose $i \in [k]$ at random.

2. Sample $x' \leftarrow \mathcal{D}'$ conditioned on $f_i(x') = x$ (this can be performed efficiently due to Item 3).

3. Output $w \leftarrow A(x')$.

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

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

A.2 Proof of Theorem 4.5

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

Theorem 4.5. *If hard-on-average* NP *languages exist then public-coin hard-on-average* NP *languages exist.*

Proof. We begin by showing that if one-way functions exist then there are hard-on-average distributions that are public-coin. This part of the proof follows by combining known transformations. First, it is known that if one-way functions exist then universal one-way hash functions (UOWHFs) exist [NY89, Rom90, KK05]. A UOWHF is a family of compressing functions $\mathcal{H}$ that have the following security requirement: for a random x and $h \in \mathcal{H}$ it is hard for any PPT algorithm to find an $x \neq x'$ such that $h(x) = h(x')$. We note that the constructions of such families are in fact public-coin: to sample h no private coins are used. Thus, we can define the following search problem: given x, h find an appropriate collision x'.[9] Second, we can apply to this search problem a transformation of Ben-David et al. [BCGL92] for converting any average-case search problem into an average-case decision problem. Although it was not mentioned explicitly in their work, we observe when applied to a search problem that has a public-coin distribution the resulting decision problem is public-coin as well.

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

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

One subtle issue is that we do not necessarily know for which input sizes is the distribution public-coin and for which not. Thus, for any input size n we apply both methods: we two samples

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

using randomness r_1 and r_2. For r_1 we release $r_1, \mathcal{D}(1^n, r_1)$, and we r_2 to sample from the distribution constructed from the one-way function, as described above. Finally, we take the "AND" of the both. For any n we know that one of the two will be hard, and thus overall we get hardness for any input size.

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

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

Let $\mathcal{N}$ be the infinite set of $n \in \mathbb{N}$ such that Equation (A.1) holds. We define the function family $f = \{f_n(r) = D_n(r)\}_{n \in \mathbb{N}}$. We claim that it is infeasible to invert f for the input sizes in $\mathcal{N}$:

Claim A.1. *For any PPT A there exists a negligible function $\mathsf{neg}(\cdot)$ such that for all $n \in \mathcal{N}$:*

$$\Pr_{r \in \{0,1\}^n}[A(f_n(r)) \in f_n^{-1}(f_n(r))] \leq \mathsf{neg}(n) \ .$$

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

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

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

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

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

The above claim concludes the proof. ☐