J Cryptol (2018) 31:698–736
https://doi.org/10.1007/s00145-017-9268-6

Journal of
CRYPTOLOGY

On Constructing One-Way Permutations from Indistinguishability Obfuscation*

Gilad Asharov†
Cornell Tech, New York, NY, USA
asharov@cornell.edu

Gil Segev‡
School of Computer Science and Engineering, Hebrew University of Jerusalem, 91904 Jerusalem, Israel
segev@cs.huji.ac.il

Communicated by Kenneth G. Paterson.

Received 18 December 2015 / Revised 16 August 2017
Online publication 15 September 2017

Abstract. We prove that there is no black-box construction of a one-way permutation family from a one-way function and an indistinguishability obfuscator for the class of all oracle-aided circuits, where the construction is "domain invariant" (i.e., where each permutation may have its own domain, but these domains are independent of the underlying building blocks). Following the framework of Asharov and Segev (FOCS '15), by considering indistinguishability obfuscation for *oracle-aided* circuits we capture the common 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 (STOC '14) and its variants, as well as sub-exponential security assumptions. For example, we fully capture the construction of a trapdoor permutation family from a one-way function and an indistinguishability obfuscator due to Bitansky, Paneth, and Wichs (TCC '16). Their construction is *not* domain invariant, and our result shows that this, somewhat undesirable property, is unavoidable using the common techniques. In fact, we observe that constructions which are not domain invariant circumvent all known negative results for constructing one-way permutations based on one-way functions, starting with Rudich's seminal work (PhD thesis '88). We revisit this classic and fundamental problem and

*A preliminary version of this work appeared in *Proceedings of the 13th Theory of Cryptography Conference* (*TCC*), pages 512–541, 2016.

† Currently supported by a Junior Fellow award from the Simons Foundation. This work was completed while the author was a post-doctoral researcher at the Hebrew University's School of Computer Science and Engineering, and supported by the Israeli Centers of Research Excellence (I-CORE) Program (Center No. 4/11).

‡ Supported by the European Union's 7th Framework Program (FP7) via a Marie Curie Career Integration Grant (Grant No. 618094), by the European Union's Horizon 2020 Framework Program (H2020) via an ERC Grant (Grant No. 714253), 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.

resolve this somewhat surprising gap by ruling out *all* such black-box constructions—even those that are not domain invariant.

Keywords. Indistinguishability obfuscation, Black-box separations, One-way permutations, Lower bounds.

1. Introduction

One-way permutations are among the most fundamental primitives in cryptography, enabling elegant constructions of a wide variety of central cryptographic primitives. Although various primitives, such as universal one-way hash functions and pseudo-random generators, can be constructed based on any one-way function [39,54], their constructions based on one-way permutations are much simpler and significantly more efficient [11,51].

Despite the key role of one-way permutations in the foundations of cryptography, only very few candidates have been suggested over the years. Whereas one-way functions can be based on an extremely wide variety of assumptions, candidate one-way permutation families are significantly more scarce. Up until recently, one-way permutation families were known to exist only based on the hardness of problems related to discrete logarithms and factoring [53,55]. Moreover, the seminal work by Rudich [57], within the framework of Impagliazzo and Rudich [42], initiated a line of research showing that a one-way permutation cannot be constructed in a black-box manner from a one-way function or from various other cryptographic primitives [21,45,47,48].

Very recently, a one-way (trapdoor!) permutation family was constructed by Bitansky et al. [15] based on indistinguishability obfuscation [9,30] and one-way functions. Their breakthrough result provides the first trapdoor permutation family that is not based on the hardness of factoring, and motivates the task of studying the extent to which indistinguishability obfuscation can be used for constructing one-way permutations. Specifically, their work leaves completely unresolved the following question, representing to a large extent the "holy grail" of constructing one-way permutations:

> *Is there a construction of a one-way permutation over* $\{0,1\}^n$ *based on indistinguishability obfuscation and one-way functions?*

While exploring this intriguing question, one immediately identifies two somewhat undesirable properties in the construction of Bitansky, Paneth, and Wichs:

- Even when not aiming for trapdoor invertibility, their approach seems limited to providing a *family* of permutations instead of a *single* permutation[1].
- Their construction provides permutations that are defined over domains which both depend on the underlying building blocks and are extremely sparse[2].

[1] Moreover, Bitansky et al. note that their permutations do not seem certifiable. That is, they were not able to provide an efficient method for certifying that a key is well formed and describes a valid permutation. In contrast, a single permutation is certifiable by its nature.

[2] Each permutation in their construction is defined over a domain of elements of the form $(x, \mathsf{PRF}_K(x))$, where PRF is a pseudorandom function, and each permutation is associated with a different key K. This domain depends on the underlying building block, i.e., the pseudorandom function (equivalently, one-way function).

From the theoretical perspective, one-way permutation families with these two properties are typically still useful for most constructions that are based on one-way permutations. However, such families lack the elegant structure that makes constructions based on one-way permutations more simple and significantly more efficient when compared to constructions based on one-way functions.

1.1. *Our Contributions*

Motivated by the recent construction of Bitansky et al. [15], we study the limitations of using indistinguishability obfuscation for constructing one-way permutations. Following the framework of Asharov and Segev [3], we consider indistinguishability obfuscation for *oracle-aided* circuits and thus capture the common 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 [59] and its variants, as well as sub-exponential security assumptions. For example, we fully capture the construction of a trapdoor permutation family from a one-way function and an indistinguishability obfuscator due to Bitansky et al. [15]. We refer the reader to Sect. 1.3.1 for an overview of our framework and of the type of constructions that it captures.

Our work considers three progressively weaker one-way permutation primitives: (1) a *domain-invariant* one-way permutation, (2) a domain-invariant one-way permutation *family*, and (3) a one-way permutation family (which may or may not be domain invariant). Roughly speaking, we say that a construction of a one-way permutation (or a one-way permutation family) is domain invariant if the domain of the permutation is independent of the underlying building blocks. (In the case of a permutation family, we allow each permutation to have its own domain, but these domains have to be independent of the underlying building blocks, or, in an oracle setting, should be the same for any valid instantiation of the oracle—see Definition 3.2.)

Within our framework, we prove the following two impossibility results, providing a tight characterization of the feasibility of constructing these three progressively weaker one-way permutation primitives based on one-way functions and indistinguishability obfuscation using the common techniques. (We summarize this characterization in Fig. 1.)

$i\mathcal{O}$+OWF $\not\Rightarrow$ domain-invariant OWP family. Bitansky et al. [15] showed that any sub-exponentially secure indistinguishability obfuscator and one-way function imply a one-way permutation family which is *not* domain invariant. We show that using the common techniques (as discussed above) one cannot construct the stronger primitive of a *domain-invariant* one-way permutation family (even when assuming sub-exponential security). In particular, we show that the above-described undesirable properties of their construction are unavoidable unless new non-black-box techniques are introduced.[3]

[3] In addition to the above-described undesirable properties, our impossibility result holds even for constructions of one-way permutation families that have a "pseudo" input sampling procedure instead of an "exact" input sampling procedure (as in [15]), as well as to constructions that are not necessarily certifiable (again, as in [15]).

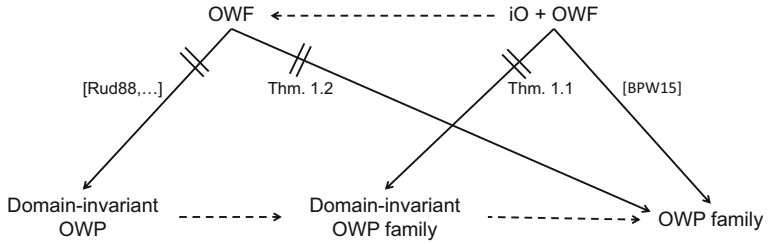

Fig. 1. A dashed arrow from a primitive A to a primitive B indicates that A implies B by definition. Bitansky et al. [15] showed that any sub-exponentially secure indistinguishability obfuscator and one-way function imply a one-way permutation family (which is not domain invariant), and we show that one cannot construct the stronger primitive of a *domain-invariant* one-way permutation family unless new non-black-box techniques are introduced (even when assuming sub-exponential security). The line of research starting with Rudich [57] showed that one cannot construct a *domain-invariant* one-way permutation from a one-way function in a black-box manner. We improve this result, showing that one cannot construct the weaker primitive of a one-way permutation *family* (even one that is *not* domain invariant) from a one-way function in a black-box manner (again, even when assuming sub-exponential security) .

Theorem 1.1. *There is no fully black-box construction of a domain-invariant one-way permutation family from a one-way function f and an indistinguishability obfuscator for the class of all oracle-aided circuits C^f.*

OWF $\not\Rightarrow$ OWP family. In fact, we observe that constructions which are not domain invariant circumvent the known negative results for constructing one-way permutations based on one-way functions, starting with Rudich's seminal work [45,48,52,57]. We revisit this classic and fundamental problem and resolve this surprising gap by ruling out *all* black-box constructions of one-way permutation *families* from one-way functions—even those that are *not* domain invariant.

Theorem 1.2. *There is no fully black-box construction of a one-way permutation family (even a non-domain-invariant one) from a one-way function.*

For conclusion, while the work of Rudich rules out constructions of a single domain-invariant one-way permutation based on one-way functions, our work extends this result in two orthogonal directions. First, we rule out constructions that are based on indistinguishability obfuscation, in addition to one-way functions (Theorem 1.1). Second, while the result of Rudich applies only to domain-invariant constructions, Theorem 1.2 rules out even non-domain-invariant constructions.

1.2. *Related Work*

The recent line of research focusing on new constructions based on indistinguishability obfuscation has been extremely fruitful so far. (See, for example, [1,2,6,7,13–20,23–26,29–31,37,41,43,59,60] and the references therein.) However, the extent to which indistinguishability obfuscation can be used as a building block has been insufficiently explored. Our approach for proving meaningful impossibility results for constructions based on indistinguishability obfuscation is based on that of Asharov and Segev [3] (which, in turn, was inspired by that of Brakerski et al. [10]). They showed that the com-

mon techniques (including non-black-box ones) that are used in constructions based on indistinguishability obfuscation can be captured by considering the stronger notion of indistinguishability obfuscation for oracle-aided circuits. (See Sect. 1.3.1 for an elaborate discussion) Generalizing the work of Simon [58] and Haitner et al. [38], they showed that using these common techniques one cannot construct a collision-resistant hash function family from a general-purpose indistinguishability obfuscator (even when assuming sub-exponential security). In addition, generalizing the work of Impagliazzo and Rudich [42] and Brakerski et al. [10], they showed a similar result from constructing a perfectly complete key agreement protocol from a private-key functional encryption scheme (again, even when assuming sub-exponential security).

It is far beyond the scope of this paper to provide an overview of the lines of research on black-box impossibility results in cryptography. (See, for example, [4,5,8,12,22,27,28,32–34,40,42,49,50,56,58,61] and the references therein.) Impossibility results for constructing one-way permutations start with the seminal work of Rudich [57]. This line of research has successfully shown that one-way permutations cannot be based on a variety of fundamental cryptographic primitives (e.g., [21,45,47,48]). However, these impossibility results capture only constructions of a *single* permutation that is *domain invariant*, and do not seem to capture more general constructions (such as the construction of Bitansky et al. [15] producing a permutation *family* which is *not* domain invariant).

The notion of "domain invariance" that we consider in this work for black-box constructions is somewhat related to that of "function obliviousness" that was introduced by Dachman-Soled et al. [28] for coin-flipping protocols. They proved an impossibility result for constructing an optimally fair coin-flipping protocol based on any one-way function, as long as the outcome of the protocol is completely independent of the specific one-way function that is used.

1.3. *Overview of Our Results*

In this section we provide a high-level overview of our two results. First, in Sect. 1.3.1 we describe the framework that enables us to prove a meaningful impossibility result for constructions that are based on indistinguishability obfuscation. Next, in Sect. 1.3.2 we describe Rudich's attack for inverting any domain-invariant permutation relative to a random oracle. Extending Rudich's approach, we then discuss the main technical ideas underlying our results: In Sect. 1.3.3 we present an attack on any domain-invariant permutation family relative to our, significantly more structured, oracle, and in Sect. 1.3.4 we generalize Rudich's attack to non-domain-invariant permutation families in the random-oracle model.

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

The fact that constructions that are based on indistinguishability obfuscation are almost always *non-black-box* makes it extremely challenging to prove any impossibility results. For 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 [3], most of the non-black-box techniques that are used on 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 2.4), 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 [59] and its variants (as well as sub-exponential security assumptions—which are already captured by most frameworks for black-box impossibility results).

Example: The Sahai–Waters Approach. Consider, for example, the construction of a public-key encryption scheme from a one-way function and a general-purpose indistinguishability obfuscator by Sahai and Waters [59]. Their construction relies on the underlying one-way function in a non-black-box manner. However, relative to an oracle that allows the existence of a one-way function f and indistinguishability obfuscation $i\mathcal{O}$ for *oracle-aided circuits*, it is in fact a fully black-box construction. Specifically, Sahai and Waters use the underlying indistinguishability obfuscator for obfuscating a circuit that invokes a puncturable pseudorandom function and a pseudorandom generator as subroutines. Given that puncturable pseudorandom functions and pseudorandom generators can be based on any one-way function in a fully black-box manner, from our perspective such a circuit is a polynomial-size oracle-aided circuit C^f—which can be obfuscated using $i\mathcal{O}$. (We refer to reader to [3, Sec. 4.6] for an in-depth technical treatment.)

This reasoning extends to various variants of the punctured programming approach by Sahai and Waters [59] and, in particular, fully captures the construction of a trapdoor permutation family from a one-way function and an indistinguishability obfuscator due to Bitansky et al. [15]. As noted in [3], this approach does not capture constructions that rely on the obfuscator itself in a non-black-box manner (e.g., [13])[4], or constructions that rely on zero-knowledge techniques and require using NP reductions[5].

The Oracle. Our first result is obtained by presenting an oracle Γ relative to which the following two properties hold: (1) There is no domain-invariant one-way permutation family, 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 quite intuitive 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 $C^f(x)$. We refer the reader to Sect. 3.2 for more details.

[4] With the exception of obfuscating a function that may invoke an indistinguishability obfuscator in a black-box manner. This is captured by our approach—see [3, Sec. 3.1].

[5] Such techniques are captured by the work of Brakerski et al. [10], and we leave it as an intriguing open problem to see whether the two approaches for capturing non-black-box techniques can be unified.

The vast majority of our effort is in showing that relative to Γ there is no domain-invariant one-way permutation family. Specifically, as for the second part, our oracle Γ is somewhat similar to the oracle introduced by [3], relative to which they proved the existence of an exponentially secure one-way function and an exponentially secure indistinguishability obfuscator. (See Sect. 3.2 for the differences between the oracles.)

In the remainder of this section, we first provide a high-level overview of Rudich's attack on any single domain-invariant permutation in the random-oracle model. Inspired by this attack, in Sects. 1.3.3 and 1.3.4 we explain the main challenges in extending Rudich's attack to domain-invariant constructions relative to our oracle and to non-domain-invariant constructions in the random-oracle model. We again refer the reader to Fig. 1 which summarizes our characterization of the feasible constructions.

1.3.2. *Warm-up: Rudich's Attack in the Random-Oracle Model*

Following [45,48,57] we show that for any oracle-aided polynomial-time algorithm P, if P^f implements a permutation over the same domain $\mathcal{D}$ for all functions f (i.e., P is domain invariant), then there exists an oracle-aided algorithm $\mathcal{A}$ that for any function f inverts P^f with probability 1 by querying f for only a polynomial number of times. The algorithm $\mathcal{A}$ is given some string $y^* \in \mathcal{D}$ and oracle access to f and is required to find the unique $x^* \in \mathcal{D}$ such that $P^f(x^*) = y^*$. It first initializes a set of queries/answers Q, which will contain the actual queries made by $\mathcal{A}$ to the true oracle f. It repeats the following steps polynomially many times:

1. **Simulation:** $\mathcal{A}$ finds an input $x' \in \mathcal{D}$ and a set of oracle queries/answers f' that is consistent with Q (i.e., $f'(w) = f(w)$ for every $w \in Q$) such that $P^{f'}(x') = y^*$.
2. **Evaluation:** $\mathcal{A}$ evaluates $P^f(x')$ (i.e., evaluation with respect to the true oracle f). If the output is y^*, it terminates and outputs x'.
3. **Update:** $\mathcal{A}$ asks f for all queries in f' that are not in Q, and updates the set Q.

The proof relies on the following observation: In each iteration, either (1) $\mathcal{A}$ finds the pre-image x^* such that $P^f(x^*) = y^*$, or (2) in the update phase, $\mathcal{A}$ queries f with at least one new query that is also made by P during the computation of $P^f(x^*) = y^*$.

Intuitively, if neither of the above holds, then we can construct a "hybrid" oracle $\widetilde{f}$ that behaves like f in the evaluation of $P^f(x^*) = y^*$ and behaves like f' in the evaluation of $P^{f'}(x') = y^*$. This hybrid oracle can be constructed since the two evaluations $P^{f'}(x')$ and $P^f(x^*)$ have no further intersection queries rather than the queries which are already in Q. According to this hybrid oracle $\widetilde{f}$, it holds that $P^{\widetilde{f}}(x') = P^{\widetilde{f}}(x^*) = y^*$ but yet $x^* \neq x'$, and thus, relative to $\widetilde{f}$ the value y^* has two pre-images, in contradiction to the fact that P always implements a permutation. Using this claim, since there are only polynomially many f-queries in the evaluation of $P^f(x^*) = y^*$, the algorithm $\mathcal{A}$ must output x^* 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 P).

1.3.3. *Attacking Domain-Invariant Permutation Families Relative to Our Oracle*

We extend the attack described above in two different aspects. First, we rule out constructions of domain-invariant permutation *families* and not just a single permutation. Second, we extend the attack to work relative to our oracle, which is a significantly more

structured oracle than a random oracle and therefore raises new technical challenges. Indeed, by the discussion in Sect. 1.3.1, relative to our oracle *there exists a non-domain-invariant construction of one-way permutation family* [15]. This mere fact represents the subtleties we have to deal with in our setting. In the following overview, we focus our attention on the challenges that arise due to the structure of our oracle, as these are the most important and technically challenging ones.

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.

1. 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 f and $\mathcal{O}$) which may give immense power to the construction P. Specifically, unlike in Rudich's case, here it is no longer true that the computation $P^{\Gamma}(x^*)$ performs a polynomial number of oracle queries (although P itself is of polynomial size).
2. 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 Rudich's case a hybrid oracle always exists.)
3. Finally, the third challenge is the fact that $\mathcal{O}$ is *injective*, which causes the following problem (somewhat similar to [48]). In our case, we are forced to assume that P^{Γ} is a permutation only when $\mathcal{O}$ is an *injective length-increasing* function and not just any arbitrary function as in Rudich's case (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 oracle $\mathcal{O}'$ on elements that are not in Q.

We revise the attack and its analysis to deal with the above obstacles. As in Rudich's attack, the algorithm $\mathcal{A}$ considers the collection of all oracles that are consistent with Q. However, for dealing with the third challenge, it then chooses one of these oracles *uniformly at random* and does not pick just an arbitrarily one as in Rudich's attack. We then show that with all but an exponentially small probability, there is no overlap between the range of the sampled oracle $\mathcal{O}'$ and the true oracle $\mathcal{O}$, and therefore, the hybrid oracle $\widetilde{\mathcal{O}}$ can almost always be constructed in an injective manner. Then, dealing with the first challenge, we show that Eval does not give P a significant capability as one may imagine. Intuitively, this is due to the fact that $\mathcal{O}$ is length increasing, and therefore, its range is very sparse. As a result, it is hard to sample a valid image of $\mathcal{O}$ without first querying it, and almost any Eval query can be simulated by the construction P itself. Finally, due to the dependencies between the oracles, for dealing with the second challenge, the algorithm $\mathcal{A}$ will have to sample additional, carefully chosen, polynomially many queries that do not necessarily appear in the evaluations $P^{\Gamma}(x^*) = y^*$ or $P^{\Gamma'}(x') = y^*$, but are related to the set of queries that appears in these evaluations. This results in a rather involved proof, where we carefully define this set of queries and extend the analysis accordingly.

As expected, our proof does not extend to constructions that are not domain invariant. For example, in such constructions for two distinct (injective) functions Γ and Γ', the domain of the permutations P^{Γ} and $P^{\Gamma'}$ may be completely distinct, and this forces additional restrictions on the number of oracles Γ' that are "valid" (i.e., can be used to construct the hybrid oracle $\widetilde{\Gamma}$ as above). As a result, while in the original proof of Rudich all of the oracles Γ' that the adversary may pick are valid, and while in our case all but some exponentially small amount of oracles Γ' are valid, here the number of valid oracles may be significantly smaller and therefore the attack may succeed with only a negligibly small probability.

1.3.4. *Attacking Non-domain-Invariant Permutation Families in the Random-Oracle Model*

At a first sight, it seems that a natural approach toward ruling out non-domain-invariant families relative to a random oracle is to reduce them to the case of a single permutation. That is, the adversary receives some index α of some permutation in the family, together with the challenge element $y^* \in \mathcal{D}_\alpha^f$ which it needs to invert. (Note that now the respective domain $\mathcal{D}_\alpha^f$ may depend on both f and α.) A natural approach is to apply Rudich's attack to the single permutation $P^f(\alpha, \cdot)$.

However, this approach seems somewhat insufficient due to the following reasons. First, since the construction is not domain invariant, the set of valid indices depends on the underlying primitive, and the set of valid indices for the true oracle f may be completely different than the set of valid indices for the oracle f' that will be sampled by $\mathcal{A}$ in each iteration (e.g., α might even not be a valid index with respect to the sampled f').

Second, when $\mathcal{A}$ inverts y^* relative to f', it may be that the pre-image x' that it finds is not even in the domain $\mathcal{D}_\alpha^f$ of the permutation $P^f(\alpha, \cdot)$ that it needs to invert. That is, it may be that even when the index α is valid relatively to both f and f', the domain of the permutation indexed by α relative to f is completely different than the domain relative to f'. One can try restricting $\mathcal{A}$ to sampling x' from the domain $\mathcal{D}_\alpha^f$, but conditioning on $P^{f'}(\alpha, x') = y^*$ it is not clear that such an x' even exists (and, even if it exists, $\mathcal{A}$ would typically need an exponential number of queries to f for finding it—since $\mathcal{A}$ has no "simple" representation of the sets $\mathcal{D}_\alpha^f$ and $\mathcal{D}_\alpha^{f'}$).

Finally, even when x' is the pre-image of y^* relative to f' and x^* is the pre-image of y^* relative to f, we have no guarantee that neither x' or x^* are even in the domain of the permutation indexed by α when considering the hybrid oracle $\widetilde{f}$. Therefore, the fact that $P^f(\alpha, x^*) = P^{f'}(\alpha, x')$ and $x^* \neq x'$ may not indicate any contradiction.

In Sect. 4 we show how to overcome these obstacles. Intuitively, when sampling some function f' and the element x', the algorithm $\mathcal{A}$ samples in addition two "certificates" that ensure that α is a valid index relative to f', and that x' is in the respective domain. These certificates include the randomness used by the index sampling and input sampling procedures of the permutation family, as well as all oracle queries and answers that are involved in the execution of these two procedures. We later use these certificates when defining the hybrid function $\widetilde{f}$ and thus ensure that α is a valid index relative to $\widetilde{f}$ and that x' is in the respective domain. Similarly, relative to the true oracle f, there exist some other certificates (which are unknown to $\mathcal{A}$) that ensure that α and x^* are valid

and are considered as well when defining the hybrid $\widetilde{f}$. Only then we can conclude the existence of a hybrid oracle $\widetilde{f}$ relative to which there exist an index α and two distinct inputs x^* and x' in the domain of α such that $P^{\widetilde{f}}(\alpha, x^*) = P^{\widetilde{f}}(\alpha, x')$.

1.4. *Paper Organization*

The remainder of this paper is organized as follows. In Sect. 2 we introduce the cryptographic primitives under consideration in this paper, oracle-aided one-way permutation families and indistinguishability obfuscation for oracle-aided circuits, as well as some standard notation. In Sect. 3 we present our negative result for constructing domain-invariant one-way permutation families from indistinguishability obfuscation and one-way functions. Then, in Sect. 4 we present our negative result for constructing one-way permutation families from one-way functions.

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 $\{1, \ldots, n\}$. A function $\mathsf{negl} : \mathbb{N} \to \mathbb{R}^+$ is *negligible* if for every constant $c > 0$ there exists an integer N_c such that $\mathsf{negl}(n) < n^{-c}$ for all $n > N_c$. Throughout the paper, we denote by n the security parameter. For a function $f : \{0,1\}^* \to \{0,1\}^*$, we let $\mathrm{Im}(f)$ to denote the image of f, that is $\mathrm{Im}(f) = \{y \in \{0,1\}^* \mid \exists x \in \{0,1\}^* \text{ s.t. } f(x) = y\}$.

2.1. *Oracle-Aided One-Way Permutation Families*

We consider the standard notion of a one-way permutation family (see, for example, [36]) when naturally generalized to the setting of oracle-aided algorithms (as required within the context of black-box reductions [42,56]). We start by formalizing the notion of an oracle-aided permutation family and then introduce the standard one-wayness requirement.

Definition 2.1. Let $(\mathsf{Gen}, \mathsf{Samp}, \mathsf{P})$ be a triplet of oracle-aided polynomial-time algorithms. We say that $(\mathsf{Gen}, \mathsf{Samp}, \mathsf{P})$ is an oracle-aided permutation family relative to an oracle Γ if the following properties are satisfied:

- **Index Sampling:** $\mathsf{Gen}^{\Gamma}(\cdot)$ is a probabilistic algorithm that takes as input the security parameter 1^n and produces a distribution over indices α. For every $n \in \mathbb{N}$ we denote by $\mathcal{I}_n^{\Gamma}$ the support of the distribution $\mathsf{Gen}^{\Gamma}(1^n)$, and we let $\mathcal{I}^{\Gamma} \stackrel{\text{def}}{=} \bigcup_{n \in \mathbb{N}} \mathcal{I}_n^{\Gamma}$.
- **Input Sampling:** $\mathsf{Samp}^{\Gamma}(\cdot)$ is a probabilistic algorithm that takes as input an index $\alpha \in \mathcal{I}^{\Gamma}$, and produces a uniform distribution over a set denoted $\mathcal{D}_{\alpha}^{\Gamma}$.
- **Permutation Evaluation:** For any index $\alpha \in \mathcal{I}^{\Gamma}$, $\mathsf{P}^{\Gamma}(\alpha, \cdot)$ is a deterministic algorithm that computes a permutation over the set $\mathcal{D}_{\alpha}^{\Gamma}$.

Definition 2.2. An oracle-aided permutation family $(\mathsf{Gen}, \mathsf{Samp}, \mathsf{P})$ is one way relative to an oracle Γ if for any probabilistic polynomial-time algorithm $\mathcal{A}$ there exists a negligible function $\mathsf{negl}(\cdot)$ such that

$$\Pr\left[\mathcal{A}^{\Gamma}(\alpha, \mathsf{P}^{\Gamma}(\alpha, x)) = x\right] \leq \mathsf{negl}(n)$$

for all sufficiently large $n \in \mathbb{N}$, where the probability is taken over the choice of $\alpha \leftarrow \mathsf{Gen}^{\Gamma}(1^n)$, $x \leftarrow \mathsf{Samp}^{\Gamma}(\alpha)$, and over the internal randomness of $\mathcal{A}$.

2.2. *Indistinguishability Obfuscation for Oracle-Aided Circuits*

We consider the standard notion of indistinguishability obfuscation [9,30] when naturally generalized to oracle-aided circuits (i.e., circuits that may contain oracle gates in addition to standard gates). 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 2.3. 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 2.4. 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

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

  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. Impossibility for Constructions Based on $i\mathcal{O}$ and One-Way Functions

In this section we present our negative result for domain-invariant constructions of a one-way permutation family from a one-way function and an indistinguishability obfuscator. In Sect. 3.1 we formally define the class of constructions to which our negative result applies. Then, in Sect. 3.2 we present the structure of our proof, which is provided in Sects. 3.3, 3.4 and 3.5.

3.1. *The Class of Constructions*

We consider fully black-box constructions of a one-way permutation family from a one-way function f and an indistinguishability obfuscator for all oracle-aided circuits C^f. Following [3], we model these primitives as two independent building blocks due to the following reasons. First, although indistinguishability obfuscation is known to imply one-way functions under reasonable assumptions [43], this enables us to prove an unconditional result. Second, and more importantly, this enables us to capture the common techniques that have been used so far in constructions based on indistinguishability obfuscation. As discussed in Sect. 1.3.1, these include, in particular, *non-black-box* techniques such as the punctured programming approach of Sahai and Waters [59] and its variants.

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 2.3.) The following definition is based on those of [3] (which, in turn, are motivated by [35,46,56]).

Definition 3.1. A fully black-box construction of a one-way permutation family from a one-way function and an indistinguishability obfuscator for the class $\mathcal{C} = \{C_n\}_{n\in\mathbb{N}}$ of all polynomial-size oracle-aided circuits, consists of a triplet of oracle-aided probabilistic polynomial-time algorithms $(\mathsf{Gen}, \mathsf{Samp}, \mathsf{P})$, an oracle-aided algorithm M that runs in time $T_M(\cdot)$, and functions $\epsilon_{M,1}(\cdot)$ and $\epsilon_{M,2}(\cdot)$, such that the following conditions hold:

- **Correctness:** For any function f and for any function $i\mathcal{O}$ such that $i\mathcal{O}(C;r)^f \equiv C^f$ for all $C \in \mathcal{C}$ and $r \in \{0,1\}^*$, the triplet $(\mathsf{Gen}, \mathsf{Samp}, \mathsf{P})$ is a permutation family relative to the oracle $(f, i\mathcal{O})$ (as in Definition 2.1).
- **Black-Box Proof of Security:** For any function f, for any function $i\mathcal{O}$ 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 $T_{\mathcal{A}} = T_{\mathcal{A}}(n)$, and for any function $\epsilon_{\mathcal{A}} = \epsilon_{\mathcal{A}}(n)$, if

$$\Pr\left[\mathcal{A}^{f,i\mathcal{O}}(\alpha, \mathsf{P}^{f,i\mathcal{O}}(\alpha, x)) = x\right] \geq \epsilon_{\mathcal{A}}(n)$$

for infinitely many values of $n \in \mathbb{N}$, where the probability is taken over the choice of $\alpha \leftarrow \mathsf{Gen}^{f,i\mathcal{O}}(1^n)$, $x \leftarrow \mathsf{Samp}^{f,i\mathcal{O}}(\alpha)$, and over the internal randomness of $\mathcal{A}$, then either

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

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

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

for infinitely many values of $n \in \mathbb{N}$. (See Definition 2.4 for the description of the experiment $\mathsf{Exp}^{\mathsf{iO}}_{(f,i\mathcal{O}),i\mathcal{O},M^{\mathcal{A}},\mathcal{C}}(n)$.)

The "Security Loss" Functions. Black-box constructions are typically formulated with a reduction algorithm M that runs in *polynomial* time and offers a *polynomial* security loss. In our setting, as we are interested in capturing constructions that may be based on super-polynomial security assumptions, we allow the algorithm M to run in arbitrary time $T_M(n)$ and to have an arbitrary security loss.

In general, the security loss of a reduction is a function of the adversary's running time $T_{\mathcal{A}}(n)$, of its success probability $\epsilon_{\mathcal{A}}(n)$, and of the security parameter $n \in \mathbb{N}$. Following Luby [46] and Goldreich [35], we simplify the presentation by considering Levin's unified security measure $T_{\mathcal{A}}(n)\cdot\epsilon_{\mathcal{A}}^{-1}(n)$. Specifically, our definition captures the security loss of a reduction by considering an "adversary-dependent" security loss $\epsilon_{M,1}(T_{\mathcal{A}}(n)\cdot\epsilon_{\mathcal{A}}^{-1}(n))$ and an "adversary-independent" security loss $\epsilon_{M,2}(n)$. By considering arbitrary security loss functions, we are indeed able to capture constructions that rely on super-polynomial security assumptions. For example, in the recent construction of Bitansky et al. [15] (and in various other recent constructions based on indistinguishability obfuscation), the adversary-dependent loss is polynomial whereas the adversary-independent loss is sub-exponential[6].

Domain-Invariant Constructions. We now define the notion of *domain invariance* which allows us to refine the above class of constructions. Recall that for an oracle-aided permutation family $(\mathsf{Gen},\mathsf{Samp},\mathsf{P})$ and for any oracle Γ, we denote by $\mathcal{I}_n^{\Gamma}$ the support of the distribution $\mathsf{Gen}^{\Gamma}(1^n)$ for every $n \in \mathbb{N}$, and we let $\mathcal{I}^{\Gamma} \stackrel{\text{def}}{=} \bigcup_{n\in\mathbb{N}}\mathcal{I}_n^{\Gamma}$ (i.e., $\mathcal{I}^{\Gamma}$ is the set of all permutation indices). In addition, for any permutation index $\alpha \in \mathcal{I}^{\Gamma}$ we denote by $\mathcal{D}_{\alpha}^{\Gamma}$ the domain of the permutation $\mathsf{P}^{\Gamma}(\alpha,\cdot)$.

Definition 3.2. An oracle-aided one-way permutation family $(\mathsf{Gen},\mathsf{Samp},\mathsf{P})$ is domain invariant relative to a set $\mathfrak{S}$ of oracles if the following conditions hold:

1. For every two oracles $\Gamma,\Gamma' \in \mathfrak{S}$ and for every $n \in \mathbb{N}$, the distributions $\mathsf{Gen}^{\Gamma}(1^n)$ and $\mathsf{Gen}^{\Gamma'}(1^n)$ are identical. In particular, there exists a sequence $\{\mathcal{I}_n\}_{n\in\mathbb{N}}$ such that $\mathcal{I}_n^{\Gamma} = \mathcal{I}_n$ for every $\Gamma \in \mathfrak{S}$ and for every $n \in \mathbb{N}$.

[6] This is also the situation, for example, when using "complexity leveraging" for arguing that any selectively secure identity-based encryption scheme is in fact adaptively secure.

2. For every two oracles $\Gamma, \Gamma' \in \mathfrak{S}$ and for every $\alpha \in \bigcup_{n\in\mathbb{N}} \mathcal{I}_n$, the domains $\mathcal{D}_\alpha^\Gamma$ and $\mathcal{D}_\alpha^{\Gamma'}$ are identical. In particular, there exists a sequence $\{\mathcal{D}_\alpha\}_{\alpha\in\mathcal{I}}$ such that $\mathcal{D}_\alpha^\Gamma = \mathcal{D}_\alpha$ for every $\Gamma \in \mathfrak{S}$ and for every $\alpha \in \bigcup_{n\in\mathbb{N}} \mathcal{I}_n$.

Perfect Correctness. We consider constructions where the triplet $(\mathsf{Gen}, \mathsf{Samp}, \mathsf{P})$ defines a permutation family relative to any (correct) oracles f and $i\mathcal{O}$. A stronger result would be to rule out constructions where the triplet $(\mathsf{Gen}, \mathsf{Samp}, \mathsf{P})$ defines a permutation family with high probability over the choice of f and $i\mathcal{O}$. We leave this as an interesting open problem.

We note that the combination of the works of Rudich [57] and Kahn et al. [44] does allow a similar freedom and rules out constructions of a single permutation P from one-way function even when the algorithm P^f defines a permutation with high probability over f. In contrast, perfect correctness was also assumed in the work of [48], and a first step toward generalizing our results would be to generalize imperfect correctness in this simplified setting. That is, to rule out domain-invariant constructions without perfect correctness of one-way permutations from injective one-way function.

3.2. *Proof Overview and the Oracle* Γ

Our result in this section is obtained by presenting a distribution over oracles Γ relative to which the following two properties hold: (1) There is no domain-invariant one-way permutation family $(\mathsf{Gen}, \mathsf{Samp}, \mathsf{P})$, 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. Equipped with the notation and terminology introduced in Sect. 3.1, we prove the following theorem:

Theorem 3.3. *Let* $(\mathsf{Gen}, \mathsf{Samp}, \mathsf{P}, M, T_M, \epsilon_{M,1}, \epsilon_{M,2})$ *be a fully black-box domain-invariant construction of a one-way permutation family from a one-way function* f *and an indistinguishability obfuscator for the class of all polynomial-size oracle-aided circuits* C^f*. Then, at least one of the following propertied holds:*

1. $T_M(n) \geq 2^{\zeta n}$ *for some constant* $\zeta > 0$ *(i.e., the reduction runs in exponential time).*
2. $\epsilon_{M,1}(n^c) \cdot \epsilon_{M,2}(n) \leq 2^{-n/4}$ *for some constant* $c > 1$ *(i.e., the security loss is exponential).*

In particular, the theorem implies that if the running time $T_M(\cdot)$ of the reduction is sub-exponential and the adversary-dependent security loss $\epsilon_{M,1}(\cdot)$ is polynomial as in the vast majority of constructions (and, in particular, as in the construction of Bitansky et al. [15]), then the adversary-independent security loss $\epsilon_{M,2}(\cdot)$ must be exponential (thus ruling out even constructions that rely on sub-exponential security assumptions—as discussed in Sect. 3.1).

In what follows, we describe the oracle Γ (more accurately, the distribution over such oracles) and then explain the structure of our proof.

The Oracle Γ. The oracle Γ is a triplet $\left(f, \mathcal{O}, \mathsf{Eval}^{f,\mathcal{O}}\right)$ that is sampled from a distribution $\mathfrak{S}$ defined 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 $\mathcal{O}_n : \{0,1\}^{2n} \to \{0,1\}^{10n}$ chosen uniformly at random. The function $\mathsf{Eval}_n^{f,\mathcal{O}}$ on input $(\widehat{C}, x) \in \{0,1\}^{10n} \times \{0,1\}^n$ 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 there is no domain-invariant one-way permutation family, 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.

The vast majority of our effort in this proof is in showing that relative to Γ there is no domain-invariant one-way permutation family. Specifically, as for the second part, our oracle Γ is somewhat similar to the oracle introduced by [3], relative to which they proved the existence of an exponentially secure one-way function and an exponentially secure indistinguishability obfuscator. The main difference between the oracles is that the function $\mathcal{O}$ in their case is a *permutation*, whereas in our case it is an *injective length-increasing* function. Since our aim here is to rule out constructions of one-way permutations, then clearly we cannot allow $\mathcal{O}$ to be a permutation. This requires us to revisit the proof of [3] and generalize it to the case where $\mathcal{O}$ is injective and length increasing.

In what follows, we say that an algorithm $\mathcal{A}$ that has oracle access to Γ is a q-query algorithm if it makes at most q queries to Γ, and each of its queries to Eval consists of a circuit of size at most q. (See Definition 3.12.)

Part 1: Inverting Any Domain-Invariant Construction. Building upon and generalizing the work of Rudich [57], we show that relative to the oracle Γ there are no domain-invariant one-way permutations families. As discussed in Sect. 1.3.2, Rudich presented an attacker that inverts any *single* domain-invariant permutation that has oracle access to a random function. Here we need to deal with constructions that have oracle access to a significantly more structured functionality[7] and that are permutation *families*. Nevertheless, inspired by the main ideas underlying Rudich's attacker we prove the following theorem in Sect. 3.3:

Theorem 3.4. (simplified) *Let* $(\mathsf{Gen}, \mathsf{Samp}, \mathsf{P})$ *be an oracle-aided domain-invariant permutation family. Then, there exist a polynomial $q(\cdot)$ and a q-query algorithm $\mathcal{A}$ such that*

$$\Pr\left[\mathcal{A}^{\Gamma}(\alpha, \mathsf{P}^{\Gamma}(\alpha, x)) = x\right] \geq 1 - 2^{-10}$$

[7] For example, there are dependencies between $\mathcal{O}$, Eval and f which allow Eval to query $\mathcal{O}$ for a exponential number of times.

for any $n \in \mathbb{N}$*, where the probability is taken over the choice of* $\Gamma \leftarrow \mathfrak{S}$, $\alpha \leftarrow \mathsf{Gen}^{\Gamma}(1^n)$, $x \leftarrow \mathsf{Samp}^{\Gamma}(\alpha)$*, and over the internal randomness of* $\mathcal{A}$*. Moreover, the algorithm* $\mathcal{A}$ *can be implemented in polynomial time given access to a* PSPACE*-complete oracle.*

Part 2: The Existence of a One-Way Function and an Indistinguishability Obfuscator. As discussed above, by refining the proof of [3] we prove that f is an exponentially secure one-way function relative to Γ, and we construct an exponentially secure indistinguishability obfuscator $i\mathcal{O}$. Our obfuscator 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 $\widehat{C}$. We prove the following theorem in Sects. 3.4 and 3.5:

Theorem 3.5. (simplified) *For any oracle-aided* $2^{n/4}$*-query algorithm* $\mathcal{A}$*, it holds that*

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

for all sufficiently large $n \in \mathbb{N}$*, where the probability is taken over the choice of* $\Gamma \leftarrow \mathfrak{S}$ *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.*

We note that Theorem 3.5 holds even if the adversary $\mathcal{A}$ has an access to a PSPACE-complete oracle, since our analysis holds for computationally unbounded adversaries where the only limitation is on their number of queries to the oracle Γ.

3.3. *Attacking Domain-Invariant Permutation Families Relative to* $\boldsymbol{\Gamma}$

We show that relative to the oracle Γ there are no domain-invariant one-way permutations families. As discussed in Sect. 1.3.2, Rudich presented an attacker that inverts any *single* domain-invariant permutation that has oracle access to a random function. Here we need to deal with constructions that have oracle access to a significantly more structured functionality. We prove the following theorem:

Theorem 3.6. *Let* $(\mathsf{Gen}, \mathsf{Samp}, \mathsf{P})$ *be an oracle-aided permutation family that is domain invariant relative to the support of the distribution* $\mathfrak{S}$*. Then, there exist a polynomial* $q(\cdot)$ *and a* q*-query algorithm* $\mathcal{A}$ *such that*

$$\Pr\left[\mathcal{A}^{\Gamma}(\alpha, \mathsf{P}^{\Gamma}(\alpha, x^*)) = x^*\right] \geq 1 - 2^{-10}$$

for any $n \in \mathbb{N}$*, where the probability is taken over the choice of* $\Gamma \leftarrow \mathfrak{S}$, $\alpha \leftarrow \mathsf{Gen}^{\Gamma}(1^n)$, $x^* \leftarrow \mathsf{Samp}^{\Gamma}(\alpha)$*, and over the internal randomness of* $\mathcal{A}$*. Moreover, the algorithm* $\mathcal{A}$ *can be implemented in polynomial time given access to a* PSPACE*-complete oracle.*

We first provide additional notation definitions that we require for the proof of the above theorem, and then we provide its formal proof.

The Event spoof. The event spoof will help up show that the oracle Eval does not provide the construction with any significant capabilities. We formally define this event and then state an important claim that will help us to prove both Theorem 3.6 and Theorem 3.5.

Definition 3.7. For any oracle-aided algorithm M, consider the following event spoof_n that may occur during an execution of $M^{\Gamma}(1^n)$: The algorithm makes a query $\mathsf{Eval}_n(\widehat{C}, a)$ with $|\widehat{C}| = 10n$ whose output is not $\perp$, yet $\widehat{C}$ was not an output of a previous $\mathcal{O}_n$-query.

In Sect. 3.5 we prove the following claim:

Claim 3.8. *For any $n \in \mathbb{N}$, for any f and $\mathcal{O}_{-n} = \{\mathcal{O}_m\}_{m\in\mathbb{N}, m\neq n} m$ and for any q-query algorithm M, the probability that* spoof_n *occurs in an execution of $M^{\Gamma}(1^n)$ satisfies*

$$\Pr_{\mathcal{O}_n}\left[\ \mathsf{spoof}_n\ \right] \leq q \cdot 2^{-8n}.$$

Notation. Denote by $\mathcal{T}$ the support of the distribution $\mathfrak{S}$ from which our oracle $\Gamma = (f, \mathcal{O}, \mathsf{Eval}^{f,\mathcal{O}})$ is sampled. Note that the oracle Eval is fully determined given f and $\mathcal{O}$, and therefore, it is enough to consider the choice of the latter only. For every $n \in \mathbb{N}$ we let $\mathcal{I}_n$ denote the support of $\mathsf{Gen}^{\Gamma}(1^n)$, which is the same for every $\Gamma \in \mathcal{T}$ due to the domain-invariant assumption, and we let $\mathcal{I} = \bigcup_{n\in\mathbb{N}} \mathcal{I}_n$. In addition, we let $\mathcal{D} = \{D_\alpha\}_{\alpha\in\mathcal{I}}$ be the set of domains (which is again the same for any $\Gamma \in \mathcal{T}$).

We let $\mathsf{Partial}(\Gamma')$ denote the set of oracle queries that our adversary $\mathcal{A}$ will sample in each iteration. We let Q denote the set of actual queries that made by $\mathcal{A}$ to the true oracle Γ. We write, e.g., $[\mathcal{O}_n(C, r) = \widehat{C}] \in Q$ to denote that Q contains an $\mathcal{O}_n$-query with input (C, r) and output $\widehat{C}$. Likewise, $[f_n(x) = y] \in \mathsf{Partial}(\Gamma')$ denotes that there is some f_n query in $\mathsf{Partial}(\Gamma')$ with input x and output y. We also 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.

The Set of Queries/Answers that the Adversary Samples. Our adversary $\mathcal{A}$ will sample in each iteration some oracle queries/answers $\mathsf{Partial}(\Gamma') = (f', \mathcal{O}', \mathsf{Eval}')$ that are consistent with the actual queries Q it made so far. However, 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 adversary samples. (Looking ahead, by doing so, we will be able to construct a hybrid oracle $\widetilde{\Gamma}$.) Formally, we define:

Definition 3.9. (*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 $\left[\mathsf{Eval}_m(\widehat{C}, \star) = \beta\right] \in \mathsf{Eval}'$ with $\beta \neq \perp$, 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 $\beta \neq \perp, |\widehat{C}| = 10m$ and $|a| = m$, let $\left[\mathcal{O}_m(C, r) = \widehat{C}\right] \in \mathcal{O}'$ that is guaranteed to exist by the previous requirement.

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.

Augmented Oracle Queries. For the analysis, we consider the queries that are associated with the execution of $\mathsf{P}^{\Gamma}(\alpha, x^*) = y^*$, for some $\alpha \in \mathcal{I}$. In fact, the set that we consider may contain some additional queries that do not necessarily appear in the execution of $\mathsf{P}^{\Gamma}(\alpha, x^*)$, but are still associated with this execution. Let $\mathsf{RealQ}(\Pi, \Gamma, \alpha, x^*)$ denote the set of actual queries to Γ in the evaluation of $\mathsf{P}^{\Gamma}(\alpha, x^*)$.

We define:

Definition 3.10. (*Augmented oracle queries*) The set of extended queries, denoted $\mathsf{AugQ}(\Pi, \Gamma, \alpha, x^*)$, consists of the following queries:

1. All the queries in $\mathsf{RealQ}(\Pi, \Gamma, \alpha, x^*)$.
2. For every query $[\mathsf{Eval}_m(\widehat{C}, a) = \beta] \in \mathsf{RealQ}(\Pi, \Gamma, \alpha, x^*)$ with $|\widehat{C}| = 10m$, $|a| = m$ and $\beta \neq \perp$, let $C, r \in \{0,1\}^m$ be the unique pair such that $\mathcal{O}_m(C, r) = \widehat{C}$. Then, the set $\mathsf{AugQ}(\Pi, \Gamma, \alpha, x^*)$ contains also the query $[\mathcal{O}_m(C, r) = \widehat{C}]$ and all the f-queries/answers sufficient for to the evaluation of $C^f(a)$.

Note that these additional queries correspond to the consistent oracle queries/answers that the adversary samples in the attack, as in Definition 3.9. We do not explicitly require the first requirement of Definition 3.9 here. This is because our analysis focuses on the case where there is no Eval query on an obfuscated circuit $\widehat{C}$ that is not an output of a previous $\mathcal{O}$-query.

Looking ahead, all the circuits that will be evaluated by the oracle Eval are of some polynomial size in the security parameter, and therefore, each evaluation adds some polynomial number of oracle queries to f. Therefore, the overall size of $\mathsf{AugQ}(\Pi, \Gamma, \alpha, x^*)$ is some polynomial. Let $\ell = \ell(n) > n$ be an upper bound of $|\mathsf{AugQ}(P, \widetilde{\Gamma}, x)|$ for all possible $\widetilde{\Gamma} \in \mathcal{T}$ and all $x \in D_\alpha$.

Equipped with the above notation and definitions, we are now ready to prove Theorem 3.6.

Proof of Theorem 3.6. Let $\Pi = (\mathsf{Gen}, \mathsf{Samp}, \mathsf{P})$ be an oracle-aided permutation family that is domain invariant relative to the support of the distribution $\mathfrak{S}$. Consider the following oracle-aided algorithm $\mathcal{A}$:

The Algorithm $\mathcal{A}$.

- **Input:** An index $\alpha \in \mathcal{I}$ and a value $y^* \in D_\alpha$.
- **Oracle Access:** The oracle Γ.
- **The Algorithm:**
  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.)
  2. **Avoiding spoof_m for small m.** Let $t = \log(16\ell)$. The adversary $\mathcal{A}$ queries the oracle f_m on all inputs $|x| = m$ for all $m \leq t$. It queries $\mathcal{O}_m(C, r)$ for all $|C| = |r| = m \leq t$; and queries $\mathsf{Eval}_m(\widehat{C}, a)$ on all $m \leq t$ with $|\widehat{C}| = 10m$ and $|a| = m$. Denote this set of queries by Q^*.
  3. Run the following for $\ell + 1$ iterations:

(a) **Simulation Phase:** $\mathcal{A}$ finds a value $x' \in D_\alpha$ and a set $\mathsf{Partial}(\Gamma')$ of consistent oracle queries/answers that is consistent with the list of queries/answers Q, such that $\mathsf{P}^{\mathsf{Partial}(\Gamma')}(\alpha, x') = y^*$ as follows:[8]

i. $\mathcal{A}$ samples an oracle $\Gamma' = (f', \mathcal{O}', \mathsf{Eval}')$ uniformly at random from the set of all oracles that are consistent with Q. That is, f' and $\mathcal{O}'$ are sampled uniformly at random conditioned on Q, and then Eval' is defined accordingly.

ii. $\mathcal{A}$ inverts y^* relative to Γ'. Specifically, $\mathcal{A}$ enumerates over $\mathcal{D}_\alpha$ and find the unique input $x' \in D_\alpha$ for which $\mathsf{P}^{\Gamma'}(\alpha, x') = y^*$.

iii. $\mathcal{A}$ sets $\mathsf{Partial}(\Gamma')$ to be all the queries in Q, and all the queries included in the evaluation of $\mathsf{P}^{\Gamma'}(\alpha, x')$.

(b) **Evaluation Phase:** The adversary evaluates $\mathsf{P}^{\Gamma}(\alpha, x')$. If the output of the evaluation is y^*, it halts and outputs x'.

(c) **Update Phase:** Otherwise, $\mathcal{A}$ makes all the queries in $\mathsf{Partial}(\Gamma') \setminus Q$ to the true oracle Γ, and continues to the next iteration.

4. In case the adversary has not halted yet, it outputs $\perp$.

Analysis. We show that in each iteration the adversary either finds x^* or learns some query associated with the evaluation $\mathsf{P}^{\Gamma}(\alpha, x^*)$. We now define these two "bad" events and show that they occur with small probability. These two events play a central role in our analysis. We then proceed to the analysis, showing that in every iteration in which these two events do not occur, the algorithm either inverts y^* or succeeds to learn at least one new query that appears in $\mathsf{AugQ}(\Pi, \Gamma, \alpha, x^*)$.

The Event spoof. For any $m \in \mathbb{N}$, define spoof_m to be the event where

$$\begin{aligned} &\left[\mathsf{Eval}_m(\widehat{C}, a) \neq \perp\right] \in \mathsf{AugQ}(\Pi, \Gamma, \alpha, x^*), \\ &\text{but} \quad \left[\mathcal{O}_m(\star, \star) = \widehat{C}\right] \notin \mathsf{AugQ}(\Pi, \Gamma, \alpha, x^*) \cup Q^*. \end{aligned}$$

Let $\mathsf{spoof}_\Gamma = \bigvee_m \mathsf{spoof}_m$. By construction, Q^* contains all possible $\mathcal{O}_m$-queries for every $m \leq t$, and therefore, spoof_m cannot occur for $m \leq t$. Moreover, by Claim 3.8, we have that

$$\begin{aligned} \Pr\left[\,\mathsf{spoof}_\Gamma\,\right] &\leq \Pr\left[\,\bigvee\nolimits_m \mathsf{spoof}_m\,\right] \leq \sum_{m=t}^{\infty} \Pr\left[\,\mathsf{spoof}_m\,\right] \\ &\leq \sum_{m=\log 16\ell}^{\infty} \ell \cdot 2^{-8m} \leq 2 \cdot \ell \cdot 2^{-8\log 16\ell} \leq 2^{-31}. \end{aligned}$$

Let spoof'_m be the event where the adversary $\mathcal{A}$ queries the real oracle Γ some query $[\mathsf{Eval}_m(\widehat{C}, \star)]$, receives a value differ than $\perp$, but $\widehat{C}$ was not an output of Γ on some

[8] Note that the set of queries/answers $\mathsf{Partial}(\Gamma')$ may be inconsistent with the true oracle Γ on all queries $\mathsf{Partial}(\Gamma') \setminus Q$.

previous query of $\mathcal{A}$ to $\mathcal{O}_m$. Let $\mathsf{spoof}_{\mathcal{A}} = \bigvee_m \mathsf{spoof}'_m$. Similarly to the above,[9] the probability of $\mathsf{spoof}_{\mathcal{A}}$ is bounded by 2^{-31}. Finally, we let $\mathsf{spoof} = \mathsf{spoof}_{\Gamma} \vee \mathsf{spoof}_{\mathcal{A}}$, and this probability is bounded by 2^{-30}.

The Event fail. The second bad event that we consider is the event fail. This event occurs whenever $\mathcal{A}$ samples an oracle Γ' that has some contradiction with the oracle Γ, and therefore, the hybrid oracle $\widetilde{\Gamma}$ cannot be constructed.

Let $\mathcal{T}(Q)$ be the set of all oracles Γ' that are consistent with Q. (Namely, each query in Q is answered the same for all $\Gamma' \in \mathcal{T}(Q)$, with the same answer as Γ.) In each iteration, the adversary $\mathcal{A}$ samples the oracle Γ' which is consistent with the true oracle queries Q. Let $\mathcal{T}$-$\mathsf{admissible}$ denote the set of "valid" oracles that $\mathcal{A}$ may sample; the set $\mathcal{T}$-$\mathsf{admissible}$ contains all oracles $\Gamma' = (f', \mathcal{O}', \mathsf{Eval}')$ such that:

- Γ' *is consistent with* Q.
- Γ' *avoids the outputs of* $\mathcal{O}$. For every $m \in \mathbb{N}$, the true oracle $\mathcal{O}_m$ and the sampled oracle $\mathcal{O}'_m$ should have disjoint outputs (except for the queries in Q). Formally, let $Q_m^{\mathcal{O}} = \{x \in \{0,1\}^{2m} \mid [\mathcal{O}_m(x) = \star] \in Q\}$. Then, we require that for every $x, y \notin Q_m^{\mathcal{O}}$ it holds that $\mathcal{O}_m(x) \neq \mathcal{O}'_m(y)$.
- Γ' *avoids invalid* Eval*-queries.* That is, for every $\left[\mathsf{Eval}_m(\widehat{C}, a) = \perp\right] \in \mathsf{AugQ}(\Pi, \Gamma, \alpha, x^*)$, with $|\widehat{C}| = 10m$, for every $C, r \in \{0,1\}^m$ it holds that $\mathcal{O}'_m(C, r) \neq \widehat{C}$.

Notice that the first two conditions relate to the set of queries Q, whereas the third condition relates to the set $\mathsf{AugQ}(\Pi, \Gamma, \alpha, x^*)$. Moreover, note that the second condition defines $2^{2m} - |Q|$ outputs of $\mathcal{O}'_m$ that are invalid, and the third condition defines at most q invalid outputs. Therefore, there are overall at most 2^{2m} outputs of $\mathcal{O}'_m$ that are invalid.

Note that between iterations, the set Q varies. We define by $\mathsf{Invalid\text{-}Im}_m^{(i)}$ the set of all invalid outputs for $\mathcal{O}'_m$, in the ith iteration. In all iterations, the set $\mathsf{Invalid\text{-}Im}_m^{(i)}$ is bounded by 2^{2m}.

Let $\mathsf{fail}_m^{(i)}$ denote the event where $\mathcal{A}$ samples an invalid oracle $\mathcal{O}'_m$ in some iteration i. Let $\mathsf{fail}^{(i)} = \bigvee_m \mathsf{fail}_m^{(i)}$, and let $\mathsf{fail} = \bigvee_i \mathsf{fail}^{(i)}$. For every m, we have that:

$$\Pr_{\mathcal{O}'_m}\left[\mathsf{fail}_m^{(i)}\right] = \Pr_{\mathcal{O}'_m}\left[\exists x \in \{0,1\}^{2m} \text{ s.t. } \mathcal{O}'_m(x) \in \mathsf{Invalid\text{-}Im}_m^{(i)}\right]$$
$$\leq 2^{2m} \cdot \frac{\left|\mathsf{Invalid\text{-}Im}_m^{(i)}\right|}{2^{10m} - 2^{2m}} \leq 2^{-5m}.$$

As a result, we get that the probability that sampling $\mathcal{O}$ fails for some length $m > t$ is bounded by

$$\Pr_{\mathcal{O}'}\left[\mathsf{fail}^{(i)}\right] \leq \sum_{m=t}^{\infty} 2^{-5m} \leq 2 \cdot 2^{-5t}.$$

[9] This holds since y^* and α due to the domain invariance property. In particular, the property enables sampling y^* and α independently of Γ, and therefore, we can apply Claim 3.8.

We therefore conclude that the probability that in some of the $\ell + 1$ iterations, the adversary $\mathcal{A}$ samples some oracle $\Gamma' \notin \mathcal{T}$-$\mathsf{admissible}$ is bounded by

$$\Pr\left[\,\mathsf{fail}\,\right] \leq \sum_{i=1}^{\ell+1} \Pr\left[\,\mathsf{fail}^{(i)}\,\right] \leq (\ell+1)\cdot 2\cdot 2^{-5t} = 2(\ell+1)\cdot\left(2^{-4}\cdot\ell^{-1}\right)^5 \leq 2^{-19},$$

where recall that $t = \log(16\ell)$. We are now ready for the main claim of the analysis.

Claim 3.11. *Assume that* fail *and* spoof *do not occur. Then, in every iteration at least one of the following occurs:*

1. *$\mathcal{A}$ finds the pre-image x^* such that $\mathsf{P}^{\Gamma}(\alpha, x^*) = y^*$.*
2. *During the update phase, $\mathcal{A}$ queries Γ with at least one of the queries in $\mathsf{AugQ}(\Pi, \Gamma, \alpha, x^*)$.*

Proof. Assume that neither one of the above conditions holds. Then, we show that there exists an oracle $\widetilde{\Gamma} \in \mathcal{T}$ that behaves like the true oracle Γ on $\mathsf{P}^{\widetilde{\Gamma}}(\alpha, x^*) = \mathsf{P}^{\Gamma}(\alpha, x^*) = y^*$, and on the other hand, it behaves like Γ' in the evaluation of $\mathsf{P}^{\widetilde{\Gamma}}(\alpha, x') = \mathsf{P}^{\mathsf{Partial}(\Gamma')}(\alpha, x') = y^*$. According to this oracle $\widetilde{\Gamma}$, the following hold:

1. Since Π is a domain-invariant construction, and since $\widetilde{\Gamma} \in \mathcal{T}$, there exists some randomness $r \in \{0,1\}^*$ such that $\mathsf{Gen}^{\widetilde{\Gamma}}(1^n; r) = \alpha$.
2. Since Π is a domain-invariant construction, it holds that $\mathrm{Im}(\mathsf{Samp}^{\widetilde{\Gamma}}(\alpha)) = \mathrm{Im}(\mathsf{Samp}^{\Gamma}(\alpha)) = \mathrm{Im}(\mathsf{Samp}^{\mathsf{Partial}(\Gamma')}(\alpha)) = D_\alpha$. As a result, there exists some randomness $r' \in \{0,1\}^*$ such that $\mathsf{Samp}^{\widetilde{\Gamma}}(\alpha; r') = x'$ and $\mathsf{Samp}^{\widetilde{\Gamma}}(\alpha; r^*) = x^*$.
3. As mentioned above, $\mathsf{P}^{\widetilde{\Gamma}}(\alpha, x') = y^*$ and $\mathsf{P}^{\widetilde{\Gamma}}(\alpha, x^*) = y^*$.

Since the first condition in the statement does not hold, we conclude that $x' \neq x^*$ but still $\mathsf{P}^{\widetilde{\Gamma}}(\alpha, x') = \mathsf{P}^{\widetilde{\Gamma}}(\alpha, x^*)$, in contradiction to the assumption that $\mathsf{P}^{\widetilde{\Gamma}}(\alpha, \cdot)$ defines a permutation.

We now show that the oracle $\widetilde{\Gamma} = (\widetilde{f}, \widetilde{\mathcal{O}}, \widetilde{\mathsf{Eval}})$ as above can be constructed. Recall that we assume that the both conditions of the statement of the claim do not hold, and therefore, in particular it holds that $\mathsf{AugQ}(\Pi, \Gamma, \alpha, x^*) \cap \mathsf{Partial}(\Gamma') \subseteq Q$.

The Oracle $\widetilde{f}$. Note that 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 $[f_m(x) = y'] \in \mathsf{AugQ}(\Pi, \Gamma, \alpha, x^*)$, we set $\widetilde{f}_m(x) = y'$. For every $[f_m(x) = y] \in \mathsf{Partial}(\Gamma')$, we set $\widetilde{f}_m(x) = y$. Since $\mathsf{AugQ}(\Pi, \Gamma, \alpha, x^*) \cap \mathsf{Partial}(\Gamma') \subseteq Q$, we have that there is no contradiction, i.e., there are no input x and outputs y, y' such that $y \neq y'$ and $[f_m(x) = y'] \in f'$ and $[f_m(x) = y] \in \mathsf{AugQ}(\Pi, \Gamma, \alpha, x^*)$. For any other value $x \notin \mathsf{Partial}(\Gamma') \cap \mathsf{AugQ}(\Pi, \Gamma, \alpha, x^*)$, 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

$$\mathsf{avoid}\text{-}\mathcal{O}_m = \left\{\widehat{C} \in \{0,1\}^{10m} \;\middle|\; \exists\, [\mathsf{Eval}_m(\widehat{C}, \star) = \star] \in \mathsf{AugQ}(\Pi, \Gamma, \alpha, x^*) \cup \mathsf{Partial}(\Gamma')\right\}.$$

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 $[\mathcal{O}_m(x) = y] \in \mathsf{AugQ}(\Pi, \Gamma, \alpha, x^*)$, we set $\widetilde{\mathcal{O}}_m(x) = y$. Likewise, for every $[\mathcal{O}_m(x) = y] \in \mathsf{Partial}(\Gamma')$, we set $\widetilde{\mathcal{O}}_m(x) = y$. Since $\mathsf{AugQ}(\Pi, \Gamma, \alpha, x^*) \cap \mathsf{Partial}(\Gamma') \subseteq Q$, we have that there is no contradiction, that is, there is no pre-image that has two possible outputs. Moreover, since fail does not occur, it holds that $\Gamma' \in \mathcal{T}\text{-}\mathsf{admissible}$, the two functions $\mathcal{O}_m$ and (the partially defined function) $\mathcal{O}'_m$ do not evaluate to the same output, and so the partially defined function $\widetilde{\mathcal{O}}_m$ is injective. We continue to define $\widetilde{\mathcal{O}}_m$ on the additional values, such that $\widetilde{\mathcal{O}}_m$ is injective and avoids the set $\mathsf{avoid}\text{-}\mathcal{O}_m$.

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 $\mathsf{AugQ}(\Pi, \Gamma, \alpha, x^*)$ and $\mathsf{Partial}(\Gamma')$. That is, that every query $[\mathsf{Eval}_m(\star, \star)] \in \mathsf{AugQ}(\Pi, \Gamma, \alpha, x^*) \cup \mathsf{Partial}(\Gamma')$ has the same answer with $\widetilde{\mathsf{Eval}}$, and therefore, $\mathsf{P}^{\Gamma}(\alpha, x^*) = \mathsf{P}^{\widetilde{\Gamma}}(\alpha, x^*)$ and $\mathsf{P}^{\Gamma'}(\alpha, x') = \mathsf{P}^{\widetilde{\Gamma}}(\alpha, x')$. We have:

1. Assume that there exists $[\mathsf{Eval}(\widehat{C}, a) = \beta] \in \mathsf{Eval}'$ for some $\beta \neq \perp$. Since the oracle $\mathsf{Partial}(\Gamma') = (f', \mathcal{O}', \mathsf{Eval}')$ is consistent (recall Definition 3.9), then there exists a query $\left[\mathcal{O}_m(C, r) = \widehat{C}\right] \in \mathsf{Partial}(\Gamma')$ 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 $\mathsf{Partial}(\Gamma')$ has 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 $[\mathsf{Eval}(\widehat{C}, a) = \beta] \in \mathsf{AugQ}(\Pi, \Gamma, \alpha, x^*)$ for some $\beta \neq \perp$. Since spoof does not occur, there exists a query $[\mathcal{O}(C, r) = \widehat{C}] \in \mathsf{AugQ}(\Pi, \Gamma, \alpha, x^*)$ as well, and $\mathsf{AugQ}(\Pi, \Gamma, \alpha, x^*)$ contains also all the f-queries necessary for the evaluation $C^f(a)$. Since these queries appear in $\mathsf{AugQ}(\Pi, \Gamma, \alpha, x^*)$, 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 $[\mathsf{Eval}(\widehat{C}, a) = \perp] \in \mathsf{Partial}(\Gamma') \cup \mathsf{AugQ}(\Pi, \Gamma, \alpha, x^*)$ we show that $\widetilde{\mathsf{Eval}}(\widehat{C}, a) = \perp$ as well. Specifically
   (a) If $[\mathsf{Eval}(\widehat{C}, a) = \perp] \in \mathsf{Partial}(\Gamma')$ then $\widehat{C} \notin \mathrm{Im}(\mathcal{O}')$. Then there is a contradiction only if there exist C and r such that $[\mathcal{O}(C, r) = \widehat{C}] \in \mathsf{AugQ}(\Pi, \Gamma, \alpha, x^*)$. (If the query does not appear in $\mathsf{AugQ}(\Pi, \Gamma, \alpha, x^*)$, then there is no contradiction as $\widehat{C} \in \mathsf{avoid}\text{-}\mathcal{O}$ and therefore $\widehat{C} \notin \mathrm{Im}(\widetilde{\mathcal{O}})$.) However, if such a query exists, then at the end of this iteration the event spoof' occurs: $\mathcal{A}$ queries Eval on a value which is differ than $\perp$ without receiving $\widehat{C}$ as an output of a previous $\mathcal{O}$-query.
   (b) If $[\mathsf{Eval}'(\widehat{C}, a) = \perp] \in \mathsf{AugQ}(\Pi, \Gamma, \alpha, x^*)$ then $\widehat{C} \notin \mathrm{Im}(\mathcal{O})$. Then, there is a contradiction only if there exists C and r such that $[\mathcal{O}'(C, r) = \widetilde{C}] \in \mathsf{Partial}(\Gamma')$. However, this implies that the event fail occurs.

This completes the proof of Claim 3.11. □

From Claim 3.11 we conclude that:

$$\Pr_{\substack{\Gamma\leftarrow\mathfrak{S}\\ \alpha\leftarrow\mathsf{Gen}^{\Gamma}(1^n)\\ x^*\leftarrow\mathsf{Samp}^{\Gamma}(\alpha)}}\left[\mathcal{A}^{\Gamma}(\alpha,\mathsf{P}^{\Gamma}(\alpha,x^*))=x^* \mid \overline{\mathsf{fail}}\wedge\overline{\mathsf{spoof}}\right]=1.$$

Since $\Pr\left[\mathsf{fail}\right]+\Pr\left[\mathsf{spoof}\right]\leq 2^{-10}$, it holds that:

$$\Pr_{\substack{\Gamma\leftarrow\mathfrak{S}\\ \alpha\leftarrow\mathsf{Gen}^{\Gamma}(1^n)\\ x^*\leftarrow\mathsf{Samp}^{\Gamma}(\alpha)}}\left[\mathcal{A}^{\Gamma}(\alpha,\mathsf{P}^{\Gamma}(\alpha,x^*))=x^*\right]\geq 1-2^{-10}.$$

Finally, we observe that $\mathcal{A}$ makes at most a polynomial number of oracle queries to Γ, and all other computations that are done by $\mathcal{A}$ can be done using a polynomial number of queries to a PSPACE-complete oracle (as in the work of Impagliazzo and Rudich [42]): In each iteration, sampling x' and $\mathsf{Partial}(\Gamma')$ can be done in polynomial space, requires access only to Q which is of polynomial size, and does not require access to Γ. This results in a polynomial-time adversary. □

3.4. *f is a One-Way Function Relative to* $\boldsymbol{\Gamma}$

In this section we prove that f is one way relative to the oracle Γ. This is a rather standard proof, relying on the fact that each query to Eval leads to a bounded number of queries to f. We first define the notion of a q-query algorithm, and then prove that f is one way relative to Γ for $q(n)=2^{\Theta(n)}$.

Definition 3.12. Let $\mathcal{A}$ be an oracle-aided algorithm that interacts with the oracle Γ. Then $\mathcal{A}$ is a $(\mathsf{q}_{\mathsf{f}},\mathsf{q}_{\mathcal{O}},\mathsf{q}_{\mathsf{Eval}})$-query algorithm if for every $n\in\mathbb{N}$ and for every input $y\in\{0,1\}^n$, the algorithm $\mathcal{A}(y)$ makes at most $q_f(n)$,$q_{\mathcal{O}}(n)$ and $q_{\mathsf{Eval}}(n)$ queries to the oracle f, $\mathcal{O}$ and Eval, respectively, and its queries to Eval are with circuits of size at most $q_{\mathsf{Eval}}(n)$. In addition, $\mathcal{A}$ is a q-query algorithm if it is (q,q,q)-query algorithm.

Claim 3.13. *For any q-query algorithm $\mathcal{A}$ it holds that:*

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

for any $n\in\mathbb{N}$, where the probability is taken over the choice of $\Gamma\leftarrow\mathfrak{S}$, $x\leftarrow\{0,1\}^n$, and over the internal randomness of $\mathcal{A}$.

Proof. We show that the claim in fact holds for any fixing of $\mathcal{O}$ and $f_{-n}=\{f_m\}_{m\in\mathbb{N},m\neq n}$, where the probability is taken over the choice of $x\leftarrow\{0,1\}^n$, $f_n:\{0,1\}^n\to\{0,1\}^n$ and over the internal randomness of $\mathcal{A}$. An execution of a q-query algorithm $\mathcal{A}$ can be simulated using an adversary $\mathcal{B}$ that makes at most $q(n)^2$ oracle queries to f_n, an unlimited number of queries to f_{-n} and $\mathcal{O}$, and no oracle queries to

Eval. Specifically, $\mathcal{B}$ follows the computation of $\mathcal{A}^{\Gamma}(f_n(x))$ and responds to its oracle queries as follows:

- Whenever $\mathcal{A}$ queries f or $\mathcal{O}$, the algorithm $\mathcal{B}$ simply forwards the query and delivers back the result.
- Whenever $\mathcal{A}$ queries Eval with some input $(\widehat{C}, a) \in \{0,1\}^{10m} \times \{0,1\}^m$, for some $m \in \mathbb{N}$, the algorithm $\mathcal{B}$ enumerates over all possible pair $(C, r) \in \{0,1\}^{2m}$ and check whether $\mathcal{O}_m(C, r) = \widehat{C}$. If so, it evaluates the circuit $C^f(a)$, which may lead to at most additional q queries to f and returns the result to $\mathcal{A}$. Otherwise, it returns $\perp$ to $\mathcal{A}$.

Since $\mathcal{A}$ is a $q(n)$-query algorithm, it makes (by definition) at most $q(n)$ queries to f_n and to Eval, and each Eval query is bounded to circuits of size at most $q(n)$. Therefore, $\mathcal{B}$ makes at most $q(n)^2$ queries to f_n. Since f_n is a random function, any such $\mathcal{B}$ can outputs an inverse of $f_n(x)$ with probability at most $q(n)^2/2^n$. □

3.5. *$i\mathcal{O}$ is an Indistinguishability Obfuscator Relative to* $\mathbf{\Gamma}$

In this section we show that relative to Γ there exists an exponentially secure indistinguishability obfuscator $i\mathcal{O}$ for the class $\mathcal{C}$ of all polynomial-time oracle-aided circuits C^f. We first formally describe the construction of the obfuscator, and then prove its security relative to Γ.

Construction 3.14. *The (randomized) algorithm $i\mathcal{O}(C)$ is given as input an oracle-aided circuit C (represented as n-bit string for some $n \in \mathbb{N}$), chooses a random $r \leftarrow \{0,1\}^n$, and computes $\mathcal{O}_n(C, r) = \widehat{C}$. Then, it outputs the oracle-aided circuit* $\mathsf{Eval}(\widehat{C}, \cdot)$.

Functionality. From the fact that $\mathcal{O}$ is an injective function, it is easy to see that the construction always preserves the functionality of the underlying circuit C. That is, for any oracle-aided circuit C and for any $r \in \{0,1\}^*$ it holds that C and $i\mathcal{O}(C; r)$ are functionally equivalent with respect to any function f.

Indistinguishability. Since the output of $i\mathcal{O}$ is a circuit consists of a single Eval-gate with some hardwired value $\widehat{C}$, when proving indistinguishability of two obfuscated circuits, we show indistinguishability of these two hardwired values and ignore the wrapping Eval-gate (i.e., the wrapping circuits). We prove the following theorem:

Theorem 3.15. *For every $n \in \mathbb{N}$ and for every $2^{n/4}$-query algorithm $\mathcal{A}$, it holds that*

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

where the probability is taken over the choice of $\Gamma \leftarrow \mathcal{G}$.

In fact, we show that the above holds for any fixing of the functions f and $\mathcal{O}_{-n} = \{\mathcal{O}_m\}_{m \in \mathbb{N}, m \neq n}$. From this point and forward, we fix $n \in \mathbb{N}$ and the functions $f, \mathcal{O}_{-n}$.

The proof of Theorem 3.15 consists of two somewhat independent parts. First, in Sect. 3.5.1 we show that the evaluation oracle Eval does not provide the adversary

with any significant capabilities, and it can almost always be simulated by the adversary itself. Specifically, since the output space of the function $\mathcal{O}_n$ is much larger than its input space, the adversary should not be able to find a valid output of $\mathcal{O}_n$ without querying it beforehand. As a result, with an overwhelming probability, all queries to Eval on values that were not obtained from previous queries to $\mathcal{O}_n$ can be replied to with $\perp$.

Then, in Sect. 3.5.2, we show that the only way in which an adversary can obtain any advantage in the experiment $\mathsf{Exp}^{\mathsf{iO}}_{\Gamma,i\mathcal{O},\mathcal{A},\mathcal{C}}(n)$ without accessing the evaluation oracle Eval is by "hitting" the randomness r^* used for generating the challenge obfuscated circuit in one of its $\mathcal{O}$-queries. We then show that since the adversary makes a bounded number of such queries (specifically, at most $2^{n/4}$), the probability of hitting r^* is very small.

3.5.1. *Simulating the Evaluation Oracle* Eval

The event spoof will help up show that the oracle Eval can be simulated by the adversary itself. We formally define this event and then show that it occurs with very small probability. We have:

Definition 3.16. For any oracle-aided algorithm M, consider the following event spoof_n that may occur during an execution of $M^{\Gamma}(1^n)$: The algorithm makes a query $\mathsf{Eval}_n(\widehat{C}, a)$ with $|\widehat{C}| = 10n$ whose output is not $\perp$, yet $\widehat{C}$ was not an output of a previous $\mathcal{O}_n$-query.

We prove that for any q-query algorithm (recall Definition 3.12), the event spoof_n occurs with probability that in linear in q and inverse exponential in n.

Claim 3.17. *For any $n \in \mathbb{N}$, for any f and $\mathcal{O}_{-n} = \{\mathcal{O}_m\}_{m\in\mathbb{N},m\neq n}$, and for any q-query algorithm M, the probability that spoof_n occurs in an execution of $M^{\Gamma}(1^n)$ satisfies*

$$\Pr_{\mathcal{O}_n}\left[\,\mathsf{spoof}_n\,\right] \leq q \cdot 2^{-8n}.$$

Proof. Fix M, n, f and $\mathcal{O}_{-n}$. The input space of $\mathcal{O}_n$ is of size 2^{2n}, whereas its output space is 2^{10n}. Since $\mathcal{O}_n$ is chosen uniformly at random, there are at most 2^{2n} elements in the range $\mathcal{O}_n$ and these are distributed uniformly in a space of size 2^{10n}. Any query to $\mathcal{O}_n$ reveals one point in the range of $\mathcal{O}_n$, but gives no information about other points in the range. Similarly, any Eval query may give information regarding one point in the range of $\mathcal{O}_n$, but nothing else. Therefore, the oracle queries do not give significant information regarding the range of $\mathcal{O}$, and an adversary cannot hit points in the range without previous queries to $\mathcal{O}$.

Formally, we follow the computation of $M^{\Gamma}(1^n)$. During this computation, we store a table T of oracle queries and answers for $\mathcal{O}_n$, initialized to $\emptyset$. We now show that the oracle $\mathcal{O}_n$ sampled lazily during the computation of M^{Γ}, where we reply to M's queries as follows:

- Each f_n-query of M can be answered since this oracle is fixed and this does not trigger the event spoof_n.

- With each $\mathcal{O}_n$ query on some input $(C, r) \in \{0,1\}^{2n}$, we first check in T whether (C, r) was queries before. If so—we answer the stored value in T. Otherwise, we choose a random output $\widehat{C} \in \{0,1\}^{10n}$ that does not appear in T.
- With each Eval_n-query $(\widehat{C}, a)$, with $|\widehat{C}| = 10n$ and $|a| = n$, we check whether there exists a valid pair $((C, r), \widehat{C}) \in T$ with $(C, r) \in \{0,1\}^n \times \{0,1\}^n$, or an invalid pair $(\perp, \widehat{C}) \in T$.
  If there does not exist such pairs, then we toss a coin α with probability $p_j = (2^{2n} - j)/2^{10n}$ to be 1, where j is the number of valid pairs in T. If $\alpha = 1$, we choose a uniformly random $(C, r) \in \{0,1\}^n \times \{0,1\}^n$ such that $(C, r) \notin T$ and add it to T. Otherwise, we store $(\perp, \widehat{C}) \in T$.
  Now, in T there is a pre-image of $\widehat{C}$. If this pre-image is $\perp$, we reply $\perp$ to the adversary. Otherwise, let (C, r) be the valid pre-image, and return the evaluation $C^f(a)$ to the adversary.

Note that spoof_n occurs whenever $\alpha = 1$. With each Eval_n query, this coin may be tossed at most once. Since there are at most q queries overall, using union bound the probability that in one of the queries $\alpha = 1$ is bounded by:

$$\Pr_{\mathcal{O}_n}\left[\ \mathsf{spoof}_n\ \right] \leq \frac{q}{2^{8n}}.$$

□

In what follows, recall that $\mathsf{Exp}^{\mathsf{iO}}_{\Gamma, i\mathcal{O}, \mathcal{B}, \mathcal{C}}(n)$ denotes the indistinguishability experiment associated with the obfuscator $i\mathcal{O}$ and a distinguisher $\mathcal{B}$. (See Definition 2.4.) We denote by $\mathsf{Exp}^{\mathsf{iO}}_{\Gamma, i\mathcal{O}, \mathcal{B}, \mathcal{C}}(n; b, r^*)$ this experiment when using specific values b and r^*, where b is the bit chosen in step 1 of the experiment, and r^* is the randomness used by the algorithm $i\mathcal{O}$ to obfuscating the challenge circuit in step 3.

In addition, we denote by $\widetilde{\mathsf{Exp}}^{\mathsf{iO}}_{\Gamma, i\mathcal{O}, \mathcal{B}, \mathcal{C}}(n; b, r^*)$ an execution of the experiment, where the distinguisher $\mathcal{B}$ has an oracle access to f ,$\mathcal{O}$ and Eval_{-n}, but has no access to the oracle Eval_n. We show that if there exists an algorithm $\mathcal{A}$ whose advantage in the experiment $\mathsf{Exp}^{\mathsf{iO}}_{\Gamma, i\mathcal{O}, \mathcal{A}, \mathcal{C}}(n; b, r^*)$ is ϵ, then there exists an algorithm $\mathcal{B}$ that does not use the oracle Eval_n at all, and its advantage in the experiment $\widetilde{\mathsf{Exp}}^{\mathsf{iO}}_{\Gamma, i\mathcal{O}, \mathcal{B}, \mathcal{C}}(n; b, r^*)$ is very close to ϵ.

Claim 3.18. *For every $n \in \mathbb{N}$, if there exists a q-query algorithm $\mathcal{A}$ with $q(n) \leq 2^{n/2}$ such that*

$$\left| \Pr_{\mathcal{O}_n, (b, r^*) \leftarrow \{0,1\}^{n+1}} \left[\mathsf{Exp}^{\mathsf{iO}}_{\Gamma, i\mathcal{O}, \mathcal{A}, \mathcal{C}}(n; b, r^*) = 1 \right] - \frac{1}{2} \right| > \epsilon,$$

then there exists a q^2-query algorithm $\mathcal{B}$ that does not make any queries to Eval_n, such that

$$\left| \Pr_{\mathcal{O}_n, (b, r^*) \leftarrow \{0,1\}^{n+1}} \left[\widetilde{\mathsf{Exp}}^{\mathsf{iO}}_{\Gamma, i\mathcal{O}, \mathcal{B}, \mathcal{C}}(n; b, r^*) = 1 \right] - \frac{1}{2} \right| > \epsilon - 2^{-7n}.$$

Proof. Fix n. Given the algorithm $\mathcal{A}$, we build an algorithm $\mathcal{B}$ that makes no oracle queries to Eval_n. As long as spoof_n does not occur, all oracle queries to Eval_n can be answered by the previous queries to $\mathcal{O}_n$ and some additional f_n-queries. We proceed with a formal description of $\mathcal{B}$.

The Algorithm $\mathcal{B} = (\mathcal{B}_1, \mathcal{B}_2)$. We show how to construct the algorithm $\mathcal{B} = (\mathcal{B}_1, \mathcal{B}_2)$ using the algorithm $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$. Recall that $\mathcal{B}_1$ participates in the experiment in Step 2 (before the algorithm receives challenge), and $\mathcal{B}_2$ participates in the experiment in Step 4 (i.e., after the algorithm receives the challenge).

The algorithm $\mathcal{B}_1$ is invoked on a security parameter 1^n; it invokes $\mathcal{A}_1$ on the same input and simulates all its oracle queries as follows:

- Whenever $\mathcal{A}_1$ queries its $f, \mathcal{O}_{-n}, \mathsf{Eval}_{-n}$ oracles, $\mathcal{B}_1$ just submits theses queries to its respective own oracle and gives $\mathcal{A}$ the result.
- Whenever $\mathcal{A}_1$ makes an $\mathcal{O}_n$ query, $\mathcal{B}_1$ submits the query to its own $\mathcal{O}_n$ query, but sores the query and the response.
- Whenever $\mathcal{A}_1$ makes an Eval_n query $(\widehat{C}, a)$ with $|\widehat{C}| = 10n$, the algorithm $\mathcal{B}_1$ looks whether there was a previous query of $\mathcal{O}_n$ with output $\widehat{C}$. If there was such a query, let (C, r) be the pre-image of $\mathcal{O}_n$. It evaluates $C^f(a)$ and replies $\mathcal{A}_1$ with the result. (Note that in order to evaluate this circuit, $\mathcal{B}_1$ may perform some additional queries to f.) If there was no such a pair, then it replies with $\perp$.

Whenever $\mathcal{A}_1$ outputs a pair of circuits (C_0, C_1) and state, the algorithm $\mathcal{B}_1$ outputs these same values. When $\mathcal{B}_2$ receives the challenge obfuscated circuit $\widehat{C}$ (and state), it submits them to the algorithm $\mathcal{A}_2$ and continues to simulate its oracle queries as $\mathcal{B}_1$ answers $\mathcal{A}_1$ as above with the following modification:

- Whenever $\mathcal{A}_2$ makes an Eval_n query $(\widehat{C}, a)$ where $\widehat{C}$ is the challenge obfuscated circuit that $\mathcal{B}$ has receives, the algorithm $\mathcal{B}_2$ evaluates $C_0^f(a)$ and replies with this result. This is correct since $\mathcal{A}$ is a valid algorithm that outputs two circuits C_0, C_1 that are functionally equivalent.

When $\mathcal{A}_2$ outputs a bit b', $\mathcal{B}_2$ outputs the same bit and halts.

Analysis. Clearly, the algorithm $\mathcal{B}$ makes no oracle queries to Eval_n. Moreover, it makes the same amount of oracle queries to $\mathcal{O}_n$ as $\mathcal{A}$, but may make at most q^2 queries to f_n. Therefore, $\mathcal{B}$ is a q^2-query algorithm.

Assume that the event spoof_n does not occur (where the machine M that we consider includes the challenger and the algorithm $\mathcal{A}$). Then, assuming that spoof_n does not occur, an execution of $\mathcal{B}$ without the oracle Eval_n is equivalent to an execution of $\mathcal{A}$ with the oracle, where both adversaries also have accesses to $f, \mathcal{O}_{-n}, \mathsf{Eval}_{-n}$. That is:

$$\Pr_{\substack{\mathcal{O}_n \\ (b,r^*)\leftarrow\{0,1\}^{n+1}}}\left[\widetilde{\mathsf{Exp}}^{\mathsf{iO}}_{\Gamma,i\mathcal{O},\mathcal{B},\mathcal{C}}(n;b,r^*)=1 \ \wedge \ \overline{\mathsf{spoof}_n}\right] = \Pr_{\substack{\mathcal{O}_n \\ (b,r^*)\leftarrow\{0,1\}^{n+1}}}\left[\mathsf{Exp}^{\mathsf{iO}}_{\Gamma,i\mathcal{O},\mathcal{A},\mathcal{C}}(n;b,r^*)=1 \ \wedge \ \overline{\mathsf{spoof}_n}\right]$$

Moreover, we have that

$$\Pr\left[\mathsf{Exp}^{\mathsf{iO}}_{\Gamma,i\mathcal{O},\mathcal{A},\mathcal{C}}(n;b,r^*)=1\right] \le \Pr\left[\mathsf{Exp}^{\mathsf{iO}}_{\Gamma,i\mathcal{O},\mathcal{A},\mathcal{C}}(n;b,r^*)=1 \ \wedge\ \overline{\mathsf{spoof}_n}\right] + \Pr\left[\ \mathsf{spoof}_n\ \right].$$

where the probability is taken over $\mathcal{O}_n$ and the choice $(b,r^*) \leftarrow \{0,1\}^{n+1}$. Since $\mathcal{A}$ makes at most $q(n) < 2^{n/2}$ oracle queries, from Claim 3.17 it holds that $\Pr\left[\ \mathsf{spoof}_n\ \right] \le 2^{-7n}$. We conclude:

$$\left| \Pr_{\substack{\mathcal{O}_n \\ (b,r^*)\leftarrow\{0,1\}^{n+1}}} \left[\widetilde{\mathsf{Exp}}^{\mathsf{iO}}_{\Gamma,i\mathcal{O},\mathcal{B},\mathcal{C}}(n;b,r^*)=1\right] - \frac{1}{2} \right| > \epsilon - 2^{-7n}.$$

□

3.5.2. *From Distinguishing to Hitting*

In this section we show that an algorithm can gain an advantage in the indistinguishability experiment only if it "hits" the randomness r^* which is used in the encryption of the challenge message. We then show that the probability of hitting r^* using its oracles is very small. Formally:

Definition 3.19. For a given $n \in \mathbb{N}$ and an oracle-aided algorithm $\mathcal{B} = (\mathcal{B}_1, \mathcal{B}_2)$, we consider the following events that may occur during the execution of $\widetilde{\mathsf{Exp}}^{\mathsf{iO}}_{\Gamma,i\mathcal{O},\mathcal{B},\mathcal{C}}(n;b,r^*)$.

1. Denote by $\mathsf{initialHit}$ the event where $\mathcal{B}_1$ makes some $\mathcal{O}_n$ query of the form (C,r^*) for some circuit $C \in \mathcal{C}$.
2. Let C_0, C_1 be the two circuits that $\mathcal{B}_1$ outputs in step 2 of the experiment. Denote by hit the event in which $\mathcal{B}_2$ makes $\mathcal{O}_n$-query with input (C_0,r^*) or (C_1,r^*).

Claim 3.20. *For every $n \in \mathbb{N}$, q-query algorithm $\mathcal{B}$ with $q(n) < 2^{n/2}$ that does not make any Eval_n queries, if*

$$\left| \Pr_{\substack{\mathcal{O}_n \\ (b,r^*)\leftarrow\{0,1\}^{n+1}}} \left[\widetilde{\mathsf{Exp}}^{\mathsf{iO}}_{\Gamma,i\mathcal{O},\mathcal{B},\mathcal{C}}(n;b,r^*)=1\right] - \frac{1}{2} \right| > \epsilon,$$

then

$$\Pr_{\substack{\mathcal{O}_n \\ (b,r^*)\leftarrow\{0,1\}^{n+1}}} \left[\ \mathsf{initialHit} \ \vee\ \mathsf{hit}\ \right] > \epsilon.$$

Proof. Note that

$$\Pr_{\substack{\mathcal{O}_n \\ (b,r^*)\leftarrow\{0,1\}^{n+1}}} \left[\widetilde{\mathsf{Exp}}^{\mathsf{iO}}_{\Gamma,i\mathcal{O},\mathcal{B},\mathcal{C}}(n;b,r^*)=1\right]$$

$$\leq \Pr_{\substack{\mathcal{O}_n \\ (b,r^*)\leftarrow\{0,1\}^{n+1}}}\left[\widetilde{\mathsf{Exp}}^{\mathsf{iO}}_{\Gamma,i\mathcal{O},\mathcal{B},\mathcal{C}}(n;b,r^*)=1 \mid \overline{\mathsf{initialHit}} \wedge \overline{\mathsf{hit}}\right] + \Pr_{\substack{\mathcal{O}_n \\ (b,r^*)\leftarrow\{0,1\}^{n+1}}}\left[\mathsf{initialHit} \vee \mathsf{hit}\right].$$

We prove the claim by showing that

$$\Pr_{\substack{\mathcal{O}_n \\ (b,r^*)\leftarrow\{0,1\}^{n+1}}}\left[\widetilde{\mathsf{Exp}}^{\mathsf{iO}}_{\Gamma,i\mathcal{O},\mathcal{B},\mathcal{C}}(n;b,r^*)=1 \mid \overline{\mathsf{initialHit}} \wedge \overline{\mathsf{hit}}\right]=\frac{1}{2}.$$

In order to show this, fix the entire probability space except for the oracle $\mathcal{O}_n$ on all the values $\mathcal{O}_n \backslash \mathcal{O}_n(\cdot; r^*)$, that is, we fix everything except for the answers to the oracle queries $\mathcal{O}_n(C, r^*)$ for any $C \in \{0,1\}^n$. We now show that giving this fixing, and assuming that $\mathsf{initialHit}$ does not occur, the two circuits C_0, C_1 and the state state that the algorithm $\mathcal{B}_1$ outputs in step 2 in the experiment $\widetilde{\mathsf{Exp}}^{\mathsf{iO}}_{\Gamma,i\mathcal{O},\mathcal{B},\mathcal{C}}(n;b,r^*)$ are fully determined. In particular, we show that all the oracle queries that $\mathcal{B}_1$ may produce can be answered. Specifically:

- The oracle $f, \mathcal{O}_{-n}, \mathsf{Eval}_{-n}$ are fully determined.
- On an oracle query to $\mathcal{O}_n$ can be answered. This is because $\mathsf{initialHit}$ does not occur, and therefore, $\mathcal{B}$ never makes a query $(\cdot, r^*)$ to $\mathcal{O}_n$.
- Recall that $\mathcal{B}$ never makes an Eval_n-query.

As a result, the two circuits C_0, C_1 that $\mathcal{B}_1$ outputs at step 2 of the experiment and the state state are fully determined. We now proceed with fixing the oracle $\mathcal{O}_n(\cdot; r^*)$ on all values except for $\mathcal{O}_n(C_0, r^*)$ and $\mathcal{O}_n(C_1, r^*)$. Choose two random values $t, t' \in \{0,1\}^{10n}$ for which there does not exist some $s \in \{0,1\}^{2n}$ for which $\mathcal{O}_n(s) = t$ and $\mathcal{O}_n(s) = t'$. We now consider two cases, one corresponds to $b = 0$, where we set $\mathcal{O}_n(C_0, r^*)$ to t (and $\mathcal{O}_n(C_1, r^*)$ to t'), and the other case, corresponds to $b = 1$, where we assign the opposite values (i.e., $\mathcal{O}_n(C_0, r^*) = t'$ and $\mathcal{O}_n(C_1, r^*) = t$). In both cases we give $\mathcal{B}_2$ the value t. The two cases are equally likely, but yield different values to b. We show that if $\mathcal{B}_2$ makes no hit, then its view is independent of b and it must output the same value in the two cases. In particular, we show that all queries $\mathcal{B}_2$ may query in step 4 of $\widetilde{\mathsf{Exp}}^{\mathsf{iO}}_{\Gamma,i\mathcal{O},\mathcal{B},\mathcal{C}}(n;b,r^*)$ can be answered. Specifically:

- The oracle $f, \mathcal{O}_{-n}, \mathsf{Eval}_{-n}$ are fully determined.
- All oracle queries to $\mathcal{O}_n$ can be answered, since hit does not occur, and so $\mathcal{B}_2$ never queries $\mathcal{O}_n(C_0, r^*)$ or $\mathcal{O}_n(C_1, r^*)$. We recall that all other queries to $\mathcal{O}_n$ are fully determined.
- Recall that $\mathcal{B}_2$ never makes an Eval_n query.

□

We now show that the probability that $\mathsf{initialHit}$ or hit occur is small. That is:

Lemma 3.21. *For every $n \in \mathbb{N}$ and for every q-query algorithm $\mathcal{B}$ that does not make any Eval_n queries, it holds that*

$$\Pr_{\substack{\mathcal{O}_n \\ (b,r^*)\leftarrow\{0,1\}^{n+1}}}\left[\,\mathsf{initialHit} \vee \mathsf{hit}\,\right] \leq \frac{q(n)}{2^n - q(n)}$$

Proof. Fix the entire probability space except for $\mathcal{O}_n$ and r^*. The view of the algorithm $\mathcal{B}_1$, i.e., the view of $\mathcal{B}$ prior to the challenge phase, is completely independent of r^*. Moreover, it makes at most $q(n)$-oracle queries to $\mathcal{O}_n$ and the responses to these queries are distributed uniformly in $\{0,1\}^{10n}$. Similarly, after receiving the obfuscated circuit $\widehat{C}$, the algorithm $\mathcal{B}_2$ (who does not have an oracle access to Eval_n) receives with each oracle query to $\mathcal{O}_n$ a uniformly chosen value in $\{0,1\}^{10n}$. These values do not provide any information regarding r^*, unless a direct query to r^* is performed.

Since r^* is distributed uniformly in $\{0,1\}^n$, the probability that the ith query hits r^*, both by $\mathcal{B}_1$ or by $\mathcal{B}_2$, is $1/(2^n - i)$. Thus, the success probability of $\mathcal{B}$ is bounded by $\frac{q(n)}{2^n - q(n)}$. □

3.5.3. *Concluding the Proof*

We are now ready for the proof of Theorem 3.15.

Proof of Theorem 3.15. Assume toward a contradiction that there exists a q-query algorithm $\mathcal{A}$ with $q(n) < 2^{n/4}$ queries, such that:

$$\left|\Pr_{\substack{\mathcal{O}_n \\ (b,r^*)\leftarrow\{0,1\}^{n+1}}}\left[\,\mathsf{Exp}^{\mathsf{iO}}_{\Gamma,i\mathcal{O},\mathcal{A},\mathcal{C}}(n;b,r^*)=1\,\right] - \frac{1}{2}\right| > 2^{-n/4}$$

for infinitely many n's. By Claim 3.18, this implies the existence of a Q-query algorithm $\mathcal{B}$ with $Q(n) = q(n)^2 < 2^{n/2}$, that does not make any Eval_n-query for which

$$\left|\Pr_{\substack{\mathcal{O}_n \\ (b,r^*)\leftarrow\{0,1\}^{n+1}}}\left[\,\widetilde{\mathsf{Exp}}^{\mathsf{iO}}_{\Gamma,i\mathcal{O},\mathcal{B},\mathcal{C}}(n;b,r^*)=1\,\right] - \frac{1}{2}\right| > 2^{-n/4} - 2^{-7n}$$

By Claim 3.20, this implies that

$$\left|\Pr_{\substack{\mathcal{O}_n \\ (b,r^*)\leftarrow\{0,1\}^{n+1}}}\left[\,\mathsf{initialHit} \vee \mathsf{hit}\,\right] - \frac{1}{2}\right| > 2^{-n/4} - 2^{-7n} > 2^{-n/4+1}.$$

However, this is in contradiction to Lemma 3.21, which shows that this probability is bounded by

$$\frac{Q(n)}{2^n - Q(n)} \leq \frac{2^{n/2}}{2^n - 2^{n/2}} = \frac{1}{2^{n/2} - 1} \leq 2^{-n/4+1}.$$

□

3.6. *Proof of Theorem 3.3*

Equipped with the proofs of Theorems 3.4 and 3.5, we are now ready to prove Theorem 3.3.

Proof of Theorem 3.3. Let $(\mathsf{Gen}, \mathsf{Samp}, \mathsf{P}, M, T_M, \epsilon_{M,1}, \epsilon_{M,2})$ be a fully black-box construction of a domain-invariant one-way permutation family from a one-way function f and an indistinguishability obfuscator $i\mathcal{O}$ for the class $\mathcal{C}$ of all oracle-aided polynomial-size circuits C^f. (Recall Definition 3.2.) Theorem 3.4 guarantees the existence of an oracle-aided algorithm $\mathcal{A}$ that runs in polynomial time $T_{\mathcal{A}}(n)$ such that

$$\Pr\left[\mathcal{A}^{\mathsf{PSPACE},\Gamma}(\alpha, \mathsf{P}^{\Gamma}(\alpha, x)) = x\right] \geq \epsilon_{\mathcal{A}}(n)$$

for any $n \in \mathbb{N}$, where $\epsilon_{\mathcal{A}}(n) = 1 - 2^{-10}$, and the probability is taken over the choice of $\Gamma \leftarrow \mathcal{S}$, $\alpha \leftarrow \mathsf{Gen}^{\Gamma}(1^n)$, $x \leftarrow \mathsf{Samp}^{\Gamma}(\alpha)$, and over the internal randomness of $\mathcal{A}$. Definition 3.1 then states that there are two possible cases to consider: $\mathcal{A}$ can be used either for inverting the one-way permutation f or for breaking the indistinguishability obfuscator $i\mathcal{O}$.

In the first case, we obtain from Definition 3.1 that

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

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. 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 $M' \stackrel{\mathsf{def}}{=} M^{\mathcal{A}^{\mathsf{PSPACE}}}$ as a single oracle-aided algorithm that has oracle access to Γ, implies that M' is a q-query algorithm where $q(n) = T_{M^{\mathcal{A}}}(n)$.[10] Theorem 3.5 then implies that either $2^{n/4} \leq q(n)$

[10] Recall that an algorithm that has oracle access to Γ is a q-query $\mathsf{algorithm}$ if it makes at most q queries to Γ, and each of its queries to Eval consists of a circuit of size at most q.

or $\epsilon_{M,1}\left(T_{\mathcal{A}}(n)\cdot\epsilon_{\mathcal{A}}^{-1}(n)\right)\cdot\epsilon_{M,2}(n)\leq 2^{-n/2}$. In the first sub-case, noting that $\ell(n)\leq T_M(n)$, we obtain that

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

The running time $T_{\mathcal{A}}(n)$ of the adversary $\mathcal{A}$ (when given access to a PSPACE-complete oracle) is some fixed polynomial in n, and therefore, $T_M(n)\geq 2^{\zeta n}$ for some constant $\zeta>0$. In the second sub-case, we have that $\epsilon_{M,1}\left(T_{\mathcal{A}}(n)\right)\cdot\epsilon_{M,2}(n)\leq 2^{-n/2}$, and since $T_{\mathcal{A}}(n)$ is some fixed polynomial in n (and $\epsilon_{\mathcal{A}}(n)$ is a constant) we obtain that $\epsilon_{M,1}(n^c)\cdot\epsilon_{M,2}(n)\leq 2^{-n/2}$ for some constant $c>1$.

In the second case, we obtain from Definition 3.1 that

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

for infinitely many values of $n\in\mathbb{N}$, where $\Gamma\leftarrow\mathfrak{S}$. As in the first case, viewing $M'\stackrel{\mathsf{def}}{=}M^{\mathcal{A}^{\mathsf{PSPACE}}}$ 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 3.5 then implies that either $2^{n/4}\leq q(n)$ or $\epsilon_{M,1}\left(T_{\mathcal{A}}(n)\cdot\epsilon_{\mathcal{A}}^{-1}(n)\right)\cdot\epsilon_{M,2}(n)\leq 2^{-n/4}$. As in the first case, this implies that either $T_M(n)\geq 2^{\zeta n}$ for some constant $\zeta>0$, or $\epsilon_{M,1}(n^c)\cdot\epsilon_{M,2}(n)\leq 2^{-n/4}$ for some constant $c>1$. □

4. Impossibility for Constructions Based on One-Way Functions

As discussed in Sect. 1.3.4, the known impossibility results for constructing one-way permutations based on one-way functions [45,48,57] fall short in two aspects. First, these results rule out constructions of a *single* one-way permutation and do not rule out constructions of a one-way permutation *family*. Second, these results rule out constructions that are *domain invariant* (recall Definition 3.2), and do not rule out constructions that are *not* domain invariant (such as the construction of Bitansky et al. [15]).

In this section we resolve this surprising gap by ruling out *all* fully black-box constructions of one-way permutation *families* from one-way functions—even constructions that are *not* domain invariant. In what follows we first formally define this class of reductions, and then state and prove our result.

Definition 4.1. A fully black-box construction of a one-way permutation family from a one-way function consists of a triplet of oracle-aided probabilistic polynomial-time algorithms $(\mathsf{Gen},\mathsf{Samp},\mathsf{P})$, an oracle-aided algorithm M that runs in time $T_M(\cdot)$, and functions $\epsilon_{M,1}(\cdot)$ and $\epsilon_{M,2}(\cdot)$, such that the following conditions hold:

- **Correctness:** For any function f the triplet $(\mathsf{Gen},\mathsf{Samp},\mathsf{P})$ is a permutation family relative to f (as in Definition 2.1).

- **Black-Box Proof of Security:** For any function f, for any oracle-aided algorithm $\mathcal{A}$ that runs in time $T_{\mathcal{A}} = T_{\mathcal{A}}(n)$, and for any function $\epsilon_{\mathcal{A}} = \epsilon_{\mathcal{A}}(n)$, if

$$\Pr\left[\mathcal{A}^f(\alpha, \mathsf{P}^f(\alpha, x)) = x\right] \geq \epsilon_{\mathcal{A}}(n)$$

for infinitely many values of $n \in \mathbb{N}$, where the probability is taken over the choice of $\alpha \leftarrow \mathsf{Gen}^f(1^n)$, $x \leftarrow \mathsf{Samp}^f(\alpha)$, and over the internal randomness of $\mathcal{A}$, then

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

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.

The above definition clearly captures constructions that are not domain invariant. First, it allows the support of the distribution $\mathsf{Gen}^f(1^n)$ to depend on f. Second, for each permutation index α that is produced by $\mathsf{Gen}^f(1^n)$, it allows the domain of the permutation $\mathsf{P}^f(\alpha, \cdot)$ to depend on f. For this general class of reductions, we prove the following theorem:

Theorem 4.2. *Let* $(\mathsf{Gen}, \mathsf{Samp}, \mathsf{P}, M, T_M, \epsilon_{M,1}, \epsilon_{M,2})$ *be a fully black-box construction of a one-way permutation family from a one-way function. Then, at least one of the following propertied holds:*

1. $T_M(n) \geq 2^{\zeta n}$ *for some constant* $\zeta > 0$ (*i.e., the reduction runs in exponential time*).
2. $\epsilon_{M,1}(n^c) \cdot \epsilon_{M,2}(n) \leq 2^{-\beta n}$ *for some constants* $c > 1$ *and* $\beta > 0$(*i.e., the security loss is exponential*).[11]

4.1. *Attacking Non-domain-Invariant Permutation Families in the Random-Oracle Model*

Toward proving Theorem 4.2, we generalize the attack presented in Sect. 1.3.2 from inverting any *single* oracle-aided *domain-invariant* permutation to inverting any oracle-aided one-way permutation *family*—even such families that are not domain invariant. We prove the following theorem:

Theorem 4.3. *Let* $(\mathsf{Gen}, \mathsf{Samp}, \mathsf{P})$ *be a triplet of oracle-aided probabilistic polynomial-time algorithms that is a permutation family relative to any oracle* f. *Then, there exists an oracle-aided algorithm* $\mathcal{A}$ *that makes a polynomial number of oracle queries such that for any function* f *it holds that*

$$\Pr\left[\mathcal{A}^f(\alpha, \mathsf{P}^f(\alpha, x)) = x\right] = 1$$

[11] In particular, if the adversary-dependent security loss $\epsilon_{M,1}(\cdot)$ is polynomial, then the adversary-independent security loss $\epsilon_{M,2}(\cdot)$ is exponential.

for any $n \in \mathbb{N}$, where the probability is taken over the choice of $\alpha \leftarrow \mathsf{Gen}^f(1^n)$ and $x \leftarrow \mathsf{Samp}^f(\alpha)$, and over the internal randomness of $\mathcal{A}$. Moreover, the algorithm $\mathcal{A}$ can be implemented in polynomial time given access to a PSPACE*-complete oracle.*

At a first sight, it seems that a natural approach toward proving the above theorem would be to fix some α, and then apply Rudich's attack on the single permutation $\mathsf{P}^f(\alpha, \cdot)$. That is, the adversary $\mathcal{A}$ is invoked on some index α and a value $y^* = \mathsf{P}^f(\alpha, x^*)$, chooses in each iteration a set of oracle queries f' and an input x' such that $\mathsf{P}^{f'}(\alpha, x') = y^*$, and updates its set of oracle queries Q similarly to the original proof.

However, this approach seems somewhat insufficient due to the following reasons. First, $\mathcal{A}$ may sample a function f' for which it may be that α is not in the support of $\mathsf{Gen}^{f'}(1^n)$, and therefore, $\mathsf{P}^{f'}(\alpha, \cdot)$ might not be a permutation. Second, when $\mathcal{A}$ inverts y^* relative to some function f', it may be that two domains of the permutations $\mathsf{P}^f(\alpha, \cdot)$ and $\mathsf{P}^{f'}(\alpha, \cdot)$ are completely different (maybe except for y^*), and therefore, it is unclear what guarantee we have when considering the hybrid function $\widetilde{f}$. Specifically, given that x^* is in the domain of the permutation $\mathsf{P}^f(\alpha, \cdot)$ and x' is in the domain of the permutation $\mathsf{P}^{f'}(\alpha, \cdot)$, we have no guarantee that neither x^* nor x' are in the domain of the permutation $\mathsf{P}^{\widetilde{f}}(\alpha, \cdot)$. We therefore revisit the attack and its analysis.

In particular, in addition to sampling oracle queries and an input x' such that $\mathsf{P}^{f'}(\alpha, x') = y^*$, our adversary also samples "certificates" that ensure that α is a valid index relative to f' and that x' is in the domain of the permutation $P^{f'}(\alpha, \cdot)$. Specifically, these certificates are the randomness of the algorithms Gen and Samp. Likewise, such certificates exist with respect to true pre-image x^* and the true oracle f. Given these certificates, we can construct an hybrid oracle $\widetilde{f}$ relative to which both x^* and x' are elements in the domain of the permutation $\mathsf{P}^{\widetilde{f}}(\alpha, \cdot)$, and for both it holds that $\mathsf{P}^{\widetilde{f}}(\alpha, x') = \mathsf{P}^{\widetilde{f}}(\alpha, x^*) = y^*$. In case $x' \neq x^*$, we reach a contradiction, guarantying that our algorithm learns a new query with each iteration.

Proof of Theorem 4.3. Let $(\mathsf{Gen}, \mathsf{Samp}, \mathsf{P})$ be a triplet of oracle-aided probabilistic polynomial-time algorithms that is a permutation family relative to any oracle f. Recall that for any oracle f and for every $n \in \mathbb{N}$ we denote by $\mathcal{I}_n^f$ the support of the distribution $\mathsf{Gen}^f(1^n)$, and we let $\mathcal{I}^f \stackrel{\text{def}}{=} \bigcup_{n\in\mathbb{N}} \mathcal{I}_n^f$. In addition, for any index $\alpha \in \mathcal{I}^f$ we denote by $\mathcal{D}_\alpha^f$ the domain of the permutation $\mathsf{P}^f(\alpha, \cdot)$. We proceed to a formal description of the adversary $\mathcal{A}$.

The Adversary $\mathcal{A}$.

- **Input:** An index $\alpha \in \mathcal{I}_n^f$ and an element $y^* \in D_\alpha^f$.
- **Oracle Access:** The function f.
- **The Algorithm:**
  1. Initialize an empty list Q of query/answers to f.
  2. Run the following for $q + 1$ iterations (where q denotes the total number of oracle gates in the circuits Gen, Samp and P):

(a) **Simulation Phase:** $\mathcal{A}$ finds values (r'_1, r'_2, x') and a set of queries/answers f' that is consistent with Q such that:

$$\mathsf{Gen}^{f'}(1^n; r'_1) = \alpha, \quad \mathsf{Samp}^{f'}(\alpha; r'_2) = x' \quad \text{and} \quad \mathsf{P}^{f'}(\alpha, x') = y^*.$$

(a) **Evaluation Phase:** The adversary evaluates $\mathsf{Gen}^f(r_1)$, if the output of this evaluation is α, then it proceeds to evaluate $\mathsf{Samp}^f(\alpha; r'_2)$ and $\mathsf{P}^f(\alpha, \mathsf{Samp}^f(\alpha; r'_2))$. If the output of this latter evaluation is y^*, then it outputs $\mathsf{Samp}^f(\alpha; r'_2)$ and halts.
(b) **Update Phase:** Otherwise, $\mathcal{A}$ makes all the queries in $f' \setminus Q$ to the true oracle f and continues to the next iteration.

3. In case the adversary has not halted yet, it aborts and outputs $\perp$.

We first observe that $\mathcal{A}$ makes at most a polynomial number of oracle queries to f, and all other computations that are done by $\mathcal{A}$ are clearly polynomial-time computations. Therefore, as in the work of Impagliazzo and Rudich [42], the algorithm $\mathcal{A}$ can be implemented in polynomial time given access to a PSPACE-complete oracle: Sampling f' can be done in polynomial space, requires access only to Q which is of polynomial size, and does not require access to f.

We now claim that after at most $q + 1$ iterations, the algorithm $\mathcal{A}$ is always successful in finding the pre-image x^* of y^*. This is a direct consequence of the following claim.

Claim 4.4. *In each iteration of $\mathcal{A}$'s execution, at least one of the following occurs:*

1. *$\mathcal{A}$ outputs the pre-image x^* such that $\mathsf{P}^f(\alpha, x^*) = y^*$.*
2. *During the update phase, $\mathcal{A}$ makes at least one of the f-queries that are made during the following computations:*

$$\mathsf{Gen}^f(1^n; r_1), \quad \mathsf{Samp}^f(\alpha; r_2), \quad \text{or} \quad \mathsf{P}^f(\alpha, x^*),$$

where $\mathsf{Gen}^f(1^n; r_1) = \alpha$, $\mathsf{Samp}^f(\alpha, r_2) = x^*$ and $\mathsf{P}^f(\alpha, x^*) = y^*$.

Proof. Assume that there exists an iteration in which neither one of the above events occurs. We show that this contradicts the correctness of $(\mathsf{Gen}, \mathsf{Samp}, \mathsf{P})$. Specifically, when neither one of the above events occurs, then there exists a function $\widetilde{f}$ that is consistent with f in the computations $\mathsf{Gen}^f(1^n; r_1)$, $\mathsf{Samp}^f(\alpha; r_2)$, and $\mathsf{P}^f(\alpha, x^*)$, and is consistent with f' in the computations of $\mathsf{Gen}^{f'}(1^n; r'_1)$, $\mathsf{Samp}^{f'}(\alpha; r'_2)$ and $\mathsf{P}^{f'}(\alpha, x')$. According to this oracle $\widetilde{f}$, the following hold:

1. $\alpha = \mathsf{Gen}^{\widetilde{f}}(1^n; r_1) = \mathsf{Gen}^{\widetilde{f}}(1^n; r'_1)$. Therefore, $\alpha \in \mathcal{I}_n^{\widetilde{f}}$. That is, α is a valid index with respect to $\widetilde{f}$ (and not only with respect to f and f').
2. $x^* = \mathsf{Samp}^{\widetilde{f}}(\alpha; r_2)$ and $x' = \mathsf{Samp}^{\widetilde{f}}(\alpha; r'_2)$. As a result, $x^*, x' \in \mathcal{D}_\alpha^{\widetilde{f}}$. That is, both x^* and x' are in the domain of the permutation $\mathsf{P}^{\widetilde{f}}(\alpha, \cdot)$.
3. $\mathsf{P}^{\widetilde{f}}(\alpha, x') = y^*$ and $\mathsf{P}^{\widetilde{f}}(\alpha, x^*) = y^*$.

Moreover, assuming that Condition 1 in the claim does not occur, we conclude that $x^* \neq x'$. As a result, the element y^* has two pre-images in the domain $\mathcal{D}_\alpha^{\widetilde{f}}$, in contradiction to the assumption that $\mathsf{P}^{\widetilde{f}}(\alpha, \cdot)$ is a permutation over $\mathcal{D}_\alpha^{\widetilde{f}}$.

The function $\widetilde{f}$ can be constructed as follows:

- For every input w of f that appears in one of the evaluations $\mathsf{Gen}^f(1^n; r_1) = \alpha$, $\mathsf{Samp}^f(\alpha; r_2) = x^*$, or $\mathsf{P}^f(\alpha, x^*) = y^*$, set $\widetilde{f}(w) = f(w)$.
- For every input w of f' that appears in one of the evaluations $\mathsf{Gen}^{f'}(1^n; r_1') = \alpha$, $\mathsf{Samp}^{f'}(\alpha, r_2') = x'$, or $\mathsf{P}^{f'}(\alpha, x') = y^*$, set $\widetilde{f}(w) = f(w)$.
- For every other input w, set $\widetilde{f}(w)$ arbitrarily (say, to 0).

The above is well defined since f' and f agree on all the queries Q that were made so far, and from the fact that Condition 2 in the statement does not hold, there are no further intersection queries between these two sets of evaluations. □

This completes the proof of Theorem 4.3. □

4.2. *Proof of Theorem 4.2*

Equipped with the proof of Theorem 4.3, we are now ready to prove Theorem 4.2.

Proof of Theorem 4.2. Let $(\mathsf{Gen}, \mathsf{Samp}, \mathsf{P}, M, T_M, \epsilon_{M,1}, \epsilon_{M,2})$ be a fully black-box construction of a one-way permutation family from a one-way function. Theorem 4.3 guarantees the existence of an oracle-aided algorithm $\mathcal{A}$ that runs in polynomial time $T_{\mathcal{A}}(n)$, such that for any function f it holds that

$$\Pr\left[\mathcal{A}^{\mathsf{PSPACE},f}(\alpha, \mathsf{P}^f(\alpha, x)) = x\right] = \epsilon_{\mathcal{A}}(n)$$

for any $n \in \mathbb{N}$, where $\epsilon_{\mathcal{A}}(n) = 1$, and the probability is taken over the choice of $\alpha \leftarrow \mathsf{Gen}^f(1^n), x \leftarrow \mathsf{Samp}^f(\alpha)$, and over the internal randomness of $\mathcal{A}$. Definition 4.1 then states that for any function f it holds that

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

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. This holds, in particular, when the function $f = \{f_n\}_{n\in\mathbb{N}}$ is chosen uniformly at random (i.e., for each $n \in \mathbb{N}$ we sample $f_n : \{0,1\}^n \to \{0,1\}^n$ uniformly at random).

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 algorithm, 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 $T_{\mathcal{A}}(\ell(n))$.) Since we now consider the task of inverting a random function relative to a PSPACE-

complete oracle, it holds that either $2^{\beta n} \leq T_{M^{\mathcal{A}}}(n)$ or $\epsilon_{M,1}(T_{\mathcal{A}}(n)) \cdot \epsilon_{M,2}(n) \leq 2^{-\beta n}$ for some constant $\beta > 0$ (see, for example, [42])[12].

In the first case, noting that $\ell(n) \leq T_M(n)$, we obtain that

$$2^{\beta n} \leq T_{M^{\mathcal{A}}}(n) \leq T_M(n) \cdot T_{\mathcal{A}}(\ell(n)) \leq T_M(n) \cdot T_{\mathcal{A}}(T_M(n)).$$

The running time $T_{\mathcal{A}}(n)$ of the adversary $\mathcal{A}$ is some fixed polynomial in n, and therefore, $T_M(n) \geq 2^{\zeta n}$ for some constant $\zeta > 0$. In the second case, we have that $\epsilon_{M,1}(T_{\mathcal{A}}(n)) \cdot \epsilon_{M,2}(n) \leq 2^{-\beta n}$, and since $T_{\mathcal{A}}(n)$ is some fixed polynomial in n, we obtain that $\epsilon_{M,1}(n^c) \cdot \epsilon_{M,2}(n) \leq 2^{-\beta n}$ for some constant $c > 1$. □

Acknowledgements

We thank the anonymous referees and Mohammad Hajiabadi for their helpful comments.

References

[1] P. Ananth, Z. Brakerski, G. Segev, V. Vaikuntanathan, From selective to adaptive security in functional encryption, in *Advances in Cryptology—CRYPTO '15* (2015), pp. 657–677

[2] P. Ananth, A. Jain, Indistinguishability obfuscation from compact functional encryption, in *Advances in Cryptology—CRYPTO '15* (2015), pp. 308–326

[3] G. Asharov, G. Segev, Limits on the power of indistinguishability obfuscation and functional encryption, To appear in *Proceedings of the 56th Annual IEEE Symposium on Foundations of Computer Science* (2015). Available at https://eprint.iacr.org/2015/341.pdf

[4] A. Bogdanov, C. Brzuska, On basing size-verifiable one-way functions on NP-hardness, in *Proceedings of the 12th Theory of Cryptography Conference* (2015), pp. 1–6

[5] P. Baecher, C. Brzuska, M. Fischlin, Notions of black-box reductions, revisited, in *Advances in Cryptology—ASIACRYPT '13* (2013), pp. 296–315

[6] N. Bitansky, R. Canetti, H. Cohn, S. Goldwasser, Y. Tauman Kalai, O. Paneth, A. Rosen, The impossibility of obfuscation with auxiliary input or a universal simulator, in *Advances in Cryptology—CRYPTO '14* (2014), pp. 71–89

[7] N. Bitansky, R. Canetti, Y. Tauman Kalai, O. Paneth, On virtual grey box obfuscation for general circuits, in *Advances in Cryptology—CRYPTO '14* (2014), pp. 108–125

[8] C. Brzuska, P. Farshim, A. Mittelbach, Random-oracle uninstantiability from indistinguishability obfuscation, in *Proceedings of the 12th Theory of Cryptography Conference* (2015), pp. 428–455

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

[10] Z. Brakerski, J. Katz, G. Segev, A. Yerukhimovich, Limits on the power of zero-knowledge proofs in cryptographic constructions, in *Proceedings of the 8th Theory of Cryptography Conference* (2011), pp. 559–578

[11] M. Blum, S. Micali. How to generate cryptographically strong sequences of pseudo-random bits. *SIAM J. Comput.* **13**(4), 850–864 (1984)

[12] B. Barak, M. Mahmoody-Ghidary, Merkle puzzles are optimal—an $O(n^2)$-query attack on any key exchange from a random oracle, in *Advances in Cryptology—CRYPTO '09* (2009), pp. 374–390

[13] N. Bitansky, O. Paneth, ZAPs and non-interactive witness indistinguishability from indistinguishability obfuscation, in *Proceedings of the 12th Theory of Cryptography Conference* (2015), pp. 401–427

[12] It is crucial to note that the PSPACE-complete oracle does not access the function f directly.

[14] N. Bitansky, O. Paneth, A. Rosen, On the cryptographic hardness of finding a Nash equilibrium, in *Proceedings of the 56th Annual IEEE Symposium on Foundations of Computer Science* (2015), pp. 1480–1498

[15] N. Bitansky, O. Paneth, D. Wichs, Perfect structure on the edge of chaos—trapdoor permutations from indistinguishability obfuscation, in *Proceedings of the 13th Theory of Cryptography Conference* (2016), pp. 474–502

[16] Z. Brakerski, G.N. Rothblum, Virtual black-box obfuscation for all circuits via generic graded encoding, in *Proceedings of the 11th Theory of Cryptography Conference* (2014), pp. 1–25

[17] M. Bellare, I. Stepanovs, S. Tessaro, Poly-many hardcore bits for any one-way function and a framework for differing-inputs obfuscation, in *Advances in Cryptology—ASIACRYPT '14* (2014), pp. 102–121

[18] N. Bitansky, V. Vaikuntanathan, Indistinguishability obfuscation from functional encryption, in *Proceedings of the 56th Annual IEEE Symposium on Foundations of Computer Science* (2015), pp. 171–190

[19] D. Boneh, M. Zhandry, Multiparty key exchange, efficient traitor tracing, and more from indistinguishability obfuscation, in *Advances in Cryptology—CRYPTO '14* (2014), pp. 480–499

[20] R. Canetti, S. Goldwasser, O. Poburinnaya, Adaptively secure two-party computation from indistinguishability obfuscation, in *Proceedings of the 12th Theory of Cryptography Conference* (2015), pp. 557–585

[21] Y. Chang, C. Hsiao, C. Lu, The impossibility of basing one-way permutations on central cryptographic primitives. *J. Cryptol.* **19**(1), 97–114 (2006)

[22] K. Chung, H. Lin, M. Mahmoody, R. Pass, On the power of nonuniformity in proofs of security, in *Proceedings of the 4th Innovations in Theoretical Computer Science Conference* (2013), pp. 389–400

[23] K. Chung, H. Lin, R. Pass, Constant-round concurrent zero-knowledge from indistinguishability obfuscation, in *Cryptology* ePrint Archive, Report 2014/991 (2014)

[24] R. Canetti, H. Lin, S. Tessaro, V. Vaikuntanathan, Obfuscation of probabilistic circuits and applications, in *Proceedings of the 12th Theory of Cryptography Conference* (2015), pp. 468–497

[25] R. Canetti, Y. Tauman Kalai, O. Paneth, On obfuscation with random oracles, in *Proceedings of the 12th Theory of Cryptography Conference* (2015), pp. 456–467

[26] D. Dachman-Soled, J. Katz, V. Rao, Adaptively secure, universally composable, multiparty computation in constant rounds, in *Proceedings of the 12th Theory of Cryptography Conference* (2015), pp. 586–613

[27] D. Dachman-Soled, Y. Lindell, M. Mahmoody, T. Malkin, On the black-box complexity of optimally-fair coin tossing, in *Proceedings of the 8th Theory of Cryptography Conference* (2011), pp. 450–467

[28] D. Dachman-Soled, M. Mahmoody, T. Malkin, Can optimally-fair coin tossing be based on one-way functions?, in *Proceedings of the 11th Theory of Cryptography Conference* (2014), pp. 217–239

[29] S. Goldwasser, S.D. Gordon, V. Goyal, A. Jain, J. Katz, F.-H. Liu, A. Sahai, E. Shi, H.-S. Zhou, Multi-input functional encryption, in *Advances in Cryptology—EUROCRYPT '14* (2014), pp. 578–602

[30] S. Garg, C. Gentry, S. Halevi, M. Raykova, A. Sahai, B. Waters, Candidate indistinguishability obfuscation and functional encryption for all circuits, in *Proceedings of the 54th Annual IEEE Symposium on Foundations of Computer Science* (2013), pp. 40–49

[31] S. Garg, C. Gentry, S. Halevi, M. Raykova, Two-round secure MPC from indistinguishability obfuscation, in *Proceedings of the 11th Theory of Cryptography Conference* (2014), pp. 74–94

[32] R. Gennaro, Y. Gertner, J. Katz, L. Trevisan, Bounds on the efficiency of generic cryptographic constructions. *SIAM J. Comput.* **35**(1), 217–246 (2005)

[33] Y. Gertner, T. Malkin, S. Myers, Towards a separation of semantic and CCA security for public key encryption, in *Proceedings of the 4th Theory of Cryptography Conference* (2007), pp. 434–455

[34] Y. Gertner, T. Malkin, O. Reingold, On the impossibility of basing trapdoor functions on trapdoor predicates, in *Proceedings of the 42nd Annual IEEE Symposium on Foundations of Computer Science* (2001), pp. 126–135

[35] O. Goldreich, On security preserving reductions—revised terminology, in *Cryptology* ePrint Archive, Report 2000/001 (2000)

[36] O. Goldreich, Foundations of Cryptography—Volume 1: Basic Techniques (Cambridge University Press, Cambridge, 2001)

[37] S. Garg, A. Polychroniadou, Two-round adaptively secure MPC from indistinguishability obfuscation, in *Proceedings of the 12th Theory of Cryptography Conference* (2015), pp. 614–637

[38] I. Haitner, J.J. Hoch, O. Reingold, G. 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)
[39] J. Håstad, R. Impagliazzo, L.A. Levin, M. Luby, A pseudorandom generator from any one-way function. *SIAM J. Comput.* **28**(4), 1364–1396 (1999)
[40] C. Hsiao, L. Reyzin, Finding collisions on a public road, or do secure hash functions need secret coins?, in *Advances in Cryptology—CRYPTO '04* (2004), pp. 92–105
[41] S. Hohenberger, A. Sahai, B. Waters, Replacing a random oracle: full domain hash from indistinguishability obfuscation, in *Advances in Cryptology—EUROCRYPT '14* (2014), pp. 201–220
[42] R. Impagliazzo, S. Rudich, Limits on the provable consequences of one-way permutations, in *Proceedings of the 21st Annual ACM Symposium on Theory of Computing* (1989), pp. 44–61
[43] I. Komargodski, T. Moran, M. Naor, R. Pass, A. Rosen, E. Yogev, One-way functions and (im)perfect obfuscation, in *Proceedings of the 55th Annual IEEE Symposium on Foundations of Computer Science* (2014), pp. 374–383
[44] J. Kahn, M.E. Saks, C.D. Smyth, A dual version of Reimer's inequality and a proof of Rudich's conjecture, in *Proceedings of the 15th Annual IEEE Conference on Computational Complexity, Florence, Italy, July 4–7, 2000* (2000), pp. 98–103
[45] J. Kahn, M. Saks, C.D. Smyth, The dual BKR inequality and Rudich's conjecture. *Comb. Probab. Comput.* **20**(2), 257–266 (2011)
[46] M. Luby, Pseudorandomness and Cryptographic Applications (Princeton University Press, Princeton, 1996)
[47] T. Matsuda, On the impossibility of basing public-coin one-way permutations on trapdoor permutations, in *Proceedings of the 11th Theory of Cryptography Conference*, pp. 265–290 (2014)
[48] T. Matsuda, K. Matsuura, On black-box separations among injective one-way functions, in *Proceedings of the 8th Theory of Cryptography Conference* (2011), pp. 597–614
[49] M. Mahmoody, H.K. Maji, M. Prabhakaran, On the power of public-key encryption in secure computation, in *Proceedings of the 11th Theory of Cryptography Conference* (2014), pp. 240–264
[50] M. Mahmoody, R. Pass, The curious case of non-interactive commitments—on the power of black-box vs. non-black-box use of primitives, in *Advances in Cryptology—CRYPTO '12* (2012), pp. 701–718
[51] M. Naor, M. Yung, Universal one-way hash functions and their cryptographic applications, in *Proceedings of the 21st Annual ACM Symposium on Theory of Computing* (1989), pp. 33–43
[52] R. Pass, W.D. Tseng, M. Venkitasubramaniam, Towards non-black-box lower bounds in cryptography, in *Proceedings of the 8th Theory of Cryptography Conference* (2011), pp. 579–596
[53] M.O. Rabin, Digitalized signatures and public-key functions as intractable as factorization. Technical report 212, Massachusetts Institute of Technology, Laboratory for Computer Science (1979)
[54] J. Rompel, One-way functions are necessary and sufficient for secure signatures, in *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing* (1990), pp. 387–394
[55] R.L. Rivest, A. Shamir, L.M. Adleman, A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM* **21**(2), 120–126 (1978)
[56] O. Reingold, L. Trevisan, S.P. Vadhan, Notions of reducibility between cryptographic primitives, in *Proceedings of the 1st Theory of Cryptography Conference* (2004), pp. 1–20
[57] S. Rudich, Limits on the Provable Consequences of One-way Functions. PhD thesis, EECS Department, University of California, Berkeley (1988)
[58] 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* (1998), pp. 334–345
[59] A. Sahai, B. Waters, How to use indistinguishability obfuscation: deniable encryption, and more, in *Proceedings of the 46th Annual ACM Symposium on Theory of Computing* (2014), pp. 475–484
[60] B. Waters, A punctured programming approach to adaptively secure functional encryption, in *Advances in Cryptology—CRYPTO '15* (2015), pp. 678–697
[61] H. Wee, One-way permutations, interactive hashing and statistically hiding commitments, in *Proceedings of the 4th Theory of Cryptography Conference* (2007), pp. 419–433