Electronic Colloquium on Computational Complexity, Report No. 92 (2016)

Indistinguishability Obfuscation Does Not Reduce to Structured Languages

Gilad Asharov[*] Alon Rosen[†] Gil Segev[‡]

Abstract

We prove that indistinguishability obfuscation (iO) and one-way functions do not naturally reduce to any language within $\mathsf{NP}\cap\mathsf{coNP}$. This is proved within the framework introduced by Asharov and Segev (FOCS '15) that captures the vast majority of techniques that have been used so far in iO-based constructions.

Our approach is based on a two-fold generalization of a classic result due to Impagliazzo and Naor (Structure in Complexity Theory '88) on non-deterministic computations via decision trees. First, we generalize their approach from the, rather restrictive, model of decision trees to arbitrary computations. Then, we further generalize the argument within the Asharov-Segev framework.

Our result explains why iO does not seem to suffice for certain applications, even when combined with the assumption that one-way functions exist. In these cases it appears that additional, more structured, assumptions are necessary. In addition, our result suggests that any attempt for ruling out the existence of iO by reducing it "ad absurdum" to a potentially efficiently-decidable language may encounter significant difficulties.

[*]IBM T.J. Watson Research Center. Supported by NSF Grant No. 1017660. Email: `gsasharo@us.ibm.com`.

[†]Efi Arazi School of Computer Science, IDC Herzliya, Israel. Email: `alon.rosen@idc.ac.il`. Supported by ISF grant no. 1255/12, NSF-BSF Cyber Security and Privacy grant no. 2014/632, and by the ERC under the EU's Seventh Framework Programme (FP/2007-2013) ERC Grant Agreement no. 307952.

[‡]School of Computer Science and Engineering, Hebrew University of Jerusalem, Jerusalem 91904, Israel. Email: `segev@cs.huji.ac.il`. Supported by the European Union's 7th Framework Program (FP7) via a Marie Curie Career Integration Grant, by the Israel Science Foundation (Grant No. 483/13), by the Israeli Centers of Research Excellence (I-CORE) Program (Center No. 4/11), by the US-Israel Binational Science Foundation (Grant No. 2014632), and by a Google Faculty Research Award.

ISSN 1433-8092

1 Introduction

Program obfuscation, the task of making code unintelligible while preserving its functionality, was first rigorously studied by Barak et al. [BGI+01]. In that work, they defined a notion of *indistinguishability obfuscation (iO)*, which guarantees that obfuscations of any two functionally equivalent circuits of the same size are computationally indistinguishable. As shown by Barak et al. iO is always realizable, albeit inefficiently: the obfuscator can simply canonicalize the input circuit C by outputting the lexicographically first circuit that computes the same function. The interest in iO has gained considerable momentum following the works of Garg et al. [GGH+13] who proposed an *efficient* candidate construction of iO for all circuits and of Sahai and Waters [SW14], who demonstrated the applicability of iO for the construction of many powerful cryptographic primitives.

In spite of its wide applicability, we cannot even hope to prove that iO implies one-way functions. Indeed, if $\mathsf{P} = \mathsf{NP}$ then one-way functions do not exist but iO does exist (since the lexicographically first circuit that computes the same function can be found efficiently). Therefore, we do not expect to build many "cryptographically interesting" tools just from iO, but usually need to combine it with other assumptions. It is known that iO can be combined with the assumption that $\mathsf{NP} \neq \mathsf{coRP}$ to obtain one-way functions [KMN+14], which (again, combined with iO) give rise to many powerful primitives such as public-key encryption, CCA-secure encryption, identity-based encryption, attribute-based encryption, NIZKs, deniable encryption, and more [SW14].

At the same time, some primitives still elude construction based solely on iO and one-way functions. In such cases, one typically combines iO with more stuctured building blocks. An example for such a primitive is homomorphic encryption, for which the only known iO-based construction combines iO with re-randomizable encryption [CLT+15]. The latter building block is only known to be constructed based on structured number-theoretic or lattice-based assumptions.

In the case of homomorphic encryption (or any other primitive that implies collision-resistant hashing [IKO05]), assuming structured primitives appears somewhat necessary. As shown by Asharov and Segev [AS15] there is no construction of collision-resistant hash functions based on one-way functions and iO, as long as the obfuscator is used in a black-box manner (a related result, discussed in Section 1.3 was shown for one-way permutations [AS16]). This appears to mirror the situation of one-way functions, for which analogous separations were shown by Simon [Sim98] and Rudich [Rud88]. This leads us to ask:

Does iO exhibit the same lack of structure as one-way functions?

Traditionally, highly-structured cryptography has been associated with languages in $\mathsf{NP} \cap \mathsf{coNP}$. One prime example is natural decision languages associated with the search problem of inverting a one-way permutation. Other examples include many of the specific number theoretic assumptions underlying modern cryptography, such as factoring and discrete logarithm, as well as approximating shortest vectors within any factor larger than the square root of the lattice dimension [GG00, AR05].

Thus, a natural way to rephrase the above question is to ask whether iO implies any hard to decide language in $\mathsf{NP} \cap \mathsf{coNP}$. Beyond shedding more light on the nature of iO an answer to this question would also have concrete implications on our ability to rule out its existence. Perhaps the most basic approach for ruling out iO is to reduce it to a language that can be efficiently decided. Languages within $\mathsf{NP} \cap \mathsf{coNP}$ are certainly more likely to be efficiently decidable than languages outside of it. Indeed, this is one of the main reasons for which cryptographers seldom sleep well [Kil88]. Showing that iO does not reduce to such languages may be interpreted as evidence that ruling out its existence may be more challenging that one may expect.

1.1 Our Contributions

Our main result is the following theorem:

Theorem 1.1. *There is no fully black-box construction of an* $\mathsf{NP}\cap\mathsf{coNP}$*-language from a one-way function* f *and an indistinguishability obfuscator for the class of all oracle-aided circuits* C^f.

Following Asharov and Segev [AS15, AS16], by considering indistinguishability obfuscation for *oracle-aided* circuits, we capture the vast majority of techniques that have been used so far in constructions based on indistinguishability obfuscation. These include, in particular, *non-black-box* techniques such as the punctured programming approach of Sahai and Waters [SW14] and its many variants (e.g., [BPR15, BPW16, GPS15, GPS+16]), as well as sub-exponential security assumptions (we refer the reader to Section 1.2.1 for an overview of our framework and of the type of constructions that it captures).

We emphasize that our result considers a one-way function and an indistinguishability obfuscator as two independent building blocks. This is due to the following three main reasons. First, although indistinguishability obfuscation is known to imply one-way functions under reasonable worst-case assumptions [KMN+14], this enables us to prove an unconditional result. Second, since indistinguishability obfuscation on its own does not seem to imply any "useful" form of hardness (e.g., if $\mathsf{P} = \mathsf{NP}$ then efficient indistinguishability obfuscation is possible), proving such a result without taking one-way functions into account does not seem to have any meaningful cryptographic implications. Finally, and most importantly, this enables us to capture the vast majority of techniques that have been used so far in constructions based on indistinguishability obfuscation. Specifically, as a corollary, we rule out fully black-box constructions of an $\mathsf{NP}\cap\mathsf{coNP}$-language from any building block that can be constructed based on a one-way function and indistinguishability obfuscation for oracle-aided circuits. These include, trapdoor functions [SW14], one-way permutations and trapdoor permutations [BPW16, GPS+16], deniable encryption [SW14], oblivious transfer [SW14], functional encryption [Wat15], and many more.

1.2 Overview of Our Approach

We present a distribution over oracles relative to which the following two properties hold: (1) There exist a one-way function f and an indistinguishability obfuscator for the class of all oracle-aided circuits C^f, but (2) any oracle-aided $\mathsf{NP}\cap\mathsf{coNP}$-language can be decided using a relatively small number of oracle queries. The distribution is based on the Asharov-Segev framework [AS15, AS16], who showed that it enables the existence of the two building blocks under consideration. Our effort is thus devoted to devising an algorithm that decides oracle-aided $\mathsf{NP}\cap\mathsf{coNP}$-languages.

Our approach in constructing such an algorithm is based on a two-fold generalization of a classic result due to Impagliazzo and Naor [IN88] on non-deterministic computations via decision trees. Roughly speaking, they showed that for the *decision-tree analogues* of the oracle-aided complexity classes NP and coNP it holds that $\mathsf{NP} \cap \mathsf{coNP} \subseteq \mathsf{P}^{\mathsf{TFNP}}$. First, we generalize their approach from the, rather restrictive, model of decision trees to arbitrary computations. Then, we generalize their approach to arbitrary computations with the framework of Asharov and Segev, enabling us to prove meaningful negative results on the power of indistinguishability obfuscation.

In Section 1.2.1 we describe the framework and the specific distribution over oracles that enable us to prove a meaningful impossibility result for constructions that are based on indistinguishability obfuscation. Next, in Section 1.2.2, as a warm-up we show that the techniques of Impagliazzo and Naor [IN88] generalize from decision trees to arbitrary computations. We conclude this overview with a discussion in Section 1.2.3 of the main challenges that arise when trying to extend these techniques to our framework.

1.2.1 Capturing Non-Black-Box Constructions via $i\mathcal{O}$ for Oracle-Aided Circuits

Constructions that are based on indistinguishability obfuscation are almost always *non-black-box*. This fact makes it extremely challenging to prove any meaningful impossibility results using our current techniques for ruling out black-box constructions [IR89, RTV04]. As an example, a typical such construction would apply the obfuscator to a function that uses the evaluation circuit of a pseudorandom generator or a pseudorandom function, and this requires *specific implementations* of its underlying building blocks.

However, as observed by Asharov and Segev [AS15], most of the non-black-box techniques that are used with such constructions have essentially the same flavor: The obfuscator is applied to functions that can be constructed in a fully black-box manner from a low-level primitive, such as a one-way function. In particular, the vast majority of constructions rely on the obfuscator itself in a black-box manner. By considering the stronger primitive of an indistinguishability obfuscator for *oracle-aided* circuits (see Definition 2.2), Asharov and Segev showed that such non-black-box techniques in fact directly translate into black-box ones. These include, in particular, non-black-box techniques such as the punctured programming approach of Sahai and Waters [SW14] and its variants (as well as sub-exponential security assumptions – which are already captured by most frameworks for black-box impossibility results).

Our result is obtained by presenting an oracle Γ relative to which the following two properties hold: (1) any language L in $\mathsf{NP}\cap\mathsf{coNP}$ can be decided, and (2) there exist an *exponentially-secure* one-way function f and an *exponentially-secure* indistinguishability obfuscator $i\mathcal{O}$ for the class of all polynomial-size oracle-aided circuits C^f. Our oracle is the same as in [AS16], and consists of three functions: (1) a random function f that will serve as the one-way function, (2) a random injective length-increasing function $\mathcal{O}$ that will serve as the obfuscator (an obfuscation of an oracle-aided circuit C is a "handle" $\mathcal{O}(C, r)$ for a uniformly-chosen string r), and (3) a function Eval that enables evaluations of obfuscated circuits (Eval has access to both f and $\mathcal{O}$): Given a handle $\mathcal{O}(C, r)$ and an input x, it "finds" C and returns $C^f(x)$.

The vast majority of our effort is in showing that relative to Γ any $\mathsf{NP}\cap\mathsf{coNP}$-language can be decided using a rather small number of oracle queries. As for the second part, we derive the existence of an exponentially-secure one-way function and an exponentially-secure indistinguishability obfuscator directly from [AS16].

1.2.2 Warm-up: Deciding $\mathsf{NP}\cap\mathsf{coNP}$ Languages Relative to a Random Oracle

In this section we show that any oracle-aided $\mathsf{NP}\cap\mathsf{coNP}$-language can be decided using a polynomial number of queries in the random-oracle model. We begin with providing some essential notation, and then describe our algorithm for deciding any such language.

Notation. An oracle-aided language L defines a set $L^f \subseteq \{0,1\}^*$ for any possible function $f : \{0,1\}^* \to \{0,1\}^*$. This naturally leads to the following standard definition of the oracle-aided complexity class $\mathsf{NP} \cap \mathsf{coNP}$ (also known as a *type-2* complexity class – see Section 2.1 for more details): Given an oracle-aided language L, we say that $L \in \mathsf{NP} \cap \mathsf{coNP}$ if there exist oracle-aided polynomial-size circuits $\mathcal{R}$ and $\overline{\mathcal{R}}$ and a polynomial $p(\cdot)$, such that for any function f and for any $x \in \{0,1\}^*$ *exactly one* of the following two properties is satisfied:

- $x \in L^f$: There exists a witness $w \in \{0,1\}^{p(|x|)}$ such that $\mathcal{R}^f(x, w) = 1$ and for any $w' \in \{0,1\}^{p(|x|)}$ it holds that $\overline{\mathcal{R}}^f(x, w') = 0$.
- $x \notin L^f$: There exists witness $w \in \{0,1\}^{p(|x|)}$ such that $\overline{\mathcal{R}}^f(x, w) = 1$ and for any $w' \in \{0,1\}^{p(|x|)}$ it holds that $\mathcal{R}^f(x, w') = 0$.

In other words, for any function f, we have that $\mathcal{R}^f$ and $\overline{\mathcal{R}}^f$ compute the witness relations of L^f and of it complement $\overline{L}^f$.

The algorithm. Given any such pair $(\mathcal{R}, \overline{R})$ associated with an oracle-aided $\mathsf{NP} \cap \mathsf{coNP}$-language L, we now describe an oracle-aided algorithm that decides the language L^f for any function f. That is, for any function f and for any instance $x \in \{0,1\}^n$ it holds that $\mathcal{A}^f(x) = \mathbf{YES}$ if and only if $x \in L^f$. The algorithm $\mathcal{A}$ first initializes an empty set of queries/answers Q which will contain the actual queries made by $\mathcal{A}$ to the true oracle f. It then repeats the following for polynomial number of iterations (the polynomial is determined by the number of oracle-gates in the circuits $\mathcal{R}$ and $\overline{\mathcal{R}}$):

1. **Simulating YES:** $\mathcal{A}$ finds an oracle f'_{yes} that is consistent with Q, and a witness w'_{yes} such that $\mathcal{R}^{f'_{\mathsf{yes}}}(x, w'_{\mathsf{yes}}) = 1$.
   If there does not exist a pair $(f'_{\mathsf{yes}}, w'_{\mathsf{yes}})$ as above, then it halts and outputs **NO** (i.e., $x \notin L^f$).
2. **Simulating NO:** $\mathcal{A}$ finds an oracle f'_{no} that is consistent with Q, and a witness w'_{no} such that $\overline{\mathcal{R}}^{f'_{\mathsf{no}}}(x, w'_{\mathsf{no}}) = 1$.
   If there does not exist a pair $(f'_{\mathsf{no}}, w'_{\mathsf{no}})$ as above, then $\mathcal{A}$ halts and outputs **YES** (i.e., $x \in L^f$).
3. **Update:** $\mathcal{A}$ queries f with all inputs to oracle gates in the computations $\mathcal{R}^{f'_{\mathsf{yes}}}(x, w'_{\mathsf{yes}})$ and $\overline{\mathcal{R}}^{f'_{\mathsf{no}}}(x, w'_{\mathsf{no}})$.

In order to prove the correctness of the algorithm, there are two cases to consider according to whether $x \in L^f$ or not. In the first case, assume that $x \in L^f$ and let $w^*_{\mathsf{yes}} \in \{0,1\}^{p(n)}$ be the lexicographically smallest witness for which $\mathcal{R}^f(x, w^*_{\mathsf{yes}}) = 1$. We first claim that in that case, $\mathcal{A}$ never halts and outputs **NO** during the computation (in Step 1). This is because the pair (f, w^*_{yes}) exists, Q is always consistent with the true oracle f, and thus $\mathcal{A}$ can always set $(f'_{\mathsf{yes}}, w'_{\mathsf{yes}}) = (f, w^*_{\mathsf{yes}})$.

The proof then relies on the following observation: In each iteration, either (1) $\mathcal{A}$ halts and outputs **YES** in Step 2, or (2) in the update phase, $\mathcal{A}$ queries f with at least one new query that is made by $\mathcal{R}$ during the computation of $\mathcal{R}^f(x, w^*_{\mathsf{yes}}) = 1$.

Intuitively, if neither of the above holds, then we can construct a "hybrid" oracle $\widetilde{f}$ that behaves like f in the computation $\mathcal{R}^f(x, w^*_{\mathsf{yes}}) = 1$ and behaves like f'_{no} in the computation $\overline{\mathcal{R}}^{f'_{\mathsf{no}}}(x, w'_{\mathsf{no}}) = 1$. This hybrid oracle can be constructed since the two evaluations $\mathcal{R}^f(x, w^*_{\mathsf{yes}}) = 1$ and $\overline{\mathcal{R}}^{f'_{\mathsf{no}}}(x, w'_{\mathsf{no}}) = 1$ have no further intersection queries rather than the queries that are already in Q. According to this hybrid oracle $\widetilde{f}$, it holds that $\mathcal{R}^{\widetilde{f}}(x, w^*_{\mathsf{yes}}) = \overline{\mathcal{R}}^{\widetilde{f}}(x, w'_{\mathsf{no}}) = 1$, and thus $x \in L^{\widetilde{f}}$ and $x \notin L^{\widetilde{f}}$ simultaneously, in contradiction to the assumption that the pair $(\mathcal{R}^f, \overline{\mathcal{R}}^f)$ describes a language $L^{\widetilde{f}}$ for all functions $\widetilde{f}$.

Using this claim, since there are only polynomially many f-queries in the evaluation $\mathcal{R}^f(x, w^*_{\mathsf{yes}}) = 1$, the algorithm must halt and output **YES** after a polynomial number of iterations (more specifically, after at most $q+1$ iterations, where q is the number of oracle gates in the circuit $\mathcal{R}$). This is because after $q+1$ iterations, the algorithm cannot find a pair $(f'_{\mathsf{no}}, w'_{\mathsf{no}})$ in Step 2 that is consistent with Q that satisfies $\overline{\mathcal{R}}^{f'_{\mathsf{no}}}(x, w'_{\mathsf{no}}) = 1$, since all such oracles f'_{no} (that are consistent with Q) also satisfy $\mathcal{R}^{f'_{\mathsf{no}}}(x, w^*_{\mathsf{yes}}) = 1$. We therefore conclude that if $x \in L^f$, then $\mathcal{A}$ always outputs **YES** after polynomially-many iterations.

The case of $x \notin L^f$ is proven analogously, where the two executions that are being considered in each iteration are $\mathcal{R}^{f'_{\mathsf{yes}}}(x, w'_{\mathsf{yes}})$ and $\overline{\mathcal{R}}^f(x, w^*_{\mathsf{no}}) = 1$, where $w^*_{\mathsf{no}} \in \{0,1\}^{p(n)}$ is the lexicographically smallest witness for non-membership. We conclude that $\mathcal{A}$ always decides whether $x \in L^f$ or not.

1.2.3 Deciding $\mathsf{NP}\cap\mathsf{coNP}$ Languages Relative to Our Oracle

We extend the attack described above to work relative to our oracle, which is a much more structured oracle than a random oracle and therefore raises the following three technical challenges described below. We remark that the work of [AS16] deals with somewhat similar challenges, but our algorithm is significantly more involved than the inverter algorithm of [AS16] (and therefore our case turns out significantly more challenging).

Recall that our oracle Γ consists of three different oracles: A length-preserving function f, an *injective* length-increasing function $\mathcal{O}$, and an "evaluation" oracle Eval that depends on both f and $\mathcal{O}$. We now sketch the challenges that these oracles introduce.

The first challenge is that the evaluation oracle Eval is not just a "simple" function. This oracle performs (by definition) exponential-time computations (e.g., an exponential number of queries to $\mathcal{O}$). The second challenge is that since the oracle Eval depends on both f and $\mathcal{O}$, each query to Eval determines many other queries to f and $\mathcal{O}$ implicitly, which we need to make sure that they are considered in the attack. Specifically, given the structured dependencies between f, $\mathcal{O}$ and Eval, in some cases it may not be possible to construct a hybrid oracle even if there are no more intersection queries (in the basic attack, such a hybrid oracle always exists). We overcome these challenges by carefully defining the dependencies between our oracles, and by forcing the algorithm to issue various additional queries other than the queries that appear in the "real" executions (i.e., the computations $\mathcal{R}(x, w^*_{\mathsf{yes}})$ or $\overline{\mathcal{R}}^{\Gamma}(x, w^*_{\mathsf{no}})$) but are still related with these executions. By doing so, we are able to guarantee the consistency of the relevant executions between the hybrid oracle $\widetilde{\Gamma}$, the simulated oracles Γ'_{yes} and Γ'_{no}, and the real oracle Γ.

Finally, the third challenge is the fact that $\mathcal{O}$ is *injective*, which causes the following problem. In our case, we are forced to assume that $(\mathcal{R}^{\Gamma}, \overline{\mathcal{R}}^{\Gamma})$ define a language in $\mathsf{NP}\cap\mathsf{coNP}$ only when $\mathcal{O}$ is an *injective length-increasing* function and not just any arbitrary function as in the basic algorithm (as otherwise our obfuscator may not preserve functionality). Therefore, when constructing the hybrid oracle $\widetilde{\mathcal{O}}$, we must ensure that it is also *injective* in order to reach a contradiction. However, the hybrid oracle $\widetilde{\mathcal{O}}$ might be non-injective when there is some overlap between the images of the true oracle $\mathcal{O}$ and the sampled oracles $\mathcal{O}'_{\mathsf{yes}}$ and $\mathcal{O}'_{\mathsf{no}}$ on elements that are not in Q.

We deal with this challenge by showing that any such an overlap can be used to construct an algorithm that is able to hit a valid image of $\mathcal{O}$ without querying its pre-image before. We then show that the latter can occur only with a small probability. However, due to some subtle technicalities in the proof, we need to make sure that this probability is very small, and we achieve this by a fine tuning of the expansion factor of the oracle $\mathcal{O}$.

1.3 Related Work

Bitansky, Paneth and Wichs [BPW16] and Garg, Pandey and Srinivasan [GPS15] recently showed that indistinguishability obfuscation can be used for constructing a trapdoor permutation family. Their constructions are captured by our framework (as they rely on the obfuscator itself in a black-box manner), but do not imply an $\mathsf{NP}\cap\mathsf{coNP}$-language (this would have contradicted our results). Specifically, whereas any one-way permutation over $\{0,1\}^n$ does imply an $\mathsf{NP}\cap\mathsf{coNP}$-language, their constructions provide a *family* of permutations, and the domains of the permutations are subsets of $\{0,1\}^n$ which depend on the underlying building blocks – and this does not seem to enable defining an $\mathsf{NP}\cap\mathsf{coNP}$-language. The work of Asharov and Segev [AS16] showed that these drawbacks are in fact inherent to constructions of one-way permutations that rely on the obfuscator itself in a black-box manner.

As discussed above, our approach builds upon the work of Impagliazzo and Naor [IN88], which in turn can be viewed as significantly extending Rudich's techniques for showing that one-way functions do not imply one-way permutations in a black-box manner. Specifically, as recently observed by Rosen, Segev, and Shahaf [RSS16], Rudich's proof already generalizes (perhaps somewhat implicitly) to ruling out constructions of unique-TFNP instances (i.e., TFNP instances that are guaranteed to always have a unique solution). In this light, our first generalization of the work of Impagliazzo and Naor (which was described in Section 1.2.2) shows, in particular, that one-way functions do not imply $\mathsf{NP} \cap \mathsf{coNP}$-languages.

1.4 Paper Organization

The remainder of this paper is organized as follows. In Section 2 we introduce the cryptographic primitives under consideration in this paper, as well as some standard notation. In Section 3 we formalize the class of constructions to which our result applies, and present the structure of our proof, which is then provided in Section 4.

2 Preliminaries

In this section we present the notation and basic definitions that are 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. Similarly, for a set $\mathcal{X}$ we denote by $x \leftarrow \mathcal{X}$ the process of sampling a value x from the uniform distribution over $\mathcal{X}$. For an integer $n \in \mathbb{N}$ we denote by $[n]$ the set $\{1, \ldots, n\}$. For a language $L \subset \{0,1\}^*$, we let $\chi_L : \{0,1\}^* \to \{0,1\}$ denote the characteristic function of L, that is, $\chi_L(x) = 1$ if and only if $x \in L$.

2.1 Oracle-Aided Languages and Complexity Classes

We consider the standard notions of languages and complexity classes when naturally generalized to oracle-aided computations. Our definitions follow the standard approach that was introduced in the classic complexity-theory literature for proving separations between complexity classes by considering type-2 languages and complexity classes (see, for example, [BCE+95, CIY97] and the references therein)[1].

An oracle-aided language L defines a set $L^{\mathcal{O}} \subseteq \{0,1\}^*$ for any possible oracle $\mathcal{O} : \{0,1\}^* \to \{0,1\}^*$. This naturally leads to the following standard (non-uniform) definitions of the type-2 complexity classes NP^2 and coNP^2:

- We say that $L \in \mathsf{NP}^2$ if there exists a sequence $\mathcal{R} = \{\mathcal{R}_n\}_{n \in \mathbb{N}}$ of polynomial-size oracle-aided circuits such that for any oracle $\mathcal{O} : \{0,1\}^* \to \{0,1\}^*$ the sequence of circuits $\mathcal{R}^{\mathcal{O}} = \{\mathcal{R}_n^{\mathcal{O}}\}_{n \in \mathbb{N}}$ computes an NP-relation for the language $L^{\mathcal{O}}$. That is, there exists a polynomial $\ell(\cdot)$ such that for any oracle $\mathcal{O}$, for any $n \in \mathbb{N}$, and for any $x \in \{0,1\}^n$ it holds that $x \in L^{\mathcal{O}}$ if and only if there exists a "witness" $w \in \{0,1\}^{\ell(n)}$ such that $\mathcal{R}_n^{\mathcal{O}}(x, w) = 1$.
- We say that $L \in \mathsf{coNP}^2$ if there exists a sequence $\overline{\mathcal{R}} = \{\overline{\mathcal{R}}_n\}_{n \in \mathbb{N}}$ of polynomial-size oracle-aided circuits such that for any oracle $\mathcal{O} : \{0,1\}^* \to \{0,1\}^*$ the sequence of circuits $\overline{\mathcal{R}}^{\mathcal{O}} = \{\overline{\mathcal{R}}_n^{\mathcal{O}}\}_{n \in \mathbb{N}}$ computes an NP-relation for the language $\{0,1\}^* \setminus L^{\mathcal{O}}$. That is, there exists a polynomial $\ell(\cdot)$ such that for any oracle $\mathcal{O}$, for any $n \in \mathbb{N}$, and for any $x \in \{0,1\}^n$ it holds that $x \notin L^{\mathcal{O}}$ if and only if there exists a "witness" $w \in \{0,1\}^{\ell(n)}$ such that $\overline{\mathcal{R}}_n^{\mathcal{O}}(x, w) = 1$.

[1]A type-2 relation is a relation in which one of the arguments can be an oracle, and this can be directly used for defining type-2 languages, reductions, and complexity classes.

2.2 Indistinguishability Obfuscation for Oracle-Aided Circuits

We consider the standard notion of indistinguishability obfuscation [BGI$^+$12, GGH$^+$13] when naturally generalized to oracle-aided circuits (i.e., circuits that may contain oracle gates in addition to standard gates) [AS15, AS16]. We first define the notion of functional equivalence relative to a specific function (provided as an oracle), and then we define the notion of an indistinguishability obfuscation for a class of oracle-aided circuits. In what follows, when considering a class $\mathcal{C} = \{\mathcal{C}_n\}_{n\in\mathbb{N}}$ of oracle-aided circuits, we assume that each $\mathcal{C}_n$ consists of circuits of size at most n.

Definition 2.1. *Let C_0 and C_1 be two oracle-aided circuits, and let f be a function. We say that C_0 and C_1 are* functionally equivalent relative to f*, denoted $C_0^f \equiv C_1^f$, if for any input x it holds that $C_0^f(x) = C_1^f(x)$.*

Definition 2.2. *A probabilistic polynomial-time algorithm $i\mathcal{O}$ is an* indistinguishability obfuscator *relative to an oracle Γ for a class $\mathcal{C} = \{\mathcal{C}_n\}_{n\in\mathbb{N}}$ of oracle-aided circuits if the following conditions are satisfied:*

- **Functionality.** *For all $n \in \mathbb{N}$ and for all $C \in \mathcal{C}_n$ it holds that*

$$\Pr\left[C^\Gamma \equiv \widehat{C}^\Gamma \ : \ \widehat{C} \leftarrow i\mathcal{O}^\Gamma(1^n, C)\right] = 1.$$

- **Indistinguishability.** *For any probabilistic polynomial-time distinguisher $D = (D_1, D_2)$ there exists a negligible function $\mathsf{negl}(\cdot)$ such that*

$$\mathsf{Adv}^{\mathsf{iO}}_{\Gamma,i\mathcal{O},D,\mathcal{C}}(n) \overset{\mathsf{def}}{=} \left|\Pr\left[\mathsf{Exp}^{\mathsf{iO}}_{\Gamma,i\mathcal{O},D,\mathcal{C}}(n) = 1\right] - \frac{1}{2}\right| \leq \mathsf{negl}(n)$$

  for all sufficiently large $n \in \mathbb{N}$, where the random variable $\mathsf{Exp}^{\mathsf{iO}}_{\Gamma,i\mathcal{O},D,\mathcal{C}}(n)$ is defined via the following experiment:

  1. *$b \leftarrow \{0,1\}$.*
  2. *$(C_0, C_1, \mathsf{state}) \leftarrow D_1^\Gamma(1^n)$ where $C_0, C_1 \in \mathcal{C}_n$ and $C_0^\Gamma \equiv C_1^\Gamma$.*
  3. *$\widehat{C} \leftarrow i\mathcal{O}^\Gamma(1^n, C_b)$.*
  4. *$b' \leftarrow D_2^\Gamma(\mathsf{state}, \widehat{C})$.*
  5. *If $b' = b$ then output 1, and otherwise output 0.*

3 The Class of Constructions and Proof Overview

In this section we formalize the class of constructions to which our negative result applies. Then, we formally state our main theorem, and present the structure of our proof (which is provided in Section 4).

3.1 The Class of Constructions

We consider fully black-box constructions of oracle-aided languages in the intersection of NP and coNP from a one-way function f and an indistinguishability obfuscator for all oracle-aided circuits C^f (we refer the reader to Section 1.1 for the importance of considering a one-way function and an indistinguishability obfuscator as two independent building blocks). We now formally define the class of constructions considered in this section, tailoring our definitions to the specific primitives under consideration. We remind the reader that two oracle-aided circuits, C_0 and C_1, are functionally equivalent relative to a function f, denoted $C_0^f \equiv C_1^f$, if for any input x it holds that $C_0^f(x) = C_1^f(x)$ (see Definition 2.1). The following definition is based on those of [AS15, AS16] (which, in turn, are motivated by [Lub96, Gol00, RTV04]).

Definition 3.1. *A fully black-box construction of an* $\mathsf{NP}\cap\mathsf{coNP}$*-language* L *from a one-way function and an indistinguishability obfuscator for the class* $\mathcal{C}$ *of all polynomial-size oracle-aided circuits, consists of a pair of oracle-aided polynomial-time algorithms* $(\mathcal{R},\overline{\mathcal{R}})$*, a polynomial* $p(\cdot)$*, an oracle-aided polynomial-time algorithm* M*, and "security loss" functions* $\epsilon_{M,1}(\cdot)$ *and* $\epsilon_{M,2}(\cdot)$*, such that the following conditions hold:*

- **Correctness:** *For any functions* f *and* $i\mathcal{O}$ *such that* $i\mathcal{O}(C;r)^f \equiv C^f$ *for all* $C \in \mathcal{C}$ *and* $r \in \{0,1\}^*$*, for any* $n \in \mathbb{N}$ *and for any* $x \in \{0,1\}^n$, exactly one *of the following two properties is satisfied:*
  - $x \in L_n^{f,i\mathcal{O}}$*: There exists* $w \in \{0,1\}^{p(n)}$ *such that* $\mathcal{R}^{f,i\mathcal{O}}(x,w) = 1$ *and for any* $w' \in \{0,1\}^{p(n)}$ *it holds that* $\overline{\mathcal{R}}^{f,i\mathcal{O}}(x,w') = 0$*.*
  - $x \notin L_n^{f,i\mathcal{O}}$*: There exists* $w \in \{0,1\}^{p(n)}$ *such that* $\overline{\mathcal{R}}^{f,i\mathcal{O}}(x,w) = 1$ *and for any* $w' \in \{0,1\}^{p(n)}$ *it holds that* $\mathcal{R}^{f,i\mathcal{O}}(x,w') = 0$*.*
- **Black-box proof of security:** *For any functions* f *and* $i\mathcal{O}$ *such that* $i\mathcal{O}(C;r)^f \equiv C^f$ *for all* $C \in \mathcal{C}$ *and* $r \in \{0,1\}^*$*, for any oracle-aided algorithm* $\mathcal{A}$ *that runs in time* $T_{\mathcal{A}}(\cdot)$*, if* $\mathcal{A}^{f,i\mathcal{O}}$ *decides the language* $L_n^{f,i\mathcal{O}}$ *for infinitely many values of* $n \in \mathbb{N}$*, then either*

  $$\Pr\left[M^{\mathcal{A},f,i\mathcal{O}}\left(f\left(x\right)\right) \in f^{-1}(f(x))\right] \geq \epsilon_{M,1}\left(T_{\mathcal{A}}(n)\right)\cdot\epsilon_{M,2}(n)$$

  for infinitely many values of $n \in \mathbb{N}$*, where the probability is taken over the choice of* $x \leftarrow \{0,1\}^n$ *and over the internal randomness of* M*, or*

  $$\left|\Pr\left[\mathsf{Exp}^{\mathsf{iO}}_{(f,i\mathcal{O}),i\mathcal{O},M^{\mathcal{A}},\mathcal{C}}(n) = 1\right] - \frac{1}{2}\right| \geq \epsilon_{M,1}\left(T_{\mathcal{A}}(n)\right)\cdot\epsilon_{M,2}(n)$$

  for infinitely many values of $n \in \mathbb{N}$ *(see Definition 2.2 for the description of the experiment* $\mathsf{Exp}^{\mathsf{iO}}_{(f,i\mathcal{O}),i\mathcal{O},M^{\mathcal{A}},\mathcal{C}}(n)$*).*

Note that, following Asharov and Segev [AS15, AS16], we split the security loss in the above definition to an adversary-dependent security loss (the function $\epsilon_{M,1}(\cdot)$) and an adversary-independent security loss (the function $\epsilon_{M,2}(\cdot)$), as this allows us to capture constructions where one of these losses is super-polynomial whereas the other is polynomial (see, for example, [BPR15, BPW16] as well as many other recent constructions that are based on indistinguishability obfuscation and have a polynomial adversary-related security loss but a super-polynomial adversary-independent security loss).

Equipped with Definition 3.1, we are ready to state out main theorem:

Theorem 3.2. *Let* $(\mathcal{R},\overline{\mathcal{R}},p,M,T_M,\epsilon_{M,1},\epsilon_{M,2})$ *be a fully black-box construction of an* $\mathsf{NP}\cap\mathsf{coNP}$*-language* L *from a one-way function* f *and an indistinguishability obfuscator for the class of all polynomial-size oracle-aided circuits* C^f*. Then, it holds that*

$$\epsilon_{M,1}\left(2^{n^{1/\zeta}}\right)\cdot\epsilon_{M,2}(n) \leq 2^{-n/4}$$

for some constant $\zeta > 1$*. That is, either the adversary-dependent security loss* $\epsilon_{M,1}(\cdot)$ *is at least quasi-polynomial, or the adversary-independent security loss* $\epsilon_{M,2}(\cdot)$ *is exponential.*

Theorem 3.2 rules out standard "polynomial-time polynomial-loss" reductions. More generally, the theorem implies that if the adversary-dependent security loss $\epsilon_{M,1}(\cdot)$ is polynomial (as expected in cryptographic reductions), then the adversary-independent security loss $\epsilon_{M,2}(\cdot)$ must be exponential. This rules out constructions that are based on indistinguishability obfuscation with sub-exponential security (see, for example, [BPR15, BPW16] as well as many other recent constructions that are based on indistinguishability obfuscation and have a polynomial adversary-related security loss but a sub-exponential adversary-independent security loss).

3.2 Proof Overview and the Oracle Γ

Our result is obtained by presenting a distribution over oracles Γ relative to which the following two properties hold: (1) there exists an algorithm that decides languages L^{Γ} in $\mathsf{NP}^{2} \cap \mathsf{coNP}^{2}$, and (2) there exist an exponentially-secure one-way function f and an exponentially-secure indistinguishability obfuscator $i\mathcal{O}$ for the class of all polynomial-size oracle-aided circuits C^{f}. In what follows we describe the oracle Γ, and then explain the structure of our proof.

The oracle Γ. The oracle Γ is identical to that of Asharov and Segev [AS16], while slightly generalized via a parameter a constant $c > 1$ (which we will use as a fixed constant later on, depending on the reduction under consideration). The oracle Γ is a triplet $\left(f, \mathcal{O}, \mathsf{Eval}^{f,\mathcal{O}}\right)$ that is sampled from a distribution, denoted $\mathfrak{S}_c$, as follows:

- **The function $f = \{f_n\}_{n\in\mathbb{N}}$.** For every $n \in \mathbb{N}$, the function f_n is a uniformly chosen function $f_n : \{0,1\}^n \to \{0,1\}^n$.
  Looking ahead, we will prove that f is a one-way function relative to Γ.
- **The functions $\mathcal{O} = \{\mathcal{O}_n\}_{n\in\mathbb{N}}$ and $\mathsf{Eval}^{f,\mathcal{O}} = \{\mathsf{Eval}_n^{f,\mathcal{O}}\}_{n\in\mathbb{N}}$.** For every $n \in \mathbb{N}$ the function $\mathcal{O}_n$ is an injective function $\mathcal{O}_n : \{0,1\}^{2n} \to \{0,1\}^{10n^c}$ chosen uniformly at random. The function $\mathsf{Eval}_n^{f,\mathcal{O}}$ on input $(\widehat{C}, x) \in \{0,1\}^{10n^c} \times \{0,1\}^n$ finds the unique pair $(C, r) \in \{0,1\}^n \times \{0,1\}^n$ such that $\mathcal{O}_n(C, r) = \widehat{C}$, where C is an oracle-aided circuit and r is a string (uniqueness is guaranteed since $\mathcal{O}_n$ is injective). If such a pair exists, it evaluates and outputs $C^f(x)$, and otherwise it outputs $\perp$.
  Looking ahead, we will use $\mathcal{O}$ and Eval for realizing an indistinguishability obfuscator $i\mathcal{O}$ relative to Γ for the class of all polynomial-size oracle-aided circuits C^f.

The structure of our proof. Our proof consists of two parts: (1) showing that relative to Γ (or, the distribution over oracles $\mathfrak{S}_c$), there exists an algorithm that decides any language $L^{\Gamma} \in \mathsf{NP}^{\Gamma} \cap \mathsf{coNP}^{\Gamma}$ with all but exponentially-small probability over the choice of the oracle $\Gamma \leftarrow \mathfrak{S}_c$, and (2) showing that relative to Γ the function f is an *exponentially-secure* one-way function and that the pair $(\mathcal{O}, \mathsf{Eval})$ can be used for implementing an *exponentially-secure* indistinguishability obfuscator for oracle-aided circuits C^f. In what follows, we call an oracle-aided algorithm $\mathcal{A}$ a q-$\mathsf{query\ algorithm}$, for some function $q = q(n)$, if when given any input $x \in \{0,1\}^n$ it $\mathcal{A}$ makes at most $q(n)$ queries to the oracle Γ, and each of its queries to Eval consists of a circuit of size at most $q(n)$.

Part 1: Deciding languages in $\mathsf{NP} \cap \mathsf{coNP}$. We prove that relative to our oracle Γ there exists an algorithm that decides any language $L^{\Gamma} \in \mathsf{NP}^{\Gamma} \cap \mathsf{coNP}^{\Gamma}$ with all but exponentially-small probability over the choice of $\Gamma \leftarrow \mathfrak{S}_c$, by making $\mathsf{poly}(n) \cdot 2^{2n^{1/c}}$ queries to the oracle Γ. Although the algorithm makes a super-polynomial number of queries, this range of parameters suffices for proving our main result. In Section 4 we prove the following theorem:

Theorem 3.3. *Let L be an oracle-aided language, let $\mathcal{R}$ and $\overline{\mathcal{R}}$ be oracle-aided polynomial-time algorithms corresponding to the witness-relation algorithms of L and of its complement $\overline{L}$, respectively, and let $c \geq 2$. If $\mathcal{R}$ and $\overline{\mathcal{R}}$ satisfy the correctness requirement stated in Definition 3.1, then* *there exists a q-query algorithm $\mathcal{A}$ with $q(n) = \mathsf{poly}(n) \cdot 2^{2\cdot n^{1/c}}$ such that*

$$\Pr\left[\mathcal{A}^{\Gamma} \text{ decides } L_n^{\Gamma}\right] \geq 1 - 2^{-5n}$$

for every $n \in \mathbb{N}$, where the probability is taken over the choice of $\Gamma \leftarrow \mathfrak{S}_c$. Moreover, the algorithm $\mathcal{A}$ can be implemented in time $\mathsf{poly}(n) \cdot 2^{2\cdot n^{1/c}}$ given access to a PSPACE-complete oracle.

Part 2: The existence of a one-way function and an indistinguishability obfuscator. The oracle Γ that we consider here is almost identical to the one introduced in [AS16], where the only difference is the increased output length of the function $\mathcal{O}$ (which only makes their proof easier).

Specifically, the one-way function relative to Γ is simply the oracle f. The obfuscator $i\mathcal{O}$ for the class $\mathcal{C}$ of all polynomial-time oracle-aided circuits C^f is defined as follows: For obfuscating an oracle-aided circuit $C \in \{0,1\}^n$ (i.e., we denote by $n = n(C)$ the bit length of C's representation), the obfuscator $i\mathcal{O}$ samples $r \leftarrow \{0,1\}^n$ uniformly at random, computes $\widehat{C} = \mathcal{O}_n(C, r)$, and outputs the circuit $\mathsf{Eval}(\widehat{C}, \cdot)$. That is, the obfuscated circuit consists of a single Eval gate with hardwired input $\widehat{C}$. The following theorem follows directly from [AS16]:

Theorem 3.4. *For any constant $c \geq 1$ and for any oracle-aided $2^{n/4}$-query algorithm $\mathcal{A}$ it hold that*

$$\Pr\left[\mathcal{A}^{\Gamma}(f(x)) \in f^{-1}(f(x))\right] \leq 2^{-n/2}$$

and

$$\left| \Pr\left[\mathsf{Exp}^{\mathsf{iO}}_{\Gamma, i\mathcal{O}, \mathcal{A}, \mathcal{C}}(n) = 1\right] = 1 - \frac{1}{2} \right| \leq 2^{-n/4}$$

for all sufficiently large $n \in \mathbb{N}$, where the probability is taken over the choice of $\Gamma \leftarrow \mathfrak{S}_c$ and internal randomness of $\mathcal{A}$ for both cases, in addition to the choice of $x \leftarrow \{0,1\}^n$ in the former case and to the internal randomness of the challenger in the latter case.

Equipped with Theorems 3.3 and 3.4, in Appendix A we conclude the proof of Theorem 3.2.

4 Deciding $\mathsf{NP}\cap\mathsf{coNP}$ Languages Relative to Γ

In this section we prove Theorem 3.3 by presenting an oracle-aided algorithm for deciding any language satisfying the correctness requirement stated in Definition 3.1. Recall that in Section 1.2 we showed that the approach of Impagliazzo and Naor [IN88] can be generalized from decision trees to arbitrary computations. Here, we show that it can even be generalized relative to our highly-structured oracle.

In Section 4.1 we describe our algorithm, denoted $\mathcal{A}$, together with some preliminary notation that we use throughout the proof. Then, in Section 4.2 we define a "bad" event, denoted $\mathsf{bad}_{\mathcal{A},\Gamma}(x)$, and show that it occurs during an execution of $\mathcal{A}$ with probability at most 2^{-6n}, where the probability is taken over the choice of $\Gamma \leftarrow \mathfrak{S}_c$. In Section 4.3 we show that if this bad event does not occur, then the algorithm $\mathcal{A}$ is always correct on any given input. That is, we prove the following claim:

Claim 4.1. *For every $n \in \mathbb{N}$ and $x \in \{0,1\}^n$ it holds that*

$$\Pr_{\Gamma}\left[\mathcal{A}^{\Gamma}(x) = \chi_{L_n^{\Gamma}}(x) \mid \overline{\mathsf{bad}_{\mathcal{A},\Gamma}(x)}\right] = 1 .$$

This enables us to prove Theorem 3.3 via a simple union bound. That is,

$$\begin{aligned}
&\Pr_{\Gamma}\left[\exists x \in \{0,1\}^n \text{ s.t. } \mathcal{A}^{\Gamma}(x) \neq \chi_{L^{\Gamma}}(x)\right] \\
&\quad \leq \sum_{x \in \{0,1\}^n} \Pr_{\Gamma}\left[\mathcal{A}^{\Gamma}(x) \neq \chi_{L_n^{\Gamma}}(x)\right] \\
&\quad \leq 2^n \cdot \left(\Pr_{\Gamma}\left[\mathcal{A}^{\Gamma}(x) \neq \chi_{L^{\Gamma}}(x) \mid \overline{\mathsf{bad}_{\mathcal{A},\Gamma}(x)}\right] + \Pr_{\Gamma}\left[\mathsf{bad}_{\mathcal{A},\Gamma}(x)\right]\right) \\
&\quad \leq \frac{2^n}{2^{6n}} = 2^{-5n},
\end{aligned}$$

which implies that $\Pr_{\Gamma}\left[\mathcal{A}^{\Gamma} \text{ decides } L_n^{\Gamma}\right] \geq 1 - 2^{-5n}$.

4.1 The Algorithm $\mathcal{A}$

In this section we present the algorithm $\mathcal{A}$ that decides membership of an instance $x \in \{0,1\}^n$. We start with introducing our notation.

Notation. The algorithm $\mathcal{A}$ proceeds in iterations, where in each iteration $\mathcal{A}$ samples two sets of oracle queries $\mathsf{Partial}(\Gamma'_{\mathsf{yes}})$ and $\mathsf{Partial}(\Gamma'_{\mathsf{no}})$. We let Q denote the set of actual queries that are made by $\mathcal{A}$ to the true oracle Γ.

For example, we write $[\mathcal{O}_m(C,r) = \widehat{C}] \in Q$ to denote that Q contains an $\mathcal{O}_m$-query with input (C,r) and output $\widehat{C}$. Likewise, $[f_m(x) = y] \in \mathsf{Partial}(\Gamma')$ denotes that there is some f_m query in $\mathsf{Partial}(\Gamma')$ with input x and output y (for either $\Gamma' = \Gamma'_{\mathsf{yes}}$ or Γ'_{no}). We use the symbol $\star$ to indicate an arbitrary value, for instance $[\mathsf{Eval}(\widehat{C},a) = \star] \in Q$ denotes that $\mathcal{A}$ made an Eval call to Γ on the pair $(\widehat{C},a)$, but we are not interested in the value that was returned by the oracle.

For every query of the form $[\mathsf{Eval}_m(\widehat{C},a) = \perp] \in Q$ for some $a \in \{0,1\}^m$, the algorithm can also conclude that $[\mathsf{Eval}_m(\widehat{C},b) = \perp]$ for every $b \in \{0,1\}^m$. We therefore introduce the set of queries $\mathsf{ExtendedQ}$, which is the set of all queries in Q, and contains in addition the set of queries $\left\{[\mathsf{Eval}_m(\widehat{C},b) = \perp]\right\}_{b \in \{0,1\}^m}$ for every query $[\mathsf{Eval}_m(\widehat{C},a) = \perp] \in Q$.

The set of queries/answers that the algorithm samples. Our algorithm $\mathcal{A}$ samples in each iteration some oracle queries/answers $\mathsf{Partial}(\Gamma'_{\mathsf{yes}}) = (f'_{\mathsf{yes}}, \mathcal{O}'_{\mathsf{yes}}, \mathsf{Eval}'_{\mathsf{yes}})$ and $\mathsf{Partial}(\Gamma'_{\mathsf{no}}) = (f'_{\mathsf{no}}, \mathcal{O}'_{\mathsf{no}}, \mathsf{Eval}'_{\mathsf{no}})$ that are consistent (i.e., return the same results) with the list of queries $\mathsf{ExtendedQ}$, and in particular with the set of actual queries Q. Note that even though the size of $\mathsf{ExtendedQ}$ might be exponential in n, the sizes of the sets $\mathsf{Partial}(\Gamma'_{\mathsf{yes}})$ and $\mathsf{Partial}(\Gamma'_{\mathsf{no}})$ depend on the number of oracle-gates in $\mathcal{R}$ and $\overline{\mathcal{R}}$, and are polynomial in n.

Since the oracles $(f, \mathcal{O}, \mathsf{Eval})$ have some dependencies, we want that these dependencies will appear explicitly in the set of queries/answers that the algorithm samples (looking ahead, by doing so, we will be able to construct a hybrid oracle $\widetilde{\Gamma}$). Formally, we say that the set of queries/answers $\mathsf{Partial}(\Gamma')$ (for either $\mathsf{Partial}(\Gamma'_{\mathsf{yes}})$ or $\mathsf{Partial}(\Gamma'_{\mathsf{no}})$) is consistent if the following holds:

Definition 4.2 (Consistent oracle queries/answers)**.** *Let $\mathsf{Partial}(\Gamma') = (f', \mathcal{O}', \mathsf{Eval}')$ be a set of queries/answers. We say it is* consistent *if for every $m \in \mathbb{N}$ it holds that:*

1. *For every query $\left[\mathsf{Eval}_m(\widehat{C},a) = \beta\right] \in \mathsf{Eval}'$ with with $|\widehat{C}| = 10m^c$, $|a| = m$ and $\beta \neq \perp$, there exists a query $\left[\mathcal{O}_m(\star) = \widehat{C}\right] \in \mathcal{O}'$.*

2. *For every query* $\left[\mathsf{Eval}_m(\widehat{C},a)=\beta\right]\in\mathsf{Eval}'$ *with* $|\widehat{C}|=10m^c$, $|a|=m$ *and* $\beta\neq\perp$, *let* $\left[\mathcal{O}_m(C,r)=\widehat{C}\right]\in\mathcal{O}'$ *that is guaranteed to exist by the previous requirement, for some* $C,r\in\{0,1\}^m$. *Then, the oracle* f' *contains also queries/answers sufficient for the evaluation of* $C^{f'}(a)$, *and the value of this evaluation is indeed* β.
3. *For every query* $[\mathcal{O}'_m(\star)=\widehat{C}]\in\mathcal{O}'$ *there exists a query* $[\mathsf{Eval}'_m(\widehat{C},\alpha)=\beta]\in\mathsf{Eval}'$, *for some arbitrary* $\alpha\in\{0,1\}^m$.

While the first two requirements are quite natural and come to model that there is consistency in $\mathsf{Partial}(\Gamma')$, the third requirement has somewhat different role. Whenever the algorithm $\mathcal{A}$ samples some image of $\mathcal{O}$, i.e., whenever it samples a query $[\mathcal{O}'_m(\star)=\widehat{C}]\in\mathcal{O}'$, we are interested to learn whether $\widehat{C}$ is a valid image with respect to the true oracle $\mathcal{O}$. This is because the oracle $\mathcal{O}$ is injective, and any collision between the oracles $\mathcal{O}$ and $\mathcal{O}'$ disallows us from constructing the hybrid oracle $\widehat{\Gamma}$. We therefore require to sample one addition query $[\mathsf{Eval}'(\widehat{C},\alpha)]$ for some arbitrary value α. When $\mathcal{A}$ queries the true oracle Γ at the end of the iteration, it also learns whether $\widehat{C}$ is a valid image of $\mathcal{O}$ or not.

Augmented oracle queries. Assume that $x\in L^\Gamma$, and let w^*_{yes} be the lexicographically smallest witness such that $\mathcal{R}^\Gamma(x,w^*_{\mathsf{yes}})=1$. We now consider the set of oracle queries that are associated with the execution $\mathcal{R}^\Gamma(x,w^*_{\mathsf{yes}})=1$. This set of queries contain some additional queries that do not necessarily appear in that execution, but are still associated with it. Let $\mathsf{Real}(\mathcal{R},\Gamma,x,w^*_{\mathsf{yes}})$ denote the set of actual queries to Γ in the evaluation of $\mathcal{R}^\Gamma(x,w^*_{\mathsf{yes}})=1$. Then:

Definition 4.3 (Augmented oracle queries, case $x\in L^\Gamma$)**.** *Assume that* $x\in L^\Gamma$ *and let* w^*_{yes} *be the lexicographically smallest witness such that* $\mathcal{R}^\Gamma(x,w^*_{\mathsf{yes}})=1$. *The set of extended queries, denoted* $\mathsf{AugQ}_{\mathsf{yes}}(\mathcal{R},\Gamma,x,w^*_{\mathsf{yes}})$, *consists of the queries in* $\mathsf{Real}(\mathcal{R},\Gamma,x,w^*_{\mathsf{yes}})$, *and in addition:*

For every query $[\mathsf{Eval}_m(\widehat{C},a)=\beta]\in\mathsf{Real}(\mathcal{R},\Gamma,x,w^*_{\mathsf{yes}})$ *with* $|\widehat{C}|=10m^c$, $|a|=m$ *and* $\beta\neq\perp$, *let* $C,r\in\{0,1\}^m$ *be the unique pair such that* $\mathcal{O}_m(C,r)=\widehat{C}$.

1. *The set* $\mathsf{AugQ}_{\mathsf{yes}}(\mathcal{R},\Gamma,x,w^*_{\mathsf{yes}})$ *contains also the* $\mathcal{O}_m$*-query* $[\mathcal{O}_m(C,r)=\widehat{C}]$.
2. *The set* $\mathsf{AugQ}_{\mathsf{yes}}(\mathcal{R},\Gamma,x,w^*_{\mathsf{yes}})$ *contains all the* f*-queries/answers sufficient to for the evaluation of* $C^f(a)$.

Similarly to the above, we define the set of extended oracle queries for the case where $x\notin L^\Gamma$. Let w^*_{no} be the lexicographically smallest witness such that $\overline{\mathcal{R}}^\Gamma(x,w^*_{\mathsf{no}})=1$, and let $\mathsf{Real}(\overline{\mathcal{R}},\Gamma,x,w^*_{\mathsf{no}})$ be the set of actual queries to Γ in this evaluation. Let $\mathsf{AugQ}_{\mathsf{no}}(\overline{\mathcal{R}},\Gamma,x,w^*_{\mathsf{no}})$ be the equivalent set of queries, similarly to Definition 4.3.

Let $\ell(n)$ be un upper bound on the number of oracle-gates in the circuits $\mathcal{R}$ and $\overline{\mathcal{R}}$. Let $\widehat{\ell}=\max\{|\mathsf{AugQ}_{\mathsf{yes}}(\mathcal{R},\Gamma,x,w^*_{\mathsf{yes}})|,|\mathsf{AugQ}_{\mathsf{no}}(\overline{\mathcal{R}},\Gamma,x,w^*_{\mathsf{no}})|\}$. It is easy to see that $\widehat{\ell}(n)\leq(2n+1)\cdot\ell(n)\leq\ell(n)^3$.

We are now ready to describe the algorithm $\mathcal{A}$.

The algorithm $\mathcal{A}$. The algorithm $\mathcal{A}$ has oracle access to Γ, and proceeds as follows given as input an instance $x\in\{0,1\}^n$:

1. *Initialize an empty list* Q *of oracle queries/answers to* Γ *(looking ahead, the list* Q *will always be consistent with the true oracle* Γ*.). Define* $\mathsf{ExtendedQ}$ *accordingly.*

2. **Querying f_m and $\mathcal{O}_m$ for small m.** *Let $t = (\log 8\widehat{\ell})^{1/c} + n^{1/c}$. The algorithm $\mathcal{A}$ queries the oracle f_m on all inputs $|x| = m$ for all $m \leq t$. It queries $\mathcal{O}_m(C, r)$ for all $|C|=|r|=m \leq t$. Denote this set of queries by Q^*. Add all these queries to Q, $\mathsf{ExtendedQ}$.*
3. *Run the following for $\widehat{\ell} + 1$ iterations:*
   (a) **Simulating YES:**
      i. *$\mathcal{A}$ finds a set $\mathsf{Partial}(\Gamma'_{\mathsf{yes}}) = (f'_{\mathsf{yes}}, \mathcal{O}'_{\mathsf{yes}}, \mathsf{Eval}'_{\mathsf{yes}})$ of oracle queries/answers that is consistent with the list of queries/answers $\mathsf{ExtendedQ}$, and a witness w'_{yes} such that $\mathcal{R}^{\mathsf{Partial}(\Gamma'_{\mathsf{yes}})}(x, w'_{\mathsf{yes}}) = 1$.*
      ii. *If there does not exist a pair $(\mathsf{Partial}(\Gamma'_{\mathsf{yes}}), w'_{\mathsf{yes}})$ as above, then it halts and outputs* **NO** *(i.e., $x \notin L^{\Gamma}$).*
   (b) **Simulating NO:**
      i. *$\mathcal{A}$ finds an oracle $\mathsf{Partial}(\Gamma'_{\mathsf{no}}) = (f'_{\mathsf{no}}, \mathcal{O}'_{\mathsf{no}}, \mathsf{Eval}'_{\mathsf{no}})$ of oracle queries/answers that is consistent with the list of queries/answers $\mathsf{ExtendedQ}$, and a witness w'_{no} such that $\overline{\mathcal{R}}^{\mathsf{Partial}(\Gamma'_{\mathsf{no}})}(x, w'_{\mathsf{no}}) = 1$.*
      ii. *If there does not exist a pair $(\mathsf{Partial}(\Gamma'_{\mathsf{no}}), w'_{\mathsf{no}})$ as above, then $\mathcal{A}$ halts and outputs* **YES** *(i.e., $x \in L^{\Gamma}$).*
   (c) **Update:** *$\mathcal{A}$ queries Γ with all the queries in $\mathsf{Partial}(\Gamma'_{\mathsf{yes}})$ and $\mathsf{Partial}(\Gamma'_{\mathsf{no}})$ that are not in Q, and updates the list Q and $\mathsf{ExtendedQ}$ accordingly*[2].
   Moreover, for every $[\mathsf{Eval}_m(\widehat{C}, a) = \perp] \in Q$ for some $m \in \mathbb{N}$, it adds to $\mathsf{ExtendedQ}$ the set of queries $\left\{[\mathsf{Eval}(\widehat{C}, b) = \perp]\right\}_{b \in \{0,1\}^m}$.
4. *In case the algorithm has not halted yet, it outputs $\perp$.*

In the initial phase, $\mathcal{A}$ queries at most $2 \cdot 2^t$ queries of f_m and $2 \cdot 2^{2t}$ queries of $\mathcal{O}_m$. That is, at most $4 \cdot 2^{2t}$ queries, which are no more than $\mathsf{poly}(n) \cdot 2^{2n^{1/c}}$ queries. In addition, in each iteration it makes at most $2\widehat{\ell}$ queries to Γ, and runs for $\widehat{\ell} + 1$ iterations. Recall that $\widehat{\ell}(n) \leq \ell(n)^3$, and thus $\mathcal{A}$ makes no more than $\ell(n)^{10}$ queries in all iterations. We conclude that $\mathcal{A}$ makes at most $\mathsf{poly}'(n) \cdot 2^{2n^{1/c}}$ queries, for some polynomial $\mathsf{poly}'(n)$.

4.2 The Event $\mathsf{bad}_{\mathcal{A},\Gamma}$

In this section we define an event which may cause $\mathcal{A}$ to fail on a given instance $x \in \{0,1\}^n$, and bound its probability. Before we define the event $\mathsf{bad}_{\mathcal{A},\Gamma}(x)$, we start with the definition of the event Hit_n, an event that considers an execution of $M^{\Gamma}(1^n)$ for some arbitrary algorithm M and an oracle $\Gamma \leftarrow \mathfrak{S}_c$.

Definition 4.4. *For any oracle-aided algorithm M, consider the following event Hit_n that may occur during an execution of $M^{\Gamma}(1^n)$: The algorithm outputs an output $\beta \in \{0,1\}^{10n^c}$ for which there exists an input $\alpha \in \{0,1\}^n$ such that $\mathcal{O}_n(\alpha) = \beta$, yet β was not an output of a previous $\mathcal{O}$-query.*

We show:

Claim 4.5. *For any oracle-aided algorithm $M^{\Gamma}(1^n)$ making at most $q \leq 2^{n/2}$ queries, any n, f and $\mathcal{O}_{-n} = \{\mathcal{O}_m\}_{m \in \mathbb{N}, m \neq n}$,*

$$\Pr_{\mathcal{O}_n}[\mathsf{Hit}_n] \leq \frac{1}{2^{6n^c}} .$$

[2]Note that the sets $\mathsf{Partial}(\Gamma'_{\mathsf{yes}})$ and $\mathsf{Partial}(\Gamma'_{\mathsf{no}})$ contain only the necessary queries in the evaluations $\mathcal{R}^{\mathsf{Partial}(\Gamma'_{\mathsf{yes}})}(x, w'_{\mathsf{yes}}) = 1$ and $\overline{\mathcal{R}}^{\mathsf{Partial}(\Gamma'_{\mathsf{no}})}(x, w'_{\mathsf{no}}) = 1$ and therefore are of polynomial-size.

Proof. Fix M, n, f and $\mathcal{O}_{-n}$. The input space of $\mathcal{O}_n$ is 2^{2n}, whereas its output space is 2^{10n^c}. Since $\mathcal{O}_n$ is chosen uniformly at random, there are at most 2^{2n} elements in the range of $\mathcal{O}_n$, and these are distributed uniformly in a space of size 2^{10n^c}. Any query to $\mathcal{O}_n$ reveals one point in the range, but does not reveal any information about the other points. Therefore, the oracle queries do not give significant information regarding the range of $\mathcal{O}_n$, and an algorithm cannot hit points in the range without previous queries of $\mathcal{O}_n$.

Formally, we follow the computation of $M^{\Gamma}(1^n)$ and choose the function $\mathcal{O}_n$ during this computation. We store a table $T(\mathcal{O})$ of queries/answers to $\mathcal{O}$. With each $\mathcal{O}_n$-query α, we look for a pair (α, β) in $T(\mathcal{O})$ for some $\beta \in \{0,1\}^{10n^c}$. If such a pair exists, we return β. Otherwise, we choose a value $\gamma \in \{0,1\}^{10n^c}$ uniformly at random under the constraint that it avoids all previous images in $T(\mathcal{O})$ (since $\mathcal{O}$ is injective), and add the pair (α, γ) to the table $T(\mathcal{O})$. When M halts, there are at most q entries in the table $T(\mathcal{O})$. Let y denote the output of M. We then continue to choose the function $\mathcal{O}$ such that the function is injective. All the other choices of the function $\mathcal{O}$ are independent of the value y, and therefore, the probability that one of these $2^{2n} - q$ points hit the fixed point y is:

$$\Pr_{\mathcal{O}_n}[\mathsf{Hit}_n] = \sum_{i=0}^{2^{2n}-q-1} \frac{1}{2^{10n^c} - (q+i)} \leq \frac{2^{2n}}{2^{10n^c} - 2^{2n}} \leq \frac{1}{2^{6n^c}} \ .$$

■

The event $\mathsf{bad}_{\mathcal{A},\Gamma}(x)$. We consider the following event, denoted as $\mathsf{bad}_{\mathcal{A},\Gamma}(x)$.

Definition 4.6. *Let $x \in \{0,1\}^n$. If $x \in L^{\Gamma}$, then let w^*_{yes} be the lexicographically smallest witness such that $\mathcal{R}^{\Gamma}(x, w^*_{\mathsf{yes}}) = 1$, and let $\mathsf{RealQ}(\Pi, \Gamma, \alpha, x^*) \stackrel{\text{def}}{=} \mathsf{AugQ}_{\mathsf{yes}}(\mathcal{R}, \Gamma, x, w^*_{\mathsf{yes}})$. Otherwise (i.e., $x \notin L^{\Gamma}$), let w^*_{no} be the lexicographically smallest witness such that $\overline{\mathcal{R}}^{\Gamma}(x, w^*_{\mathsf{no}}) = 1$, and let $\mathsf{RealQ}(\Pi, \Gamma, \alpha, x^*) \stackrel{\text{def}}{=} \mathsf{AugQ}_{\mathsf{no}}(\overline{\mathcal{R}}, \Gamma, x, w^*_{\mathsf{no}})$.*

Consider the following events that may occur with respect to the sampled oracles $\mathsf{Partial}(\Gamma'_{\mathsf{no}})$, $\mathsf{Partial}(\Gamma'_{\mathsf{yes}})$ in some iteration i (for $i = 1\ldots, L$) and some $m \in \mathbb{N}$:

The event $\mathsf{bad}^{(1)}_{\mathcal{A},\Gamma,i,m}(x)$: *There exist values $\alpha, \beta \in \{0,1\}^{2m}$ such that $\alpha \neq \beta$, and there exists $\gamma \in \{0,1\}^{10m^c}$ for which $[\mathcal{O}'_m(\alpha) = \gamma] \in \mathsf{Partial}(\Gamma'_{\mathsf{no}}) \cup \mathsf{Partial}(\Gamma'_{\mathsf{yes}}) \setminus Q$ and $[\mathcal{O}_m(\beta) = \gamma] \in \mathsf{RealQ}(\Pi, \Gamma, \alpha, x^*)$.*

The event $\mathsf{bad}^{(2)}_{\mathcal{A},\Gamma,i,m}(x)$: *There exists a query of the form $[\mathsf{Eval}'_m(\widehat{C}, a) = \perp] \in \mathsf{Partial}(\Gamma'_{\mathsf{no}}) \cup \mathsf{Partial}(\Gamma'_{\mathsf{yes}}) \setminus Q$, for some $\widehat{C} \in \{0,1\}^{10m^c}$ and $a \in \{0,1\}^m$, but $[\mathcal{O}_m(C, r) = \widehat{C}] \in \mathsf{RealQ}(\Pi, \Gamma, \alpha, x^*)$ for some $C, r \in \{0,1\}^m$.*

We let $\mathsf{bad}_{\mathcal{A},\Gamma}(x) \stackrel{\text{def}}{=} \bigvee_{i\in[L], m\in\mathbb{N}, k\in[2]} \mathsf{bad}^{(k)}_{\mathcal{A},\Gamma,i,m}(x)$.

We show that:

Claim 4.7. *For every $x \in \{0,1\}^n$ it holds that:*

$$\Pr_{\Gamma}[\ \mathsf{bad}_{\mathcal{A},\Gamma}(x)\ ] \leq 2^{-6n} \ .$$

Proof. Fix $x \in \{0,1\}^n$. We consider each one of the internal events of $\mathsf{bad}_{\mathcal{A},\Gamma}(x)$ (Definition 4.6):

- **The event $\mathsf{bad}^{(1)}_{\mathcal{A},\Gamma,i,m}(x)$.** We claim that $\mathsf{bad}^{(1)}_{\mathcal{A},\Gamma,i,m}(x)$ implies the event Hit_m. In particular, since x is fixed we can consider an algorithm $M^{\Gamma}(1^n)$ that receives no input, and invokes $\mathcal{A}^{\Gamma}(x)$. When $\mathcal{A}$ reaches iteration i, we look at the set of simulated queries $[\mathcal{O}'_m(\alpha) = \gamma]$ and consider the values γ. If indeed there exists a query $[\mathcal{O}'_m(\alpha) = \gamma] \in \mathsf{Partial}(\Gamma'_{\mathsf{no}}) \cup \mathsf{Partial}(\Gamma'_{\mathsf{yes}}) \setminus Q$ and also $[\mathcal{O}_m(\beta) = \gamma] \in \mathsf{RealQ}(\Pi, \Gamma, \alpha, x^*)$, we hit a valid output of $\mathcal{O}$ without querying it before. Clearly, there are no more than $2\widehat{\ell}$ queries in $\mathsf{Partial}(\Gamma'_{\mathsf{no}}) \cup \mathsf{Partial}(\Gamma'_{\mathsf{yes}})$, and from Claim 4.5 we conclude that:
$$\Pr_{\Gamma}\left[\mathsf{bad}^{(1)}_{\mathcal{A},\Gamma,i,m}(x)\right] \leq \frac{2\widehat{\ell}}{2^{6m^c}}$$
- **The event $\mathsf{bad}^{(2)}_{\mathcal{A},\Gamma,i,m}(x)$.** We claim that this event also implies the event Hit_m. As previously, in case there exists a pair $(C, r) \in \{0,1\}^{2m}$ for which $[\mathcal{O}_m(C, r) = \widehat{C}] \in \mathsf{RealQ}(\Pi, \Gamma, \alpha, x^*)$, we managed to hit and output of $\mathcal{O}$ without querying it beforehand. We conclude:
$$\Pr_{\Gamma}\left[\mathsf{bad}^{(2)}_{\mathcal{A},\Gamma,i,m}(x)\right] \leq \frac{2\widehat{\ell}}{2^{6m^c}}$$

Bounding the probability of $\mathsf{bad}_{\mathcal{A},\Gamma}(x)$. By construction, Q^* contains all possible $\mathcal{O}_m$-queries for every $m \leq t$, and therefore the events $\mathsf{bad}^{(k)}_{\mathcal{A},\Gamma,i,m}(x)$ cannot occur for $m \leq t$ (for all i, k). We therefore conclude that for every $t \geq (\log 8\widehat{\ell})^{1/c} + n^{1/c}$:

$$\begin{aligned}
\Pr_{\Gamma}[\mathsf{bad}_{\mathcal{A},\Gamma}(x)] &\leq \Pr\left[\bigvee_{i\in[\widehat{\ell}],m\in\mathbb{N},k\in[2]} \mathsf{bad}^{(k)}_{\mathcal{A},\Gamma,i,m}(x)\right] \\
&\leq \sum_{i=1}^{\widehat{\ell}}\sum_{m=t}^{\infty}\sum_{k=1}^{2}\frac{2\widehat{\ell}}{2^{6m^c}} = \sum_{m=t}^{\infty}\frac{4\widehat{\ell}^2}{2^{6m^c}} \leq 2\cdot\frac{4\widehat{\ell}^2}{2^{6t^c}} \\
&\leq \frac{8\cdot\widehat{\ell}^2}{2^{(\log 8\widehat{\ell})^6}2^{6n}} = \frac{8\widehat{\ell}^2}{(8\widehat{\ell})^6\cdot 2^{6n}} \leq 2^{-6n} .
\end{aligned}$$

■

4.3 Deciding Membership

We proceed to show that $\mathcal{A}$ always succeeds to decide L_n^{Γ}, assuming that the event $\mathsf{bad}_{\mathcal{A},\Gamma}(x)$ does not occur. We have:

Claim 4.8. *For every $x \in \{0,1\}^n$*
$$\Pr_{\Gamma}\left[\mathcal{A}^{\Gamma}(x) = \chi_{L_n^{\Gamma}}(x) \mid \overline{\mathsf{bad}_{\mathcal{A},\Gamma}(x)}\right] = 1 .$$

Proof. Fix Γ and $x \in \{0,1\}^n$. We now show that if $\mathsf{bad}_{\mathcal{A},\Gamma}(x)$ does not occur, then $\mathcal{A}$ outputs $\chi_{L_n^{\Gamma}}(x)$, that is, if $x \in L_n^{\Gamma}$ it outputs **YES**, whereas if $x \notin L_n^{\Gamma}$ it outputs **NO**.

Analyzing the case where $x \in L^{\Gamma}$.

Claim 4.9. *Assume that $x \in L^{\Gamma}$, let w^*_{yes} be as above, and assume that the event $\mathsf{bad}_{\mathcal{A},\Gamma}(x)$ does not occur. Then, $\mathcal{A}$ never outputs* **NO** *in Step 3(a)ii, and in each iteration of Step 3, exactly one of the following occurs:*

1. *$\mathcal{A}$ halts and outputs* **YES** *in Step 3(b)ii.*
2. *During the update phase, $\mathcal{A}$ adds to* $\mathsf{ExtendedQ}$ *at least one of the queries that appear in* $\mathsf{AugQ}_{\mathsf{yes}}(\mathcal{R}, \Gamma, x, w^*_{\mathsf{yes}})$.

Proof. First, since the pair $(\Gamma, w^*_{\mathsf{yes}})$ exists, and since Q and $\mathsf{ExtendedQ}$ are always consistent with the true oracle Γ, the algorithm $\mathcal{A}$ never halts and outputs **NO** in Step 3(a)ii. In particular, it can always set $(\mathsf{Partial}(\Gamma'_{\mathsf{yes}}), w'_{\mathsf{yes}}) = (\Gamma, w^*_{\mathsf{yes}})$.

Assume that neither one of the above conditions holds. Then, we show that there exists an hybrid oracle $\widetilde{\Gamma} = (\widetilde{f}, \widetilde{\mathcal{O}}, \widetilde{\mathsf{Eval}})$ that relative to it, the following contradicting facts hold:

- The hybrid oracle $\widetilde{\Gamma}$ behaves identically to the oracle Γ in the evaluation of $\mathcal{R}^{\Gamma}(x, w^*_{\mathsf{yes}}) = 1$.
- The hybrid oracle $\widetilde{\Gamma}$ behaves identically to the partial oracle $\mathsf{Partial}(\Gamma'_{\mathsf{no}})$ in the evaluation of $\overline{\mathcal{R}}^{\mathsf{Partial}(\Gamma'_{\mathsf{no}})}(x, w'_{\mathsf{no}}) = 1$.

That is, relative to the oracle $\widetilde{\Gamma}$, it holds that $\mathcal{R}^{\widetilde{\Gamma}}(x, w^*_{\mathsf{yes}}) = 1$ and $\overline{\mathcal{R}}^{\widetilde{\Gamma}}(x, w'_{\mathsf{no}}) = 1$, and therefore there exists a witness w^*_{yes} for membership $x \in L^{\widetilde{\Gamma}}$ and a witness w'_{no} for non-membership $x \notin L^{\widetilde{\Gamma}}$, in contradiction.

We construct the hybrid oracle $\widetilde{\Gamma} = (\widetilde{f}, \widetilde{\mathcal{O}}, \widetilde{\mathsf{Eval}})$ using both $\Gamma = (f, \mathcal{O}, \mathsf{Eval})$ and $\mathsf{Partial}(\Gamma'_{\mathsf{no}}) = (f'_{\mathsf{no}}, \mathcal{O}'_{\mathsf{no}}, \mathsf{Eval}'_{\mathsf{no}})$, as follows. For sake of cleanliness, in the following we omit the no subscript and just write $\mathsf{Partial}(\Gamma'_{\mathsf{no}}) = (f', \mathcal{O}'\mathsf{Eval}')$. Moreover, we let $\mathsf{RealQ}(\Pi, \Gamma, \alpha, x^*) = \mathsf{AugQ}_{\mathsf{yes}}(\mathcal{R}, \Gamma, x, w^*_{\mathsf{yes}})$. Note that since condition 2 in the claim does not hold, we have that $\mathsf{RealQ}(\Pi, \Gamma, \alpha, x^*) \cap \mathsf{Partial}(\Gamma'_{\mathsf{no}}) \subseteq \mathsf{ExtendedQ}$ (i.e., $\mathcal{A}$ has learned all the queries that are in the intersection of these two evaluations.)

- **The oracle $\widetilde{f}$.** For every $m \leq t$, the set of queries Q^* contains all the functions $\{f_m\}_{m \leq t}$ and thus agrees completely with f (i.e., also with f'). We therefore set $\widetilde{f}_m = f_m$.
  For every $m > t$, we define the function $\widetilde{f}_m$ as follows. For every x such that $[f_m(x) = y'] \in \mathsf{RealQ}(\Pi, \Gamma, \alpha, x^*)$, we set $\widetilde{f}_m(x) = y'$. For every $[f_m(x) = y] \in \mathsf{Partial}(\Gamma'_{\mathsf{no}})$, we set $\widetilde{f}_m(x) = y$. Since $\mathsf{RealQ}(\Pi, \Gamma, \alpha, x^*) \cap \mathsf{Partial}(\Gamma'_{\mathsf{no}}) \subseteq \mathsf{ExtendedQ}$, we have that there is no contradiction, i.e, there are no input x and outputs y, y' such that $y \neq y'$ and $[f_m(x) = y'] \in \mathsf{Partial}(\Gamma'_{\mathsf{no}})$ and $[f_m(x) = y] \in \mathsf{RealQ}(\Pi, \Gamma, \alpha, x^*)$. For any other value $x \notin \mathsf{RealQ}(\Pi, \Gamma, \alpha, x^*) \cup \mathsf{Partial}(\Gamma'_{\mathsf{no}})$, we set $\widetilde{f}_m(x) = 0^m$.
  Before we continue to define the oracle $\widetilde{\mathcal{O}}$, we first define some set of output values that $\widetilde{\mathcal{O}}$ will have to avoid. For every $m > t$, we define the set $\mathsf{avoid}\text{-}\mathcal{O}_m$ as
  $$\mathsf{avoid}\text{-}\mathcal{O}_m = \left\{ \widehat{C} \in \{0,1\}^{10m^c} \;\middle|\; \exists\, [\mathsf{Eval}_m(\widehat{C}, \star) = \star] \in \begin{array}{c} \mathsf{RealQ}(\Pi, \Gamma, \alpha, x^*) \\ \cup\ \mathsf{Partial}(\Gamma'_{\mathsf{no}}) \end{array} \right\} .$$
- **The oracle $\widetilde{\mathcal{O}}$.** The oracle is already defined for every $m \leq t$. For every $m > t$, we define the function $\widetilde{\mathcal{O}}_m$ as follows. For every $[\mathcal{O}_m(x) = y] \in \mathsf{RealQ}(\Pi, \Gamma, \alpha, x^*)$, we set $\widetilde{\mathcal{O}}_m(x) = y$. Likewise, for every $[\mathcal{O}'_m(x) = y] \in \mathsf{Partial}(\Gamma'_{\mathsf{no}})$, we set $\widetilde{\mathcal{O}}_m(x) = y$. Since $\mathsf{RealQ}(\Pi, \Gamma, \alpha, x^*) \cap \mathsf{Partial}(\Gamma'_{\mathsf{no}}) \subseteq Q$, we have that there is no contradiction, that is, for any input x that has already been defined, it holds that $\mathcal{O}_m(x) = \mathcal{O}'_m(x)$.
  Moreover, since $\mathsf{bad}_{\mathcal{A},\Gamma}(x)$ does not occur, it holds that the partially defined function $\widetilde{\mathcal{O}}_m$ is injective. That is, there do not exist two inputs $\alpha \neq \beta \in \{0,1\}^m$ for which $\widetilde{\mathcal{O}}_m(\alpha) = \widetilde{\mathcal{O}}_m(\beta)$. Otherwise, either $\mathcal{O}'_m$ is not injective, $\mathcal{O}_m$ is not injective, or the event $\mathsf{bad}_{\mathcal{A},\Gamma}(x)$ occurs (the event $\mathsf{bad}^{(1)}_{i,m}(x)$ in Definition 4.6: there exist $\alpha, \beta \in \{0,1\}^{2m}$ such that $\alpha \neq \beta$, and there exists $\gamma \in \{0,1\}^{10m^c}$ for which $[\mathcal{O}'_m(\alpha) = \gamma] \in \mathsf{Partial}(\Gamma'_{\mathsf{no}})$ and $[\mathcal{O}_m(\beta) = \gamma] \in \mathsf{RealQ}(\Pi, \Gamma, \alpha, x^*)$).
  We then complete the definition of $\widetilde{\mathcal{O}}_m$ arbitrarily, under the restriction that $\widetilde{\mathcal{O}}_m$ is injective and avoids the image $\mathsf{avoid}\text{-}\mathcal{O}_m$. This is always possible since the possible space of $\mathcal{O}$ is of size

2^{10n^c}, there are 2^{2n} points that should be defined, and only polynomially many points in the image are already defined.

- **The oracle $\widetilde{\mathsf{Eval}}$.** We define the oracle $\widetilde{\mathsf{Eval}}$ using the oracles $\widetilde{f}$ and $\widetilde{\mathcal{O}}$ exactly as the true oracle Eval is defined using the true oracles f and $\mathcal{O}$. We now show that $\widetilde{\mathsf{Eval}}$ is consistent with $\mathsf{RealQ}(\Pi, \Gamma, \alpha, x^*)$ and $\mathsf{Partial}(\Gamma'_{\mathsf{no}})$. That is, that every query $[\mathsf{Eval}_m(\star, \star)] \in \mathsf{AugQ}(\Pi, \Gamma, x^*) \cup \mathsf{Partial}(\Gamma')$ has the same answer with $\widetilde{\mathsf{Eval}}$, and therefore $\mathcal{R}^{\widetilde{\Gamma}}(x, w^*_{\mathsf{yes}}) = \mathcal{R}^{\Gamma}(x, w^*_{\mathsf{yes}}) = 1$ and $\overline{\mathcal{R}}^{\widetilde{\Gamma}}(x, w'_{\mathsf{no}}) = \mathcal{R}^{\mathsf{Partial}(\Gamma'_{\mathsf{no}})}(x, w'_{\mathsf{no}}) = 1$. We have:
  1. Assume that there exists $[\mathsf{Eval}'(\widehat{C}, a) = \beta] \in \mathsf{Partial}(\Gamma'_{\mathsf{no}})$ for some $\beta \neq \perp$. We show that $\widetilde{\mathsf{Eval}}(\widehat{C}, a) = \beta$ as well. Since the oracle $\mathsf{Partial}(\Gamma'_{\mathsf{no}}) = (f', \mathcal{O}', \mathsf{Eval}')$ is consistent (recall Definition 4.2), then there exists a query $\left[\mathcal{O}_m(C, r) = \widehat{C}\right] \in \mathsf{Partial}(\Gamma'_{\mathsf{no}})$ and f' contains all the necessary queries/answers for the evaluation of $C^{f'}(a)$, and it also holds that $C^{f'}(a) = \beta$. However, since any $(f', \mathcal{O}')$-queries in $\mathsf{Partial}(\Gamma'_{\mathsf{no}})$ have the exact same answer with $(\widetilde{f}, \widetilde{\mathcal{O}})$, it holds that $C^{\widetilde{f}}(a) = \beta$ and $\widetilde{\mathcal{O}}(C, r) = \widehat{C}$, and so, from the definition of $\widetilde{\mathsf{Eval}}$ it holds that $\widetilde{\mathsf{Eval}}(\widehat{C}, a) = \beta$ as well.
  2. Assume that there exists $[\mathsf{Eval}(\widehat{C}, a) = \beta] \in \mathsf{RealQ}(\Pi, \Gamma, \alpha, x^*)$ for some $\beta \neq \perp$. We show that $\widetilde{\mathsf{Eval}}(\widehat{C}, a) = \beta$ as well. According to Definition 4.3, the set of queries $\mathsf{RealQ}(\Pi, \Gamma, \alpha, x^*)$ contains also the query $[\mathcal{O}(c, r) = \widehat{C}]$, and also all the necessary f-queries for the evaluation of $C^f(a)$ (which is evaluated to β).
     Since these queries appear in $\mathsf{RealQ}(\Pi, \Gamma, \alpha, x^*)$, it holds that $\widetilde{f}$ and $\widetilde{\mathcal{O}}$ agree on the same queries, and therefore $\widetilde{\mathsf{Eval}}(\widehat{C}, a) = \beta$, as well.
  3. For every query $[\mathsf{Eval}(\widehat{C}, a) = \perp] \in \mathsf{RealQ}(\Pi, \Gamma, \alpha, x^*) \cup \mathsf{Partial}(\Gamma'_{\mathsf{no}})$ we show that $\widetilde{\mathsf{Eval}}(\widehat{C}, a) = \perp$ as well. There might be a contradiction only if one of the following occurs:
     (a) Assume that $[\mathsf{Eval}'(\widehat{C}, a) = \perp] \in \mathsf{Partial}(\Gamma'_{\mathsf{no}})$ but $[\mathcal{O}(C, r) = \widehat{C}] \in \mathsf{RealQ}(\Pi, \Gamma, \alpha, x^*)$ for some $C, r, a \in \{0,1\}^m$ and $\widehat{C} \in \{0,1\}^{10m^c}$. However, this case implies that the event $\mathsf{bad}_{\mathcal{A},\Gamma}(x)$ occurs (see $\mathsf{bad}^{(2)}_{i,m}(x)$ in Definition 4.6).
     (b) Assume that $[\mathsf{Eval}(\widehat{C}, a) = \perp] \in \mathsf{RealQ}(\Pi, \Gamma, \alpha, x^*)$ but $[\mathcal{O}'(C, r) = \widehat{C}] \in \mathsf{Partial}(\Gamma'_{\mathsf{no}})$ for some $C, r, a \in \{0,1\}^m$ and $\widehat{C} \in \{0,1\}^{10m^c}$. We claim that this case implies that $\mathcal{A}$ adds to $\mathsf{ExtendedQ}$ one of the queries that appear in $\mathsf{RealQ}(\Pi, \Gamma, \alpha, x^*)$. In particular, since the oracles $\mathsf{Partial}(\Gamma'_{\mathsf{yes}})$ and $\mathsf{Partial}(\Gamma'_{\mathsf{no}})$ are consistent, and since $[\mathcal{O}'(C, r) = \widehat{C}] \in \mathsf{Partial}(\Gamma'_{\mathsf{no}})$, from Requirement 3 in Definition 4.2 we conclude that $\mathsf{Partial}(\Gamma'_{\mathsf{no}})$ includes also a query $[\mathsf{Eval}(\widehat{C}, \alpha) = \beta] \in \mathsf{Partial}(\Gamma'_{\mathsf{no}})$ for some arbitrary $\alpha \in \{0,1\}^m$. This implies that in the update phase, $\mathcal{A}$ queries Γ on $[\mathsf{Eval}(\widehat{C}, \alpha)]$, receives $\perp$, and adds to $\mathsf{ExtendedQ}$ the set $\{[\mathsf{Eval}(\widehat{C}, b) = \perp]\}_{b \in \{0,1\}^m}$, and in particular also the query $[\mathsf{Eval}(\widehat{C}, a) = \perp]$ which exists also in $\mathsf{RealQ}(\Pi, \Gamma, \alpha, x^*)$.
     Therefore, assuming that Condition 2 in Claim 4.9 does not hold, this case cannot occur.

This completes the proof of Claim 4.9. ■

Assuming that $x \in L^\Gamma$, then after $\hat{\ell}$ iterations the set $\mathsf{ExtendedQ}$ contains all the queries in $\mathsf{AugQ}_{\mathsf{yes}}(\mathcal{R}, \Gamma, x, w^*_{\mathsf{yes}})$. Therefore, the algorithm cannot find in Step 3(b)ii an oracle $\mathsf{Partial}(\Gamma'_{\mathsf{no}})$ that is consistent with $\mathsf{ExtendedQ}$ and a witness w'_{no} for which $\overline{\mathcal{R}}^{\mathsf{Partial}(\Gamma'_{\mathsf{no}})}(x, w'_{\mathsf{no}}) = 1$. As a result, $\mathcal{A}$ halts and outputs **YES**.

Analyzing the case where $x \notin L^\Gamma$. Similarly to the above, we have the following claim:

Claim 4.10. *Assume that $x \notin L^{\Gamma}$, let w^*_{no} be as above, and assume that the event $\mathsf{bad}_{\mathcal{A},\Gamma}(x)$ does not occur. Then, $\mathcal{A}$ never outputs* **YES** *in Step 3b, and in each iteration of Step 3, exactly one of the following occurs:*

1. *$\mathcal{A}$ halts and outputs* **NO** *in Step 3a.*
2. *During the update phase, $\mathcal{A}$ adds to* ExtendedQ *at least one one of the queries that appear in* $\mathsf{AugQ}_{\mathsf{no}}(\overline{\mathcal{R}}, \Gamma, x, w^*_{\mathsf{no}})$.

Assuming that $x \notin L^{\Gamma}$, then after $\hat{\ell}$ iterations the set ExtendedQ contains all the queries in $\mathsf{AugQ}_{\mathsf{no}}(\overline{\mathcal{R}}, \Gamma, x, w^*_{\mathsf{no}})$. Therefore, the algorithm cannot find in Step 3(a)ii an oracle $\mathsf{Partial}(\Gamma'_{\mathsf{yes}})$ that is consistent with ExtendedQ and a witness w'_{yes} for which $\mathcal{R}^{\mathsf{Partial}(\Gamma'_{\mathsf{yes}})}(x, w'_{\mathsf{yes}}) = 1$. As a result, $\mathcal{A}$ halts and outputs **NO**.

Combining Claims 4.9 and 4.10, we conclude that for every input $x \in \{0,1\}^n$:

$$\Pr_{\Gamma}\left[\mathcal{A}^{\Gamma}(x) \neq \chi_{L^{\Gamma}}(x) \mid \overline{\mathsf{bad}_{\mathcal{A},\Gamma}(x)}\right] = 0 \ .$$

This concludes the proof of Claim 4.8. ■

References

[AR05] D. Aharonov and O. Regev. Lattice problems in NP cap coNP. *Journal of the ACM*, 52(5):749–765, 2005.

[AS15] G. Asharov and G. Segev. Limits on the power of indistinguishability obfuscation and functional encryption. In *Proceedings of the 56th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2015*, pages 191–209, 2015.

[AS16] G. Asharov and G. Segev. On constructing one-way permutations from indistinguishability obfuscation. In *Proceedings of the 13th Theory of Cryptography Conference, TCC 2016-A*, pages 512–541, 2016.

[BCE+95] P. Beame, S. A. Cook, J. Edmonds, R. Impagliazzo, and T. Pitassi. The relative complexity of NP search problems. In *Proceedings of the 27th Annual ACM Symposium on Theory of Computing, STOC 1995*, pages 303–314, 1995.

[BGI+01] B. Barak, O. Goldreich, R. Impagliazzo, S. Rudich, A. Sahai, S. P. Vadhan, and K. Yang. On the (im)possibility of obfuscating programs. In *Advances in Cryptology – CRYPTO '01*, pages 1–18, 2001.

[BGI+12] B. Barak, O. Goldreich, R. Impagliazzo, S. Rudich, A. Sahai, S. P. Vadhan, and K. Yang. On the (im)possibility of obfuscating programs. *Journal of the ACM*, 59(2):6, 2012.

[BPR15] N. Bitansky, O. Paneth, and A. Rosen. On the cryptographic hardness of finding a Nash equilibrium. In *Proceedings of the 56th Annual IEEE Symposium on Foundations of Computer Science*, pages 1480–1498, 2015.

[BPW16] N. Bitansky, O. Paneth, and D. Wichs. Perfect structure on the edge of chaos – trapdoor permutations from indistinguishability obfuscation. In *Proceedings of the 13th Theory of Cryptography Conference*, pages 474–502, 2016.

[CIY97] S. A. Cook, R. Impagliazzo, and T. Yamakami. A tight relationship between generic oracles and type-2 complexity theory. *Information and Computation*, 137(2):159–170, 1997.

[CLT$^+$15] R. Canetti, H. Lin, S. Tessaro, and V. Vaikuntanathan. Obfuscation of probabilistic circuits and applications. In *Proceedings of the 12th Theory of Cryptography Conference*, pages 468–497, 2015.

[GG00] O. Goldreich and S. Goldwasser. On the limits of nonapproximability of lattice problems. *J. Comput. Syst. Sci.*, 60(3):540–563, 2000.

[GGH$^+$13] S. Garg, C. Gentry, S. Halevi, M. Raykova, A. Sahai, and B. Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *Proceedings of the 54th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2013*, pages 40–49, 2013.

[Gol00] O. Goldreich. On security preserving reductions – revised terminology. Cryptology ePrint Archive, Report 2000/001, 2000.

[GPS15] S. Garg, O. Pandey, and A. Srinivasan. On the exact cryptographic hardness of finding a Nash equilibrium. Cryptology ePrint Archive, Report 2015/1078, 2015.

[GPS$^+$16] S. Garg, O. Pandey, A. Srinivasan, and M. Zhandry. Breaking the sub-exponential barrier in obfustopia. Cryptology ePrint Archive, Report 2016/102, 2016.

[IKO05] Y. Ishai, E. Kushilevitz, and R. Ostrovsky. Sufficient conditions for collision-resistant hashing. In *Proceedings of the 2nd Theory of Cryptography Conference*, pages 445–456, 2005.

[IN88] R. Impagliazzo and M. Naor. Decision trees and downward closures. In *Proceedings of the 3rd Annual Structure in Complexity Theory Conference*, pages 29–38, 1988.

[IR89] R. Impagliazzo and S. Rudich. Limits on the provable consequences of one-way permutations. In *Proceedings of the 21st Annual ACM Symposium on Theory of Computing*, pages 44–61, 1989.

[Kil88] J. Kilian. Founding cryptography on oblivious transfer. In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing*, pages 20–31, 1988.

[KMN$^+$14] I. Komargodski, T. Moran, M. Naor, R. Pass, A. Rosen, and E. Yogev. One-way functions and (im)perfect obfuscation. In *Proceedings of the 55th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2014*, pages 374–383, 2014.

[Lub96] M. Luby. Pseudorandomness and Cryptographic Applications. Princeton University Press, 1996.

[RSS16] A. Rosen, G. Segev, and I. Shahaf. Can PPAD hardness be based on standard cryptographic assumptions? Cryptology ePrint Archive, Report 2016/375, 2016.

[RTV04] O. Reingold, L. Trevisan, and S. P. Vadhan. Notions of reducibility between cryptographic primitives. In *Proceedings of the 1st Theory of Cryptography Conference, TCC 2004*, pages 1–20, 2004.

[Rud88] S. Rudich. Limits on the Provable Consequences of One-way Functions. PhD thesis, EECS Department, University of California, Berkeley, 1988.

[Sim98] D. R. Simon. Finding collisions on a one-way street: Can secure hash functions be based on general assumptions? In *Advances in Cryptology – EUROCRYPT '98*, pages 334–345, 1998.

[SW14] A. Sahai and B. Waters. How to use indistinguishability obfuscation: Deniable encryption, and more. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing*, pages 475–484, 2014.

[Wat15] B. Waters. A punctured programming approach to adaptively secure functional encryption. In *Advances in Cryptology – CRYPTO '15*, pages 678–697, 2015.

A Proof of Theorem 3.2

Proof of Theorem 3.2. Let $(\mathcal{R}, \overline{\mathcal{R}}, p, M, \epsilon_{M,1}, \epsilon_{M,2})$ be a fully black-box construction of an $\mathsf{NP} \cap$ coNP-language L from a one-way function f and an indistinguishability obfuscator $i\mathcal{O}$ for the class $\mathcal{C}$ of all polynomial-size oracle-aided circuits C^f. Recall that M is an oracle-aided polynomial-time algorithm, and we let $c > 0$ denote a constant such that $T_M(n) \leq n^c$ (i.e., n^c upper bounds the running time of M on inputs of length n). By considering the distribution $\mathfrak{S}_{2c}$, Theorem 3.3 guarantees the existence of an oracle-aided algorithm $\mathcal{A}$ that runs in time $T_{\mathcal{A}}(n) = \mathsf{poly}(n) \cdot 2^{2n^{1/2c}}$ such that:

$$\Pr\left[\ \mathcal{A}^{\mathsf{PSPACE},\Gamma} \text{ decides } L_n^{\Gamma}\ \right] \geq \epsilon_{\mathcal{A}}(n)$$

for any $n \in \mathbb{N}$, where $\epsilon_{\mathcal{A}}(n) = 1 - 2^{-5n}$, and the probability is taken over the choice of $\Gamma \leftarrow \mathfrak{S}_{2c}$. Definition 3.1 then states that there are two possible cases to consider: $\mathcal{A}$ can be used either for inverting the one-way function f, or for breaking the security of the indistinguishability obfuscator $i\mathcal{O}$. In the first case we obtain from Definition 3.1 that

$$\Pr\left[M^{\mathcal{A}^{\mathsf{PSPACE},\Gamma}}(f(x)) \in f^{-1}(f(x))\right] \geq \epsilon_{M,1}\left(T_{\mathcal{A}}(n)\right) \cdot \epsilon_{M,2}(n) \cdot \epsilon_{\mathcal{A}}(n)$$

for infinitely many values of $n \in \mathbb{N}$, where the probability is taken over the choice of $x \leftarrow \{0,1\}^n$ and $\Gamma \leftarrow \mathfrak{S}_{2c}$, and over the internal randomness of M. The algorithm M may invoke $\mathcal{A}$ on various security parameters (i.e., in general M is not restricted to invoking $\mathcal{A}$ only on security parameter n), and we denote by $\ell(n)$ the maximal security parameter on which M invokes $\mathcal{A}$ (when M itself is invoked on security parameter n). Thus, viewing $M^{\mathcal{A}}$ as a single oracle-aided algorithm that has access to a PSPACE-complete oracle and to Γ, its running time $T_{M^{\mathcal{A}}}(n)$ satisfies $T_{M^{\mathcal{A}}}(n) \leq T_M(n) \cdot T_{\mathcal{A}}(\ell(n))$ (this follows since M may invoke $\mathcal{A}$ at most $T_M(n)$ times, and the running time of $\mathcal{A}$ on each such invocation is at most $T_{\mathcal{A}}(\ell(n))$). In particular, viewing $M' \stackrel{\text{def}}{=} M^{\mathcal{A}^{\mathsf{PSPACE}}}$ as a single oracle-aided algorithm that has oracle access to Γ, implies that M' is a q-query algorithm where $q(n) = T_{M^{\mathcal{A}}}(n)$. Theorem 3.4 then implies that either $2^{n/4} \leq q(n)$ or $\epsilon_{M,1}\left(T_{\mathcal{A}}(n)\right) \cdot \epsilon_{M,2}(n) \cdot \epsilon_{\mathcal{A}}(n) \leq 2^{-n/2}$. In the first sub-case, noting that $\ell(n) \leq T_M(n) \leq n^c$, we obtain that

$$2^{n/4} \leq q(n) = T_{M^{\mathcal{A}}}(n) \leq T_M(n) \cdot T_{\mathcal{A}}(\ell(n)) \leq T_M(n) \cdot T_{\mathcal{A}}(n^c)$$

The running time $T_{\mathcal{A}}(n^c)$ of the adversary $\mathcal{A}$ (when given access to a PSPACE-complete oracle) on inputs of length n^c is upper bounded by $\mathsf{poly}(n^c) \cdot 2^{2(n^c)^{1/2c}} \leq 2 \cdot 2^{2\sqrt{n}}$, and since $T_M(n) \geq n^c$ this rules out this sub-case. In the second sub-case, we have that $\epsilon_{M,1}\left(T_{\mathcal{A}}(n)\right) \cdot \epsilon_{M,2}(n) \cdot \epsilon_{\mathcal{A}}(n) \leq 2^{-n/2}$.

Applying $T_{\mathcal{A}}(n) \leq 2^{2n^c}$ and $\epsilon_{\mathcal{A}}(n) \geq 1/2$ we have that $\epsilon_{M,1}(2^{n^{1/\zeta}}) \cdot \epsilon_{M,2}(n) \leq 2^{-n/4}$ for some constant $\zeta > 1$.

In the second case we obtain from Definition 3.1 that

$$\left| \Pr \left[\mathsf{Exp}^{\mathsf{iO}}_{(f,i\mathcal{O}),i\mathcal{O},M^{\mathcal{A}},\mathcal{C}}(n) = 1 \right] - \frac{1}{2} \right| \geq \epsilon_{M,1}\left(T_{\mathcal{A}}(n)\right) \cdot \epsilon_{M,2}(n) \cdot \epsilon_{\mathcal{A}}(n).$$

Theorem 3.4 then implies that either $2^{n/4} \leq q(n)$ or $\epsilon_{M,1}\left(T_{\mathcal{A}}(n)\right) \cdot \epsilon_{M,2}(n) \cdot \epsilon_{\mathcal{A}}(n) \leq 2^{-n/4}$. Similarly to the previous case, the first sub-case can be ruled out, and in the second sub-case we have that $\epsilon_{M,1}(2^{n^{1/\zeta}}) \cdot \epsilon_{M,2}(n) \leq 2^{-n/4}$ for some constant $\zeta > 1$. ■