

# One-Way Functions vs. TFNP: Simpler and Improved

Lukáš Folwarczný 

Institute of Mathematics, Czech Academy of Sciences, Prague, Czech Republic  
Faculty of Mathematics and Physics, Charles University, Prague, Czech Republic

Mika Göös


EPFL, Lausanne, Switzerland

Pavel Hubáček 

Institute of Mathematics, Czech Academy of Sciences, Prague, Czech Republic  
Charles University, Faculty of Mathematics and Physics, Czech Republic

Gilbert Maystre 

EPFL, Lausanne, Switzerland

Weiqliang Yuan 

EPFL, Lausanne, Switzerland

---

## Abstract

Simon (1998) proved that it is impossible to construct collision-resistant hash functions from one-way functions using a black-box reduction. It is conjectured more generally that one-way functions do not imply, via a black-box reduction, the hardness of *any* total NP search problem (collision-resistant hash functions being just one such example). We make progress towards this conjecture by ruling out a large class of “single-query” reductions. In particular, we improve over the prior work of Hubáček et al. (2020) in two ways: our result is established via a novel simpler combinatorial technique and applies to a broader class of *semi black-box* reductions.

**2012 ACM Subject Classification** Theory of computation → Oracles and decision trees

**Keywords and phrases** TFNP, One-Way Functions, Oracle, Separation, Black-Box

**Digital Object Identifier** 10.4230/LIPIcs.ITCS.2024.50

**Related Version** *Full Version*: <https://eprint.iacr.org/2023/945>

**Funding** *Lukáš Folwarczný*: Supported by the Academy of Sciences of the Czech Republic (RVO 67985840) and by the Grant Agency of the Czech Republic (19-27871X).

*Mika Göös*: Supported by the Swiss State Secretariat for Education, Research and Innovation (SERI) under contract number MB22.00026.

*Pavel Hubáček*: Supported by the Academy of Sciences of the Czech Republic (RVO 67985840).

*Gilbert Maystre*: supported by the Swiss State Secretariat for Education, Research and Innovation (SERI) under contract number MB22.00026.

*Weiqliang Yuan*: Supported by the Swiss National Science Foundation project 200021-184656

**Acknowledgements** We thank anonymous reviewers for their helpful comments.

## 1 Introduction

Simon [32] (see also [27, §4.4.2] for exposition) famously proved that there is no black-box construction of collision-resistant hash functions (CRHFs) from one-way functions (OWFs). In particular, we may state Simon’s impossibility result (in a slightly weaker form) as proving the existence of a pair of oracles  $(f, \text{SOLVE})$  satisfying the following.



© Lukáš Folwarczný, Mika Göös, Pavel Hubáček, Gilbert Maystre, and Weiqliang Yuan; licensed under Creative Commons License CC-BY 4.0

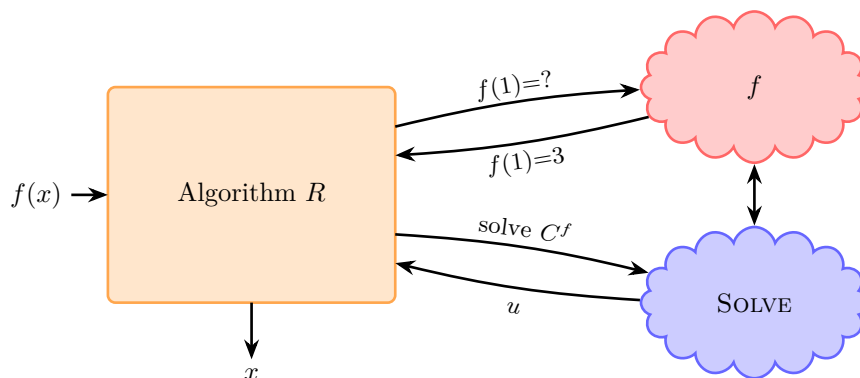
15th Innovations in Theoretical Computer Science Conference (ITCS 2024).

Editor: Venkatesan Guruswami; Article No. 50; pp. 50:1–50:14



Leibniz International Proceedings in Informatics

Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany



■ **Figure 1** Two-oracle model: The reduction algorithm aiming to invert  $f(x)$  (for a uniform random  $x$ ) has query access to  $f$  itself and also the oracle SOLVE that can solve any given TFNP instance.

- (A) **Random injection.** Oracle  $f: \{0, 1\}^* \rightarrow \{0, 1\}^*$  is an injective black-box function mapping  $n$ -bit strings to  $(n + 1)$ -bit strings. In fact,  $f$  can be chosen as a random injection.
- (B) **Collision finder.** Oracle SOLVE:  $\{0, 1\}^* \rightarrow \{0, 1\}^*$  is a black-box function that can find collisions in any *shrinking* function. In more detail, if  $C^f$  is any circuit with  $f$ -gates that computes a function from  $n$  bits to  $n - 1$  bits, then  $\text{SOLVE}(C^f)$  returns a collision pair  $(u, v)$  such that  $u \neq v$  and  $C^f(u) = C^f(v)$ .
- (C) **One-wayness.** Oracle  $f$  is *one-way* even in the presence of SOLVE. That is, given  $f(x)$  for a random  $x \sim \{0, 1\}^n$ , no poly( $n$ )-time algorithm  $R^{f, \text{SOLVE}}(f(x))$  (having oracle access to both  $f$  and SOLVE) can output  $x$  with non-negligible probability.

In summary, (A)–(C) constitute a two-oracle relativised world where injective OWFs exist but CRHFs do not (every candidate hash function constructed out of  $f$  can be broken by SOLVE). This rules out any black-box construction of CRHFs from OWFs.<sup>1</sup>

## 1.1 Our result: Generalisation to TFNP

In this paper we ask: *How far can we generalise Simon’s impossibility result?* Recent work [31, 14] has suggested a vast such generalisation. Namely, they have conjectured that it is impossible to base the hardness of any *total NP search problem* (CRHFs are just one such example) on injective OWFs in a fully black-box fashion (see also Section 1.2).

### Total search problems

An NP *search problem* is determined by a polynomial-time verifier  $V$ . On input  $x$ , the search problem is to find some witness  $u$  of polynomial length,  $|u| \leq \text{poly}(|x|)$ , that is accepted by the verifier,  $V(x, u) = 1$ . The search problem is *total* if every  $x$  admits some such polynomial-length witness. The theory of total NP search problems (class TFNP) was pioneered in the works [26, 20, 28]. See also [8, 13] for more modern expositions. For example, (worst-case) CRHFs are captured by the TFNP problem where on input a circuit

<sup>1</sup> We have stated, following [27, §4.4.2], a “two-oracle” version of Simon result where the candidate CRHF  $C^f$  is not allowed to depend on the oracle SOLVE. Simon’s original “one-oracle” result does not have this limitation.

that computes a shrinking function, the goal is to find a non-trivial collision pair. The class of all problems that reduce to this collision finding problem forms a subclass of TFNP called PWPP (“polynomial weak pigeonhole principle”) [19].

It is believed that TFNP does not have complete problems [29]. However, every problem in TFNP reduces to the following *promise* search problem: Given a circuit  $C: \{0, 1\}^n \rightarrow \{0, 1\}$  that is promised to be satisfiable, find some satisfying assignment. Indeed, if we want to solve the total search problem determined by verifier  $V$  then, on input  $x$ , we can construct the satisfiable circuit  $C(u) = V(x, u)$  whose satisfying assignments correspond to witnesses for  $x$ .

Towards extending Simon’s result to cover all TFNP problems, we replace (B) with the following *stronger* condition.

**(B<sup>+</sup>) TFNP solver.** Oracle SOLVE:  $\{0, 1\}^* \rightarrow \{0, 1\}^*$  can find a solution to any TFNP problem. In more detail, if  $C^f: \{0, 1\}^n \rightarrow \{0, 1\}$  is any circuit with  $f$ -gates that is promised to be satisfiable (for every  $f$  there is  $u$  such that  $C^f(u) = 1$ ), then  $\text{SOLVE}(C^f)$  returns some  $u$  with  $C^f(u) = 1$ . (If  $C^f$  does not satisfy the promise, then  $\text{SOLVE}(C^f)$  is undefined.)

See Section 2 for a more detailed discussion of how our SOLVE oracle is defined.

Ideally, we would like to exhibit a two-oracle world satisfying (A), (B<sup>+</sup>), (C). This would give the desired full generalisation of Simon’s result to all of TFNP. We make progress towards this goal by establishing the one-wayness of  $f$  against a large class of “single-query” algorithms. Namely, we replace (C) with the following *weaker* condition.

**(C<sup>-</sup>) Single-query one-wayness.** Oracle  $f$  is one-way against any *single-query* algorithm: Suppose  $R^{f, \text{SOLVE}}$  is a poly( $n$ )-time algorithm (see Figure 1) that on any input  $y$ :

- makes a single query to SOLVE (satisfying the promise), and
- makes no  $f$ -queries before calling SOLVE.

Then, given a challenge  $y := f(x)$  where  $x \sim \{0, 1\}^n$  is uniform, the algorithm  $R^{f, \text{SOLVE}}(y)$  cannot output  $x$  with non-negligible probability.

► **Theorem 1** (Main result). *There exists an oracle pair  $(f, \text{SOLVE})$  satisfying (A), (B<sup>+</sup>) and (C<sup>-</sup>).*

## 1.2 Comparison to [14] and other related work

Our result directly improves upon the work of Hubáček et al. [14], who were the first to rule out the existence of a restricted class of black-box constructions [17, 30] of hard TFNP problems from OWFs. Our improvements are two-fold.

**1. Ruling out semi black-box reductions.** Our results apply to the *broader* class of *semi* black-box reductions. A *fully* black-box reduction algorithm  $R^{f, \text{SOLVE}}$  must succeed in inverting the challenge  $y = f(x)$  given access to *any* TFNP solver, whereas the weaker notion of a *semi* black-box reduction allows the reduction to depend on the behaviour of the solver. The difference in the order of quantifiers makes ruling out fully black-box reductions conceptually easier. In particular, when designing a TFNP solver that would make a purported fully black-box reduction err, one can exploit the code of the reduction. Hubáček et al. [14] followed this approach and designed a TFNP solver that exploits the reduction: upon receiving a TFNP instance, their SOLVE estimates the set of likely inversion challenges  $y$  that would make the reduction query the received TFNP instance. Their solver then attempts to return a solution that does not explicitly aid in inverting the challenge  $y$ .

- 2. Simpler TFNP solver and analysis:** On the technical level, [14] employed an information theoretic incompressibility argument common to various previous works in the context of black-box separations in cryptography (e.g. [10, 9, 12]). In this work, we take a more direct, combinatorial approach to designing and analysing our solver. Crucially, this allows us to design SOLVE independently from the reduction  $R$  and extend the previous results to the weaker notion of semi black-box reductions (thus, proving a stronger separation result). We note that our techniques might be of independent interest for extending some of the known black-box separations in cryptography, e.g., towards revisiting the fully black-box separation of trapdoor predicates from trapdoor functions [11].

### Other related work

Our impossibility result can be seen as part of a long line of ongoing research seeking extrinsic evidence explaining the current lack of efficient algorithms for many subclasses of TFNP. Papadimitriou [28] already pointed out that the existence of one-way permutations implies the hardness of PPP-complete problems. More recently, similar conditional lower bounds were shown for other subclasses of TFNP (such as PPAD and CLS) under various generic [1, 6, 24, 16, 15, 23, 7, 5] or specific [19, 25, 22, 18, 3, 21] cryptographic hardness assumptions. See also [14, §1.3] for additional discussion and references.

## 2 Proof Overview

The crux of proving our main result (Theorem 1) is finding a suitable definition of  $\text{SOLVE}(C^f)$  where  $C^f$  is a satisfiable circuit: Which satisfying assignment  $u$  should we return so that  $u$  is not helpful for inverting  $f$ ? To understand why this task is tricky, it is instructive to discuss the common pitfalls associated with natural attempts at defining SOLVE.

### 2.1 Common pitfalls

Suppose we define  $\text{SOLVE}(C^f)$  as the lexicographically first satisfying assignment of  $C^f$ . Then, with a single query to SOLVE, we can in fact solve any *partial*  $\text{NP}^f$  search problem, including inverting  $f$ . For example, consider the canonical search problem where on input a circuit  $D^f$  that is not necessarily satisfiable, output either a satisfying assignment of  $D^f$  or conclude that none exists. It is easy to transform  $D^f$  into a satisfiable circuit  $C^f$  – namely, modify  $D^f$  to always accept the all-1 assignment – such that the lexicographically first satisfying assignment to  $C^f$  allows us to deduce a satisfying assignment to  $D^f$  if one exists.

The same problem arises even if we define  $\text{SOLVE}(C^f)$  as a uniformly random satisfying assignment of  $C^f$ . Here we can again solve the partial search problem associated with an  $n$ -bit circuit  $D^f$  by transforming it into a satisfiable  $(n + 1)$ -bit circuit  $C^f$  where each satisfying assignment  $u$  of  $D^f$  gives rise to two satisfying assignments,  $0u$  and  $1u$ , in  $C^f$ , and where the all-1 assignment always satisfies  $C^f$ . With probability at least  $2/3$ , the assignment returned by  $\text{SOLVE}(C^f)$  allows us to deduce a satisfying assignment for  $D^f$  if one exists. (Versions of these pitfalls were present already in Simon’s special case of CRHFs; see [2] for discussion.)

### 2.2 Stable oracle Solve

Our goal is to design  $\text{SOLVE}(C^f)$  so that it returns a satisfying assignment that leaks little information about the preimage of any particular image element. To formalise this property, let us write  $N := 2^n$  for short, and let  $\mathcal{F}$  denote the set of all injective functions from

$[N] := \{1, \dots, N\}$  to  $[2N]$ . Moreover, we let  $\mathcal{F}^y \subseteq \mathcal{F}$  for  $y \in [2N]$  denote the set of all  $f \in \mathcal{F}$  such that  $y \in \text{Im}(f)$  where  $\text{Im}(f) := f([N])$  is the image of  $f$ . Similarly, we define  $\mathcal{F}^{-y} := \mathcal{F} \setminus \mathcal{F}^y$  as the set of functions without  $y$  in their image. For  $f \in \mathcal{F}^{-y}$  and  $x \in [N]$ , we write  $f_{x \rightarrow y} \in \mathcal{F}^y$  for the function that is obtained from  $f$  by re-defining  $f(x) := y$ .

The following lemma encapsulates the key property of our solver. The solver's output is *stable* under small perturbations to  $f$ : for any image  $y \in [2N]$ , the output rarely changes when SOLVE is run on a random function  $f \sim \mathcal{F}^{-y}$  compared to a random  $f \sim \mathcal{F}^y$ .

► **Lemma 2 (Stability Lemma).** *There exists an oracle SOLVE satisfying  $(B^+)$  such that for every  $y \in [2N]$  and every satisfiable circuit  $C^f$  of size  $t$ ,*

$$\Pr_{\substack{f \sim \mathcal{F}^{-y} \\ x \sim [N]}} [\text{SOLVE}(C^f) \neq \text{SOLVE}(C^{f_{x \rightarrow y}})] \leq O(t/N^{1/2}).$$

The Stability Lemma alone is enough for us to prove property  $(C^-)$  (and, hence, the main theorem), which we do in Section 4. Next, we sketch the overall strategy for our proof of the Stability Lemma, which we present in Section 3.

### Covering of $\mathcal{F}$ by subcubes

When a size- $t$  circuit  $C^f$  is run on input  $u$ , we can view its execution as a decision tree  $Q_u$  of depth  $t$ , querying the function  $f$  on at most  $t$  domain elements  $x \in [N]$ . Each leaf of  $Q_u$  is labelled with an output bit  $C^f(u) \in \{0, 1\}$ . We identify each leaf of  $Q_u$  with a partial assignment  $T: [N] \rightarrow [2N] \cup \{*\}$  that records the query outcomes leading to that leaf. In particular, the number of non- $*$  values in  $T$  is  $|\text{Dom}(T)| \leq t$  where  $\text{Dom}(T) := \{x \in [N] : T(x) \neq *\}$ . We call such  $T$  a *conjunction of width* at most  $t$ . We say that  $T$  is *consistent* with  $f$  if  $T$  and  $f$  agree on the non- $*$  values. We also define the *subcube*  $\text{Cube}(T) := \{f \in \mathcal{F} : T \text{ is consistent with } f\}$ . Consider the set of leaves given by

$$\mathcal{T} := \bigcup_u \{T : T \text{ is a leaf of } Q_u \text{ labelled with output bit } 1\}. \quad (1)$$

That is,  $\mathcal{T}$  contains all leaves corresponding to accepting computations of  $C^f(u)$  over all  $u$ . Note that if  $C^f$  is a satisfiable circuit for all  $f \in \mathcal{F}$ , then  $\mathcal{T}$  covers the whole space of injective functions  $\mathcal{F}$ :

$$\bigcup_{T \in \mathcal{T}} \text{Cube}(T) = \mathcal{F}.$$

This is because, for each  $f \in \mathcal{F}$ , there is some  $u$  such that  $C^f(u) = 1$  and the corresponding leaf is included in the union above.

### Solver as a decision list

A width- $t$  *decision list* over  $\mathcal{F}$  is a sequence  $X = (X_i)_{i \in [m]}$ , where each  $X_i$  is a width- $t$  conjunction labelled with an output string  $\ell(X_i) \in \{0, 1\}^*$ . On input a function  $f \in \mathcal{F}$ , the decision list outputs  $\ell(X(f))$ , where  $X(f)$  is the first conjunction  $X_i$  in the sequence (if any) such that  $f \in \text{Cube}(X_i)$ . Therefore, a decision list defines a total function from  $\mathcal{F}$  to  $\{0, 1\}^*$  if the conjunctions  $X_i$  cover all of  $\mathcal{F}$ , that is,  $\bigcup_i \text{Cube}(X_i) = \mathcal{F}$ ; otherwise the decision list defines a partial function, defined only on a subset of  $\mathcal{F}$ .

We define  $\text{SOLVE}(C^f)$  by choosing a careful ordering  $X_1, \dots, X_m$  of the conjunctions in  $\mathcal{T}$ . This defines the decision list  $X := (X_i)_i$  and the solver then returns  $\ell(X(f))$  where we define  $\ell(X_i)$  as the satisfying assignment of  $C^f$  that gave rise to leaf  $X_i$ . For simplicity,

we often think of the solver as returning the whole conjunction  $X(f)$  rather than merely its output label. This will only provide more information, and Lemma 2 still holds with this interpretation.

### Non-leaky ordering

Which ordering of  $\mathcal{T}$  should we choose? Recall (Section 2.1) that a lexicographic or a random ordering cannot work. We say a conjunction  $T$  *leaks*  $y$  iff  $y \in \text{Im}(T)$  (where  $\text{Im}(T)$  is the image of  $T$  understood as a partial function). Our SOLVE constructs a decision list  $X = (X_i)_{i \in [m]}$  such that each image element is rarely leaked:

$$\forall y \in [2N]: \quad \Pr_{f \sim \mathcal{F}}[X(f) \text{ leaks } y] \leq N^{-\Omega(1)}. \quad (2)$$

To this end, we define  $X$  iteratively. Initially,  $X$  is the empty list whose output is undefined over the whole space  $\mathcal{F}$ . We then keep adding conjunctions from  $\mathcal{T}$  to  $X$  in order to increase its domain of definition. If ever in this process an image element  $y \in [2N]$  becomes close to violating (2), we start treating  $y$  as *protected*: in subsequent iterations, we only add conjunctions to  $X$  that do not further leak  $y$ . In particular, this requires us to prove a nontrivial lemma (Lemma 5) that ensures we can always find some  $T \in \mathcal{T}$  that avoids leaking protected elements and which, when appended to  $X$ , increases its domain of definition. Finally, we show that any decision list with property (2) satisfies the Stability Lemma (Lemma 2).

### 2.3 Discussion: Challenges in strengthening our result

Despite our simplified proof technique, we have been unable to extend our proof to handle algorithms in  $(C^-)$  that query SOLVE multiple times, or even algorithms that query  $f$  before the single query to SOLVE. In fact, algorithms that query  $f$  before querying the solver comprise an important class of black-box reductions. This includes, for example, the standard reduction from CRHFs to OWFs (see, e.g., [27, §4.1.3]), where the reduction algorithm first queries the hash function and only then makes a single query to a preimage finder.

One approach to handling  $f$ -queries before a single SOLVE-query would be to prove a stronger version of our Stability Lemma. In particular, we would like the stability property to hold even when the solver is run on a random function from  $\text{Cube}(T)$ , where  $T$  is a low-width conjunction that records the outcomes of the  $f$ -queries before the SOLVE-query. Importantly, the definition of SOLVE is not allowed to depend on  $T$  (that is, SOLVE does not know which values of  $f$  the algorithm has decided to query). We leave the following as an open problem.

► **Conjecture 3** (Stability for subcubes). *There is a (perhaps randomised) oracle SOLVE satisfying  $(B^+)$  such that for every  $y \in [2N]$ , a width- $t$  conjunction  $T$  not leaking  $y$ , and a size- $t$  satisfiable circuit  $C^f$ ,*

$$\Pr_{\substack{f \sim \mathcal{F}^{-y} \cap \text{Cube}(T) \\ x \sim [N]}} [SOLVE(C^f) \neq SOLVE(C^{f_{x \rightarrow y}})] \leq t^{O(1)} / N^{\Omega(1)}.$$

## 3 Oracle Solve: Definition and Stability Lemma

In this section, we first define our oracle SOLVE in Section 3.1 and then prove the Stability Lemma in Section 3.3.

### 3.1 Definition of Solve

Recall our plan from Section 2.2: Given a satisfiable circuit  $C^f$  of size  $t$ , we consider its associated width- $t$  covering  $\mathcal{T}$  of  $\mathcal{F}$  given by (1). Our SOLVE then chooses an ordering  $X_1, \dots, X_m$  of the elements of  $\mathcal{T}$  that defines the decision list  $X := (X_i)_{i \in [m]}$  computing  $\text{SOLVE}(C^f)$ . Recall also that we think of  $\text{SOLVE}(C^f)$  as returning the conjunction  $X(f)$  rather than its associated satisfying assignment.

Our goal is to find an ordering that satisfies the following non-leakiness property.

▷ **Claim 4.** For all  $C^f$  of size  $t$ , there exists a decision list  $X := (X_i)_{i \in [m]}$  such that, for every  $y \in [2N]$ , it holds that  $\Pr_{f \sim \mathcal{F}}[X(f) \text{ leaks } y] \leq 6tN^{-1/2}$ .

The construction of  $X$  consists of two phases. We start with  $X$  being the empty decision list, whose output is undefined over the whole space  $\mathcal{F}$ . Then, in the first phase, conjunctions  $T \in \mathcal{T}$  are carefully selected and appended to  $X$  to increase its domain of definition, while ensuring that no  $y$  is leaked too often. In the second phase, when  $X$  already covers most of  $\mathcal{F}$ , we complete  $X$  by adding the remaining conjunctions in  $\mathcal{T}$  to  $X$  in an arbitrary order.

We may assume that  $\mathcal{T}$  does not contain the empty conjunction, as otherwise we could define SOLVE to always return that – this covers all of  $\mathcal{F}$  while not leaking *any* image.

#### Phase I: Greedy coverage

To avoid leaking some  $y \in [2N]$  too frequently, we keep track of a “leakage vector” and protect images whose probability of leakage is too high. As long as the current coverage of  $X$  and the number of images to protect are not too large, the following lemma (proved in Section 3.2 below) will allow us to increase the coverage by adding another conjunction  $T \in \mathcal{T}$  to  $X$ .

► **Lemma 5 (Next Subcube Lemma).** *Let  $\mathcal{T}$  be a width- $t$  covering of  $\mathcal{F}$ . For any  $U \subseteq \mathcal{F}$  with  $|U| \geq 2tN^{-1/2} \cdot |\mathcal{F}|$  and for any set of protected images  $P \subseteq [2N]$  with  $|P| \leq N^{1/2}/2$ , there exists  $T^* \in \mathcal{T}$  such that  $T^*$  leaks none of  $P$  and  $\text{Cube}(T^*) \cap U \neq \emptyset$ .*

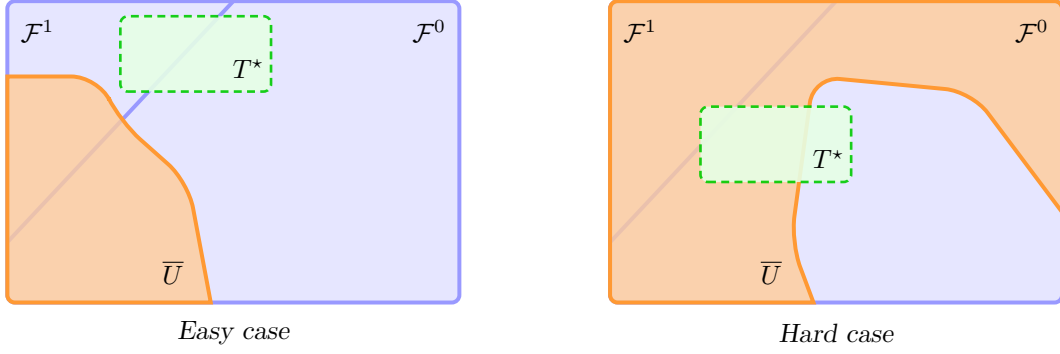
For each iteration step  $i \geq 0$  of the first phase, let  $X$  be the current decision list,  $A^{(i)} := \bigcup_{j=1}^i \text{Cube}(X_j) \subseteq \mathcal{F}$  be the set of functions already covered and let  $L^{(i)} \in [0, 1]^{2N}$  be the leakage vector defined by

$$L_y^{(i)} := \Pr_{f \sim \mathcal{F}}[f \in A^{(i)} \text{ and } X(f) \text{ leaks } y] \quad \text{where } y \in [2N].$$

Note that, in particular,  $L^{(0)} = 0^{2N}$  as  $X$  is initially the empty decision list. Fix  $\alpha := 2tN^{-1/2}$ . The first phase proceeds as long as  $|A^{(i)}| \leq (1 - \alpha)|\mathcal{F}|$  and ensures that each image is leaked with probability at most  $2\alpha$  for a random  $f \sim \mathcal{F}$ . More precisely, at step  $i$ , the set of images to be protected is defined as  $P^{(i)} := \{y \in [2N] : L_y^{(i)} \geq \alpha\}$ . Since each conjunction  $T \in \mathcal{T}$  leaks at most  $t$  different images, it holds that throughout the first phase

$$|P^{(i)}| \leq \frac{t \cdot |A^{(i)}|}{\alpha \cdot |\mathcal{F}|} \leq \frac{t}{\alpha} = \frac{N^{1/2}}{2}.$$

Therefore, Lemma 5 can be invoked with  $U := \mathcal{F} \setminus A^{(i)}$  to obtain some  $T^*$  which does not leak any image  $y \in P$  and finally we set  $X_{i+1} := T^*$  to extend the coverage of  $X$ .



■ **Figure 2** In Lemma 5, the goal is to extend the covering  $\bar{U}$  of  $\mathcal{F}$  by appending a conjunction  $T^*$  (green subcube) with  $\text{Cube}(T^*) \cap U \neq \emptyset$  that does not leak any protected image  $y \in P$ . As long as there exists an  $f \in \mathcal{F}^1 \setminus \bar{U}$ , the easy case applies and one can pick some  $T^* \in \mathcal{T}^1$  that covers  $f$ . If  $\bar{U}$  covers all of  $\mathcal{F}^1$ , then we rely on a spill-over argument that exploits the small width of the conjunctions in  $\mathcal{T}^1$ .

### Phase II: Completion

In the second phase,  $|A^{(i)}| > (1 - \alpha)|\mathcal{F}|$ , and hence the fraction of  $\mathcal{F}$  yet to be covered is tiny enough that we can afford to complete  $X$  arbitrarily: We append all the so-far unused  $T \in \mathcal{T}$  to  $X$  in an arbitrary order. This completes the description of  $X$ . It remains to establish Claim 4.

Proof of Claim 4. Fix some  $y \in [2N]$  and let  $i^*$  be the last step of the first phase, that is, the largest  $i^*$  such that  $|A^{(i^*)}| \leq (1 - \alpha)|\mathcal{F}|$ . We first bound the probability that  $y$  is leaked and  $f \in A^{(i^*+1)}$ . Let us first suppose that  $y \in P^{(i)} \setminus P^{(i-1)}$  for some  $i \leq i^*$  and observe that:

$$\begin{aligned}
 \Pr_{f \sim \mathcal{F}}[f \in A^{(i^*+1)} \text{ and } X(f) \text{ leaks } y] &\leq \Pr_{f \sim \mathcal{F}}[f \in A^{(i-1)} \text{ and } X(f) \text{ leaks } y] \\
 &\quad + \Pr_{f \sim \mathcal{F}}[f \in \text{Cube}(X_i)] \\
 &\leq \alpha + \Pr_{f \sim \mathcal{F}}[f \in \text{Cube}(X_i)] \\
 &\leq 2\alpha.
 \end{aligned}$$

The first inequality and second inequalities hold by construction: as soon as  $L_y \geq \alpha$ ,  $y$  is protected and not leaked by any upcoming subcube of the first phase. We still need to account for an extra subcube that might overflow the  $\alpha$ -threshold. Note that since  $X_i$  is not empty,  $\Pr_{f \sim \mathcal{F}}[f \in X_i] \leq 1/2N \leq \alpha$  using the crude approximation corresponding to  $X_i$  having unit width. A similar calculation can be carried out to get the same bound in the case  $y \notin P^{(i)}$  for all  $i \leq i^*$ . Finally, note that since the first phase covers most of  $\mathcal{F}$ , we have:

$$\begin{aligned}
 \Pr_{f \sim \mathcal{F}}[X(f) \text{ leaks } y] &\leq \Pr_{f \sim \mathcal{F}}[f \in A^{(i^*+1)} \text{ and } X(f) \text{ leaks } y] + \Pr_{f \sim \mathcal{F}}[f \notin A^{(i^*+1)}] \\
 &\leq 2\alpha + \alpha \\
 &= \frac{6t}{\sqrt{N}}.
 \end{aligned}$$

◁



### 3.2 Proof of Next Subcube Lemma

**Proof of Lemma 5.** Let us partition  $\mathcal{F} = \mathcal{F}^0 \cup \mathcal{F}^1$  into  $\mathcal{F}^0 := \{f \in \mathcal{F} : P \cap \text{Im}(f) \neq \emptyset\}$  and  $\mathcal{F}^1 := \{f \in \mathcal{F} : P \cap \text{Im}(f) = \emptyset\}$  and fix  $\mathcal{T}^1 := \{T \in \mathcal{T} : \text{Cube}(T) \cap \mathcal{F}^1 \neq \emptyset\}$ . Observe that each  $T \in \mathcal{T}^1$  covers some  $f \in \mathcal{F}^1$  and, as such, leaks no protected image  $y \in P$ . To prove the lemma, it is thus sufficient to exhibit some conjunction  $T^* \in \mathcal{T}^1$  with  $\text{Cube}(T^*) \cap U \neq \emptyset$ . Since  $\mathcal{T}^1$  is a covering of  $\mathcal{F}^1$ , the existence of such  $T^*$  is immediate if there exists  $f \in U \cap \mathcal{F}^1$ . If this is not the case, we leverage the fact that subcubes of  $\mathcal{T}^1$  have bounded width so that  $\mathcal{T}^1$  must ultimately cover a good portion of  $\mathcal{F}^0$ , too (see Figure 2). To make this intuition formal, we show how to transform a sample of  $D^1$ , the uniform distribution on  $\mathcal{F}^1$ , into a sample of  $D^0$ , the uniform distribution on  $\mathcal{F}^0$ . Consider the following process:

1. sample  $g \sim D^1$ ,
2. sample  $h \sim D^0$  and let  $Q := \text{Im}(h) \cap P$ ,
3. sample a random subset  $I \subseteq [N]$  of size  $|Q|$ ,
4. let  $f$  be obtained from  $g$  by redefining the values on  $I$  by  $f(I) := Q$  where the exact mapping is chosen according to a uniform random bijection  $\pi : I \rightarrow Q$ .

Observe that the distribution of  $f$  is exactly  $D^0$  and in particular that  $Q \neq \emptyset$  by definition of  $\mathcal{F}^0$ . We use this alternative way to sample  $D^0$  to argue that, with high probability, a random  $f \sim D^0$  is covered by  $\mathcal{T}^1$ . Using the notation of the above process,

$$\Pr_{f \sim D^0} [\mathcal{T}^1 \text{ covers } f] = \Pr_{g, Q, I, \pi} [\mathcal{T}^1 \text{ covers } f] = \sum_{g \in \mathcal{F}^1} \Pr_{Q, I, \pi} [\mathcal{T}^1 \text{ covers } f] \cdot D^1[g].$$

Fix some  $g \in \mathcal{F}^1$  and let  $T \in \mathcal{T}^1$  be a width- $t$  conjunction consistent with  $g$  (which exists because  $\mathcal{T}^1$  covers  $\mathcal{F}^1$ ). We argue that the output of the process, with high probability, remains consistent with  $T$ :

$$\Pr_{Q, I, \pi} [\mathcal{T}^1 \text{ covers } f] \geq \Pr_{Q, I, \pi} [T \text{ covers } f] \geq \Pr_{Q, I, \pi} [\text{Dom}(T) \cap I = \emptyset] = \sum_{q=1}^{|P|} \frac{\binom{N-t}{q}}{\binom{N}{q}} \cdot \Pr_Q[|Q| = q].$$

Note that the fraction within the sum is minimized for  $q := |Q|$ . Indeed, as more images get re-defined, the probability that  $T$  remains consistent with  $f$  shrinks. Hence, we get

$$\Pr_{Q, I, \pi} [\mathcal{T}^1 \text{ covers } f] \geq \frac{\binom{N-t}{q}}{\binom{N}{q}} = \prod_{i=0}^{q-1} \frac{N-t-i}{N-i} \geq \left(1 - \frac{t}{N-q}\right)^q \geq 1 - \frac{q \cdot t}{N-q}.$$

As  $q \leq N^{1/2}/2$ , we can bound

$$\frac{q}{N-q} = \frac{1}{\frac{N}{q} - 1} \leq \frac{1}{\frac{N}{N^{1/2}/2} - 1} = \frac{1}{2N^{1/2} - 1} \leq N^{-1/2}.$$

Finally, we have  $\Pr_{f \sim D^0} [\mathcal{T}^1 \text{ covers } f] \geq 1 - tN^{-1/2}$ . Since  $D^0$  is the uniform distribution over  $\mathcal{F}^0$ , we thus obtain that  $\mathcal{T}^1$  covers at least a  $(1 - tN^{-1/2})$ -fraction of  $\mathcal{F}^0$  and moreover, as  $U$  covers at least a  $(2tN^{-1/2})$ -fraction of  $\mathcal{F}^0$  too, there must be some  $f \in U$  that is indeed covered by some  $T^* \in \mathcal{T}^1$ . ◀

### 3.3 Proof of Stability Lemma

▶ **Lemma 2** (Stability Lemma). *There exists an oracle SOLVE satisfying  $(B^+)$  such that for every  $y \in [2N]$  and every satisfiable circuit  $C^f$  of size  $t$ ,*

$$\Pr_{\substack{f \sim \mathcal{F}^{-y} \\ x \sim [N]}} [\text{SOLVE}(C^f) \neq \text{SOLVE}(C^{f_{x \rightarrow y}})] \leq O(t/N^{1/2}).$$

## 50:10 One-Way Functions vs. TFNP: Simpler and Improved

**Proof.** Fix any  $y \in [2N]$  and a satisfiable circuit  $C^f$  of size  $t$ . Recall that  $\text{SOLVE}(C^f)$  is computed by a decision list  $X = (X_i)_i$  that returns the first conjunction  $X_i$  such that  $f \in \text{Cube}(X_i)$ . Writing  $X_i \prec X_j$  (resp.  $X_i \preceq X_j$ ) if  $i < j$  (resp.  $i \leq j$ ) we have

$$\begin{aligned} \Pr_{\substack{f \sim \mathcal{F}^{-y} \\ x \sim [N]}} [\text{SOLVE}(C^f) \neq \text{SOLVE}(C^{f_{x \rightarrow y}})] &\leq \Pr_{\substack{f \sim \mathcal{F}^{-y} \\ x \sim [N]}} [X(f) \prec X(f_{x \rightarrow y})] \\ &+ \Pr_{\substack{f \sim \mathcal{F}^{-y} \\ x \sim [N]}} [X(f_{x \rightarrow y}) \prec X(f)]. \end{aligned}$$

We bound the two terms separately (see Figure 3 for an intuition). For the first term, fix any  $f \in \mathcal{F}^{-y}$  and observe that if  $x \notin \text{Dom}(X(f))$  then  $f_{x \rightarrow y} \in \text{Cube}(X(f))$  and thus  $X(f_{x \rightarrow y}) \preceq X(f)$ . As each conjunction of  $X$  has width at most  $t$ , we therefore get  $\Pr_{x \sim [N]} [X(f) \prec X(f_{x \rightarrow y})] \leq t/N$ . Finally, by averaging over all  $f \in \mathcal{F}^{-y}$ ,

$$\Pr_{\substack{f \sim \mathcal{F}^{-y} \\ x \sim [N]}} [X(f) \prec X(f_{x \rightarrow y})] \leq t/N.$$

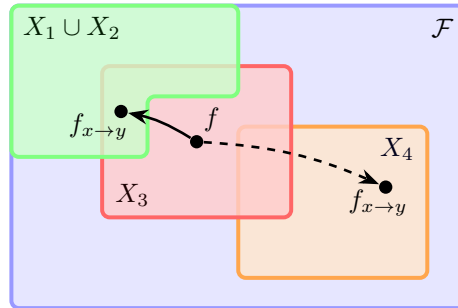
To bound the second term, let us describe an alternative way of generating the distribution  $(f, f_{x \rightarrow y})$ . First, we sample a uniform  $g \sim \mathcal{F}^y$ . Let  $w := g^{-1}(y)$  denote the preimage of  $y$  and sample a uniform  $z \in [2N] \setminus \text{Im}(g)$  in the complement of  $g$ 's image. Observe that  $(f, f_{x \rightarrow y})$  and  $(g_{w \rightarrow z}, g)$  are identically distributed. Thus, using the above notation, the second term can be equivalently expressed as

$$\Pr_{\substack{f \sim \mathcal{F}^{-y} \\ x \sim [N]}} [X(f_{x \rightarrow y}) \prec X(f)] = \Pr_{g, z} [X(g) \prec X(g_{w \rightarrow z})].$$

Observe that, as long as  $X(g)$  does not leak  $y$ ,  $X(g_{w \rightarrow z}) \in \text{Cube}(X(g))$  for any  $z \in [2N] \setminus \text{Im}(g)$  and so  $X(g_{w \rightarrow z}) \preceq X(g)$ . Therefore, since decision list  $X$  defining our  $\text{SOLVE}$  satisfies Claim 4, we get the desired bound for the second term

$$\Pr_{g, z} [X(g) \prec X(g_{w \rightarrow z})] \leq \Pr_{g \sim \mathcal{F}^y} [X(g) \text{ leaks } y] \leq 2 \Pr_{g \sim \mathcal{F}} [X(g) \text{ leaks } y] \leq 12tN^{-1/2},$$

where the second inequality follows from  $|\mathcal{F}| = 2|\mathcal{F}^y|$ . ◀



■ **Figure 3** The Stability Lemma (Lemma 2) shows that  $\text{SOLVE}$  is stable with respect to small random perturbations: for a random  $f \sim \mathcal{F}^{-y}$  and random  $x$ , it is unlikely that  $X(f_{x \rightarrow y}) \prec X(f)$  (filled arrow) or  $X(f) \prec X(f_{x \rightarrow y})$  (dashed arrow). In other words,  $\text{SOLVE}$  returns the same answer under  $f$  and  $f_{x \rightarrow y}$  with high probability.

■ **Table 1** The three hybrid games used in our proof of Lemma 6.

	$H_1$	$H_2$	$H_3$
Function		$f \sim \mathcal{F}$	
Preimage		$x \sim [N]$	
Planted Image		$y \sim [2N] \setminus \text{Im}(f)$	
Challenge	$f(x)$	$y$	$y$
Oracle	$f, \text{SOLVE}$	$f_{x \rightarrow y}, \text{SOLVE}$	$f, \text{SOLVE}$

#### 4 Proof of Main Theorem

In this section, we prove Theorem 1 by establishing the properties (A), (B<sup>+</sup>), (C<sup>-</sup>). The first two hold by construction, so we only need to prove (C<sup>-</sup>), which we do using the Stability Lemma.

► **Lemma 6.** *Suppose  $R^{f, \text{SOLVE}}: [2N] \rightarrow [N]$  is a  $t$ -time algorithm that*

- *makes a single query to SOLVE (satisfying the promise), and*
- *makes no  $f$ -queries before calling SOLVE.*

*Then the probability that  $R^{f, \text{SOLVE}}$  successfully inverts  $f$  is bounded by*

$$\Pr_{\substack{f \sim \mathcal{F} \\ x \sim [N]}} [R^{f, \text{SOLVE}}(f(x)) = x] \leq O(t/N^{1/2}).$$

**Proof.** We phrase our proof in the standard game hopping language (see, e.g., [27, §3.2.1]). In short, we argue that, compared to a real execution of  $R^{f, \text{SOLVE}}$ , the transcript of the oracle query/response pairs does not significantly change when  $R$  is given an input challenge  $y$  independent of  $x$  – in which case it is near-impossible for the algorithm to output  $x$  based on its input  $y$  (see Claim 7).

To capture the above intuition, we define three hybrid games summarised in Table 1 (Basically the same sequence of hybrid games was analysed in [4] when separating injective OWFs and  $\text{NP} \cap \text{coNP}$ . However, with a significantly less complex “decider” oracle for languages in  $\text{NP} \cap \text{coNP}$ .) In all three games, we first sample a function  $f \sim \mathcal{F}$  and a preimage  $x \sim [N]$  uniformly and independently at random.

**The “real” game  $H_1$ :** The first hybrid corresponds to the standard inversion game w.r.t.  $f$ .

Namely, the algorithm is given  $f(x)$  as the input challenge and access to the oracle pair  $(f, \text{SOLVE})$ .

**The “intermediate” game  $H_2$ :** In the second hybrid, we sample a uniformly random planted image  $y \sim [2N] \setminus \text{Im}(f)$ , and consider the perturbed oracle  $f_{x \rightarrow y}$ . The algorithm is given  $y$  as the input challenge and access to the oracle pair  $(f_{x \rightarrow y}, \text{SOLVE})$ .

**The “ideal” game  $H_3$ :** In the third hybrid, the algorithm is given a uniformly random planted image  $y \sim [2N] \setminus \text{Im}(f)$  as the input challenge and access to the original oracle pair  $(f, \text{SOLVE})$ .

In all games, the adversary  $R$  wins iff it outputs  $x$ . Next, we relate the winning probability of  $R$  in the three hybrids.

▷ **Claim 7.** For any algorithm  $R$ :

1.  $\Pr_{f,x}[R \text{ wins in } H_1] = \Pr_{f,x,y}[R \text{ wins in } H_2]$ ,
2.  $|\Pr_{f,x,y}[R \text{ wins in } H_2] - \Pr_{f,x,y}[R \text{ wins in } H_3]| \leq O(t/N^{1/2})$  and
3.  $\Pr_{f,x,y}[R \text{ wins in } H_3] = 1/N$ .

Proof. Proof of the first and the third item is trivial. The first item is due to the fact that the distribution of challenge, oracle and pre-image are identical in  $H_1$  and  $H_2$ . The third item is due to the independence of  $x$  from  $y$  in  $H_3$ . For the second item, we rely on the Stability Lemma (Lemma 2), as we explain next. Due to the bound

$$|\Pr_{f,x,y}[R \text{ wins in } H_2] - \Pr_{f,x,y}[R \text{ wins in } H_3]| \leq \Pr_{f,x,y}[R^{(f_{x \rightarrow y}, \text{SOLVE})}(y) \neq R^{(f, \text{SOLVE})}(y)],$$

it is sufficient to analyse the changes in the output of  $R$  induced by small perturbations in  $f$ . Now, fix any internal random coins used by  $R$  and the planted image  $y$ , i.e., the remaining randomness is only over  $x$  and  $f$ . Let  $E_i$  for  $i \in [t]$  denote the event that  $R$  receives the same response to the  $i$ -th oracle query w.r.t.  $f_{x \rightarrow y}$  and  $f$  and recall that the first query of  $R$  is a SOLVE-query and the remaining ones are  $f$ -queries. If all events  $E_i$  occur, the algorithm cannot distinguish between having access to  $f_{x \rightarrow y}$  or  $f$ , and, hence, it produces the same output. Thus, it is sufficient to bound the following:

$$\Pr[\neg E_1 \vee \dots \vee \neg E_t] = \Pr[\neg E_1] + \sum_{i=2}^t \Pr[\neg E_i \wedge E_1 \wedge \dots \wedge E_{i-1}]. \quad (3)$$

Observe that the satisfiable TFNP circuit submitted by  $R$  as its SOLVE-query is of size at most  $t$ . Hence, by the Stability Lemma (Lemma 2), we have  $\Pr[\neg E_1] \leq O(t/N^{1/2})$ . Consider then the  $i$ -th query for  $i \geq 2$  and fix any  $f \in \mathcal{F}^{-y}$ . If all of  $E_1, \dots, E_{i-1}$  happen, the algorithm makes the same  $i$ -th query given access to both  $f_{x \rightarrow y}$  and  $f$ . Moreover, when  $x$  is not the  $i$ -th query, the reduction will receive the same answer in the two worlds. Thus  $\neg E_i \wedge E_1 \wedge \dots \wedge E_{i-1}$  holds only if  $x$  is exactly the  $i$ -th query for  $f$ . Then by averaging over all  $f \in \mathcal{F}^{-y}$ , we have  $\Pr_{f,x}[\neg E_i \wedge E_1 \wedge \dots \wedge E_{i-1}] \leq 1/N$ . Combining these bounds, we conclude that (3)  $\leq O(t/N^{1/2}) + t/N \leq O(t/N^{1/2})$  and the claim follows by averaging over all choices of randomness of  $R$  and images  $y$ .  $\triangleleft$

We conclude the proof of Lemma 6 by combining all three items of Claim 7:

$$\begin{aligned} \Pr[R^{f, \text{SOLVE}}(f(x)) = x] &= \Pr[R \text{ wins in } H_1] \\ &= \Pr[R \text{ wins in } H_2] \\ &\leq \Pr[R \text{ wins in } H_3] + |\Pr[R \text{ wins in } H_2] - \Pr[R \text{ wins in } H_3]| \\ &\leq 1/N + O(t/N^{1/2}) \\ &= O(t/N^{1/2}). \end{aligned} \quad \blacktriangleleft$$

---

## References

- 1 Tim Abbot, Daniel Kane, and Paul Valiant. On algorithms for Nash equilibria. Unpublished manuscript, 2004. URL: <http://web.mit.edu/tabbott/Public/final.pdf>.
- 2 Paul Baecker. Simon's circuit. Cryptology ePrint Archive, Paper 2014/476, 2014. URL: <https://eprint.iacr.org/2014/476>.
- 3 Nir Bitansky, Arka Rai Choudhuri, Justin Holmgren, Chethan Kamath, Alex Lombardi, Omer Paneth, and Ron D. Rothblum. PPAD is as hard as LWE and iterated squaring. In Eike Kiltz and Vinod Vaikuntanathan, editors, *Theory of Cryptography – 20th International Conference, TCC 2022, Chicago, IL, USA, November 7-10, 2022, Proceedings, Part II*, volume 13748 of *Lecture Notes in Computer Science*, pages 593–622. Springer, 2022. doi:10.1007/978-3-031-22365-5\_21.
- 4 Nir Bitansky, Akshay Degwekar, and Vinod Vaikuntanathan. Structure versus hardness through the obfuscation lens. *SIAM J. Comput.*, 50(1):98–144, 2021. doi:10.1137/17M1136559.

- 5 Nir Bitansky and Idan Gerichter. On the cryptographic hardness of local search. In Thomas Vidick, editor, *11th Innovations in Theoretical Computer Science Conference, ITCS 2020, January 12-14, 2020, Seattle, Washington, USA*, volume 151 of *LIPICs*, pages 6:1–6:29. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPICs.ITCS.2020.6.
- 6 Nir Bitansky, Omer Paneth, and Alon Rosen. On the cryptographic hardness of finding a Nash equilibrium. In Venkatesan Guruswami, editor, *IEEE 56th Annual Symposium on Foundations of Computer Science, FOCS 2015, Berkeley, CA, USA, 17-20 October, 2015*, pages 1480–1498. IEEE Computer Society, 2015. doi:10.1109/FOCS.2015.94.
- 7 Arka Rai Choudhuri, Pavel Hubáček, Chethan Kamath, Krzysztof Pietrzak, Alon Rosen, and Guy N. Rothblum. Finding a Nash equilibrium is no easier than breaking Fiat-Shamir. In Moses Charikar and Edith Cohen, editors, *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC 2019, Phoenix, AZ, USA, June 23-26, 2019*, pages 1103–1114. ACM, 2019. doi:10.1145/3313276.3316400.
- 8 Constantinos Daskalakis. Equilibria, fixed points, and computational complexity. In *Proceedings of the International Congress of Mathematicians (ICM)*. World Scientific, 2019. doi:10.1142/9789813272880\_0009.
- 9 Rosario Gennaro, Yael Gertner, Jonathan Katz, and Luca Trevisan. Bounds on the efficiency of generic cryptographic constructions. *SIAM J. Comput.*, 35(1):217–246, 2005.
- 10 Rosario Gennaro and Luca Trevisan. Lower bounds on the efficiency of generic cryptographic constructions. In *FOCS*, pages 305–313. IEEE Computer Society, 2000.
- 11 Yael Gertner, Tal Malkin, and Omer Reingold. On the impossibility of basing trapdoor functions on trapdoor predicates. In *42nd Annual Symposium on Foundations of Computer Science, FOCS 2001, 14-17 October 2001, Las Vegas, Nevada, USA*, pages 126–135. IEEE Computer Society, 2001. doi:10.1109/SFCS.2001.959887.
- 12 Iftach Haitner, Jonathan J. Hoch, Omer Reingold, and Gil Segev. Finding collisions in interactive protocols – tight lower bounds on the round and communication complexities of statistically hiding commitments. *SIAM J. Comput.*, 44(1):193–242, 2015.
- 13 Alexandros Hollender. *Structural results for total search complexity classes with applications to game theory and optimisation*. PhD thesis, University of Oxford, 2021. URL: <https://ora.ox.ac.uk/objects/uuid:67e2d80b-76bf-4b49-9b7d-8bbd91633dd7>.
- 14 Pavel Hubáček, Chethan Kamath, Karel Král, and Veronika Slívová. On average-case hardness in TFNP from one-way functions. In Rafael Pass and Krzysztof Pietrzak, editors, *Theory of Cryptography – 18th International Conference, TCC 2020, Durham, NC, USA, November 16-19, 2020, Proceedings, Part III*, volume 12552 of *Lecture Notes in Computer Science*, pages 614–638. Springer, 2020. doi:10.1007/978-3-030-64381-2\_22.
- 15 Pavel Hubáček, Moni Naor, and Eylon Yogev. The journey from NP to TFNP hardness. In Christos H. Papadimitriou, editor, *8th Innovations in Theoretical Computer Science Conference, ITCS 2017, January 9-11, 2017, Berkeley, CA, USA*, volume 67 of *LIPICs*, pages 60:1–60:21. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2017. doi:10.4230/LIPICs.ITCS.2017.60.
- 16 Pavel Hubáček and Eylon Yogev. Hardness of continuous local search: Query complexity and cryptographic lower bounds. *SIAM J. Comput.*, 49(6):1128–1172, 2020. doi:10.1137/17M1118014.
- 17 Russell Impagliazzo and Steven Rudich. Limits on the provable consequences of one-way permutations. In *Proceedings of the 21st Annual ACM Symposium on Theory of Computing, May 14-17, 1989, Seattle, Washington, USA*, pages 44–61. ACM, 1989.
- 18 Ruta Jawale, Yael Tauman Kalai, Dakshita Khurana, and Rachel Yun Zhang. SNARGs for bounded depth computations and PPAD hardness from sub-exponential LWE. In Samir Khuller and Virginia Vassilevska Williams, editors, *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing, Virtual Event, Italy, June 21-25, 2021*, pages 708–721. ACM, 2021. doi:10.1145/3406325.3451055.

- 19 Emil Jeřábek. Integer factoring and modular square roots. *Journal of Computer and System Sciences*, 82(2):380–394, March 2016. doi:10.1016/j.jcss.2015.08.001.
- 20 David Johnson, Christos Papadimitriou, and Mihalis Yannakakis. How easy is local search? *Journal of Computer and System Sciences*, 37(1):79–100, 1988. doi:10.1016/0022-0000(88)90046-3.
- 21 Yael Tauman Kalai, Alex Lombardi, and Vinod Vaikuntanathan. SNARGs and PPAD hardness from the decisional Diffie-Hellman assumption. In Carmit Hazay and Martijn Stam, editors, *Advances in Cryptology – EUROCRYPT 2023 – 42nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Lyon, France, April 23-27, 2023, Proceedings, Part II*, volume 14005 of *Lecture Notes in Computer Science*, pages 470–498. Springer, 2023. doi:10.1007/978-3-031-30617-4\_16.
- 22 Yael Tauman Kalai, Omer Paneth, and Lisa Yang. Delegation with updatable unambiguous proofs and PPAD-hardness. In Daniele Micciancio and Thomas Ristenpart, editors, *Advances in Cryptology – CRYPTO 2020 – 40th Annual International Cryptology Conference, CRYPTO 2020, Santa Barbara, CA, USA, August 17-21, 2020, Proceedings, Part III*, volume 12172 of *Lecture Notes in Computer Science*, pages 652–673. Springer, 2020. doi:10.1007/978-3-030-56877-1\_23.
- 23 Ilan Komargodski, Moni Naor, and Eylon Yogev. White-box vs. black-box complexity of search problems: Ramsey and graph property testing. *J. ACM*, 66(5):34:1–34:28, 2019.
- 24 Ilan Komargodski and Gil Segev. From Minicrypt to Obfustopia via private-key functional encryption. *J. Cryptol.*, 33(2):406–458, 2020. doi:10.1007/s00145-019-09327-x.
- 25 Alex Lombardi and Vinod Vaikuntanathan. Fiat-Shamir for repeated squaring with applications to PPAD-hardness and VDFs. In Daniele Micciancio and Thomas Ristenpart, editors, *Advances in Cryptology – CRYPTO 2020 – 40th Annual International Cryptology Conference, CRYPTO 2020, Santa Barbara, CA, USA, August 17-21, 2020, Proceedings, Part III*, volume 12172 of *Lecture Notes in Computer Science*, pages 632–651. Springer, 2020. doi:10.1007/978-3-030-56877-1\_22.
- 26 Nimrod Megiddo and Christos Papadimitriou. On total functions, existence theorems and computational complexity. *Theoretical Computer Science*, 81(2):317–324, 1991. doi:10.1016/0304-3975(91)90200-L.
- 27 Arno Mittelbach and Marc Fischlin. *The Theory of Hash Functions and Random Oracles – An Approach to Modern Cryptography*. Information Security and Cryptography. Springer, 2021. doi:10.1007/978-3-030-63287-8.
- 28 Christos Papadimitriou. On the complexity of the parity argument and other inefficient proofs of existence. *Journal of Computer and System Sciences*, 48(3):498–532, 1994. doi:10.1016/s0022-0000(05)80063-7.
- 29 Pavel Pudlák. On the complexity of finding falsifying assignments for Herbrand disjunctions. *Archive for Mathematical Logic*, 54(7-8):769–783, 2015. doi:10.1007/s00153-015-0439-6.
- 30 Omer Reingold, Luca Trevisan, and Salil P. Vadhan. Notions of reducibility between cryptographic primitives. In *Theory of Cryptography, First Theory of Cryptography Conference, TCC 2004, Cambridge, MA, USA, February 19-21, 2004, Proceedings*, pages 1–20, 2004.
- 31 Alon Rosen, Gil Segev, and Ido Shahaf. Can PPAD hardness be based on standard cryptographic assumptions? *J. Cryptol.*, 34(1):8, 2021. doi:10.1007/s00145-020-09369-6.
- 32 Daniel R. Simon. Finding collisions on a one-way street: Can secure hash functions be based on general assumptions? In Kaisa Nyberg, editor, *Advances in Cryptology – EUROCRYPT ’98, International Conference on the Theory and Application of Cryptographic Techniques, Espoo, Finland, May 31 – June 4, 1998, Proceeding*, volume 1403 of *Lecture Notes in Computer Science*, pages 334–345. Springer, 1998. doi:10.1007/BFb0054137.