On Best-Possible Obfuscation*

Shafi Goldwasser† Guy N. Rothblum‡

Abstract

An obfuscator is a compiler that transforms any program (which we will view in this work as a boolean circuit) into an obfuscated program (also a circuit) that has the same input-output functionality as the original program, but is "unintelligible". Obfuscation has applications for cryptography and for software protection.

Barak *et al.* [CRYPTO 2001] initiated a theoretical study of obfuscation, which focused on *black-box obfuscation*, where the obfuscated circuit should leak no information except for its (black-box) input-output functionality. A family of functionalities that cannot be obfuscated was demonstrated. Subsequent research has showed further negative results as well as positive results for obfuscating very specific families of circuits, all with respect to black box obfuscation.

This work is a study of a new notion of obfuscation, which we call *best-possible obfuscation*. Best possible obfuscation makes the relaxed requirement that the obfuscated program leaks as little information as *any other program* with the same functionality (and of similar size). In particular, this definition allows the program to leak non black-box information. Best-possible obfuscation guarantees that *any* information that is not hidden by the obfuscated program is also not hidden by *any other* similar-size program computing the same functionality, and thus the obfuscation is (literally) the best possible. In this work we study best-possible obfuscation and its relationship to previously studied definitions. Our main results are:

1. A separation between black-box and best-possible obfuscation. We show a natural obfuscation task that can be achieved under the best-possible definition, but cannot be achieved under the black-box definition.
2. A hardness result for best-possible obfuscation, showing that strong (information-theoretic) best-possible obfuscation implies a collapse in the polynomial hierarchy.
3. An impossibility result for efficient best-possible (and black-box) obfuscation in the presence of random oracles. This impossibility result uses a random oracle to construct hard-to-obfuscate circuits, and thus it does *not* imply impossibility in the standard model.

*A preliminary version of this work appeared in *Proceedings of the 4th Theory of Cryptography Conference, Amsterdam, The Netherlands, 2007 (TCC 2007)*.

†Weizmann Institute of Science, Rehovot 76100, Israel and CSAIL, MIT, Cambridge MA 02139, USA. E-mail: shafi@theory.csail.mit.edu. Supported by NSF grant CNS-0430450, NSF grant CFF-0635297 and a Cymerman-Jakubskind award.

‡CSAIL, MIT, Cambridge MA 02139, USA. E-mail: rothblum@csail.mit.edu. Supported by NSF grant CNS-0430450 and NSF grant CFF-0635297.

1 Introduction

An open question in computer security is whether computer programs can be *obfuscated*; whether code can be made unintelligible while preserving its functionality. This question is important as obfuscation has wide-ranging applications, both for software protection and for cryptography. Beyond its theoretical importance, the question of obfuscation is of great practical importance. Numerous ad-hoc heuristical techniques are used every day by practitioners to obfuscate their code, even though many of these techniques do not supply any provable notion of security.

A theoretical study of obfuscation was initiated by Barak, Goldreich, Impagliazzo, Rudich, Sahai, Vadhan and Yang [2]. They studied several notions of obfuscation, primarily focusing on *black-box obfuscation*, in which an obfuscator is viewed as a compiler that, given *any* input program or circuit, outputs a program with the same functionality from which it is hard to find any deterministic information on the input program. Formally, black-box obfuscation requires that anything that can be efficiently computed from the obfuscated program, can also be computed efficiently from *black-box* (i.e. input-output) access to the program. Their main result was that this (strong) notion of obfuscation cannot always be achieved, as they were able to present an explicit family of circuits that provably cannot be black-box obfuscated.

Barak *et al.* [2] considered also an alternative notion of obfuscation called *indistinguishability obfuscation* that sidesteps the black-box paradigm. An indistinguishability obfuscator guarantees that if two circuits compute the same function, then their obfuscations are indistinguishable in probabilistic polynomial time. This definition avoids the black-box paradigm, and also avoids the impossibility results shown for the black-box obfuscation notion. Indeed, Barak *et al.* showed that it is simple to build *inefficient* indistinguishability obfuscators. One disadvantage of indistinguishability obfuscation is that it does not give an intuitive guarantee that the circuit "hides information". This is apparent in their proposed construction of an inefficient indistinguishability obfuscator, where a small circuit is revealed which is equivalent to the original circuit. For some functionalities, this is a great deal of information to give away.

1.1 This Work

We propose a new notion of obfuscation, *best-possible obfuscation*, that avoids the black-box paradigm, and also gives the appealing intuitive guarantee that the obfuscated circuit leaks less information than *any other* circuit (of a similar size) computing the same function. This work is a study of this new notion of best-possible obfuscation.

Instead of requiring that an obfuscator strip a program of *any* non black-box information, we require only that the (best-possible) obfuscated program leak as little information as possible. Namely, the obfuscated program should be "as private as" any other program computing the same functionality (and of a certain size). A *best-possible* obfuscator should transform any program so that anything that can be computed given access to the obfuscated program should also be computable from *any other* equivalent program (of some related size). A best-possible obfuscation may leak non black-box information (e.g. the code of a hard-to-learn function), as long as whatever it leaks is efficiently learnable from any other similar-size circuit computing the same functionality.

While this relaxed notion of obfuscation gives no absolute guarantee about what information is hidden in the obfuscated program, it does guarantee (literally) that the obfuscated code is the best possible. It is thus a meaningful notion of obfuscation, especially when we consider that programs are obfuscated every day in the real world without *any* provable security guarantee.

In this work we initiate a study of *best-possible* obfuscation. We explore its possibilities and limitations, as well as its relationship with other definitions of obfuscation that have been suggested. We formalize the best-possible requirement in Definition 2.5, by requiring that for every *efficient learner* who tries to extract information from an obfuscated circuit, there exists an *efficient simulator* that extracts similar information from any other circuit with the same functionality and of the same size. We consider both *computationally* best-possible obfuscation, where the outputs of the learner and simulator are indistinguishable with respect to *efficient* distinguishers, and *information theoretically* best-possible obfuscation (perfect or statistical), where even an *unbounded* distinguisher cannot tell the difference between the two. We emphasize that statistically or perfectly best-possible obfuscation refer to the *distinguisher*, whereas for the learner we only consider information that can be learned *efficiently* given the obfuscated circuit. This strengthens negative results.[1] Moreover, while we focus on *efficient obfuscation*, where the obfuscator's running time is polynomial in the size of the circuit being obfuscated, we find even the challenging question of achieving *inefficient* best-possible (or black-box) obfuscation to be of complexity-theoretic (and possibly practical) interest. Some of our negative results rule out even inefficient obfuscation (whenever we refer to obfuscation we mean efficient obfuscation unless we explicitly state otherwise).

Relationship with Previous Definitions. We study how best-possible obfuscation relates to black-box obfuscation, and present a separation between the two notions of obfuscation. The proof of this result also gives the first known separation between black-box and *indistinguishability* obfuscation. The separation result considers the complexity class of languages computable by polynomial sized ordered decision diagrams or POBDDs; these are log-space programs that can only read their input tape once, from left to right (see Section 3). We observe that *any* POBDD can be efficiently best-possible obfuscated *as a POBDD* (Proposition 3.2), whereas there are many natural functions computable by POBDDs that provably cannot be black-box obfuscated *as any POBDD* (Proposition 3.3). These two results give new possibility results (for best-possible and indistinguishability obfuscation), and simple natural impossibility results (for black-box obfuscation). Note that the impossibility result for black-box obfuscation only applies when we restrict the representation of the obfuscator's output to be a POBDD itself.

We also compare the notions of best-possible and indistinguishability obfuscation. Proposition 3.4 shows that any best-possible obfuscator is also an indistinguishability obfuscator. For *efficient* obfuscators the definitions are equivalent (Proposition 3.5). For inefficient obfuscation, the difference between the two definitions is sharp, as inefficient information-theoretic indistinguishability obfuscators are easy to construct (see [2]), but the existence of inefficient statistically best-possible obfuscators even for the class of languages recognizable by 3-CNF circuits (a sub-class of AC^0) implies that the polynomial hierarchy collapses to its second level.

We believe that the equivalence of these two definitions for efficient obfuscation motivates further research on both, as the "best-possible" definition gives a strong intuitive security guarantee, and the indistinguishability definition may sometimes be technically easier to work with.

Impossibility Results. We explore the limits of best-possible obfuscation. As noted above, we begin by considering information-theoretically (statistically) best-possible obfuscation. In Theorem 4.1 we show that if there exist (not necessarily efficient) statistically secure best-possible obfuscators for the simple circuit family of 3-CNF circuits (a sub-class of AC^0), then the polynomial hierarchy

[1] Our positive result on perfectly best-possible obfuscation applies also to unbounded *learners*.

collapses to its second level. Corollary 4.2 of this theorem states that also if there exists an *efficient* statistically secure indistinguishability obfuscator for the same simple circuit family, then the polynomial hierarchy collapses to its second level. This is the first impossibility result for indistinguishability obfuscation in the standard model.

We also consider best-possible obfuscation in the (programmable) random oracle model. In this model, circuits can be built using special random oracle gates that compute a completely random function. Previously, this model was considered by Lynn, Prabhakaran and Sahai [18] as a promising setting for presenting positive results for obfuscation. We show that the random oracle can also be used to prove strong *negative results* for obfuscation. In Theorem 5.1 we present a simple family of circuits with access to the random oracle, that are provably hard best-possible obfuscate. This impossibility results extends to the indistinguishability and black-box notions of obfuscation. We note that using random oracles for obfuscation was originally motivated by the hope that giving circuits access to an idealized "box" computing a random function would make it easier to obfuscate more functionalities (and eventually perhaps the properties of the "box" could be realized by a software implementation). We, on the other hand, show that the existence of such boxes (or a software implementation with the idealized properties) could actually allow the construction of circuits that are impossible to obfuscate. Although this negative result does not rule out that every circuit without random oracle gates *can* be best-possible obfuscated, we believe it is illuminating for two reasons. First, as a warning sign when considering obfuscation in the random oracle model, and secondly as its proof hints that achieving general purpose best-possible obfuscation *in the standard model* would require a significant leap (a discussion of this point appears at the end of Section 5).

1.2 Related Work

Negative Results. Hada [16] considered the problem of obfuscation, and gave negative results for obfuscating pseudo-random functions under a strong definition of obfuscation. Barak *et al.* [2] showed that black-box obfuscation cannot always be achieved. They showed this by presenting families of functions that cannot be black-box obfuscated: there exists a predicate that cannot be computed from black-box access to a random function in the family, but can be computed from (non black-box access to) a circuit implementing any function in the family. Thus they showed that there *exist* circuits that cannot be obfuscated, but it remained possible that almost any natural circuit could be obfuscated. Goldwasser and Kalai [14], showed that if the definition of obfuscation is strengthened even further with a requirement that the obfuscation leak no more information than black-box access *even in the presence of auxiliary input*, then a large class of more natural circuits cannot be obfuscated.

Positive Results. The functionalities for which obfuscation was ruled out in [2] and [14] are somewhat complex. An interesting open question is whether obfuscation can be achieved for simpler classes of functionalities and circuits. A significant amount of attention has been paid to the question of obfuscating *point functions*. A point function $I_p(x)$ is defined to be 1 if $x = p$, or 0 otherwise. Canetti [6] showed (implicitly) how to obfuscate point functions (even under a strong auxiliary-input definition), using a strong variant of the Decisional Diffie-Hellman assumption. Lynn, Prabhakaran and Sahai [18] suggested working in the random oracle model and focused on obfuscating access control functionalities (note that impossibility results of [2] and [14] extend to the random oracle model). They observed that in the random oracle model point functions can

be obfuscated, leading to obfuscation algorithms for more complex access control functionalities. Wee [23] presented a point function obfuscator based on the existence of one-way permutations that are hard to invert in a very strong sense.

Other solutions for obfuscating point functions are known if the obfuscator doesn't need to work for *every* point, but rather for a point selected at random from a distribution with some min-entropy. For this relaxed requirement Canetti, Micciancio and Reingold [8] presented a scheme that uses more general assumptions than those used by [6] (their solution is not, however, secure in the presence of auxiliary inputs). Dodis and Smith [9] were able to obfuscate proximity queries in this framework.

Hohenberger, Rothblum, shelat and Vaikuntanathan [17] gave a positive result for obfuscating a very natural and more complex cryptographic functionality (i.e. re-encryption) under a security-oriented definition.

The Random Oracle Model. The random oracle model is an idealization, in which it is assumed that all parties have oracle access to a truly random function $\mathcal{R}$. The parties can access this function by querying the random oracle at different points. The Random oracle methodology is a heuristic methodology, in which the random oracle is used for building provably secure cryptographic objects, but then, to implement the cryptographic object in the real world, the random oracle is replaced by some real function with a succinct representation. This methodology was introduced by Fiat and Shamir [11], and later formalized by Bellare and Rogaway [3].

A clear question raised by this methodology is whether the security of the cryptographic objects in an ideal world with a random oracle can be translated into security for the real-world implementation. In principle, this was answered negatively by Canetti, Goldreich and Halevi [7], who showed that there exist cryptographic schemes that are secure in the presence of a random oracle, but cannot be secure in the real world, regardless of the implementation of the random oracle. Their work left open the possibility that the random oracle methodology could still work for "natural" cryptographic practices. This was ruled out by Goldwasser and Kalai [13] for the Fiat-Shamir method [11], which uses a random oracle for obtaining digital signatures from identification schemes. The method was shown to have the potential of leading to insecure signature schemes regardless of the possible implementation of the random oracle.

In the context of obfuscation, Lynn, Prabhakaran and Sahai [18] explored which circuits could be obfuscated in the (programmable) random oracle model, where the view generated by the black-box simulator is indistinguishable when taken over a randomly selected oracle (Section 5 of this work considers the same model). They used the random oracle $\mathcal{R}$ to obfuscate a point function I_p (when p is given to the obfuscator) using the value $\mathcal{R}(p)$. On input x the obfuscated circuit outputs 1 if and only if $\mathcal{R}(x) = \mathcal{R}(p)$. The only information about p in the obfuscated circuit is the value $\mathcal{R}(p)$, and this ensures that the obfuscation does not leak any non black-box information about I_p. They then proceeded to show how to obfuscate point functions with more general outputs (on input $x = p$ the function outputs some value, and otherwise it outputs $\perp$), multi-point functions and other more complex access control circuits. Narayanan and Shmatikov [20] gave a positive result for obfuscating databases in the random oracle model. In Section 5 of this work we explore whether indeed the random oracle model is a promising setting for further work on obfuscation.

1.3 Organization

We begin by presenting notation and formal definitions in **Section 2**. We compare our new definition of obfuscation with previous definitions in **Section 3**. In **Section 4** we present impossibility results for statistically best-possible obfuscation. **Section 5** deals with obfuscation in the random oracle model, including an overview of the model, and gives impossibility results for computationally best-possible obfuscation (in the random oracle model). We conclude with discussions and extensions in **Section 6**.

2 Definitions and Discussion

2.1 Notation and Preliminaries

Notation. Let $[n]$ be the set $\{1, 2, \ldots n\}$. For $x, y \in \{0,1\}^n$ we use $x \circ y$ to denote the concatenation of x and y (a string in $\{0,1\}^{2n}$). For a (discrete) distribution D over a set X we denote by $x \sim D$ the experiment of selecting $x \in X$ by the distribution D. A function $f(n)$ is *negligible* if it smaller than any (inverse) polynomial: for any positive polynomial $p()$, there exists some n_0 such that for all $n \geq n_0$ it holds that $f(n) < \frac{1}{p(n)}$. When we say that for several input lengths $f(n) \leq neg(n)$, we mean that there exists a negligible function $neg()$ that is larger than $f()$ for those input lengths. When we say that $f() \leq neg()$ we mean that there exists a negligible function that is larger than $f()$ for all but finitely many input lengths.

Distributions, Ensembles and Indistinguishability. An ensemble $D = \{D_n\}_{n \in N}$ is a sequence of random variables, each ranging over $\{0,1\}^{\ell(n)}$ where $l : \mathbb{N} \to \mathbb{N}$. We consider only ensembles where $\ell(n)$ is polynomial in n (we occasionally abuse notation and use D in place of D_n). An ensemble D is polynomial time constructible if there exists a probabilistic polynomial time Turing Machine (PPTM) $\mathcal{M}$ such that $D_n = \mathcal{M}(1^n)$.

Definition 2.1. The *statistical distance* between two distributions X and Y over $\{0,1\}^\ell$, which we denote by $\Delta(X, Y)$, is defined as:

$$\Delta(X, Y) = \frac{1}{2} \sum_{\alpha \in \{0,1\}^\ell} |Pr[X = \alpha] - Pr[Y = \alpha]|$$

Definition 2.2 (Computational Indistinguishability [15, 24])**.** Two probability ensembles D and F are computationally indistinguishable if there exists a negligible function neg, such that for any PPTM $\mathcal{M}$, that receives 1^n and one sample s from D_n or F_n, and outputs 0 or 1, for large enough n's:

$$|Pr_{s \sim D_n}[\mathcal{M}(1^n, s) = 1] - Pr_{s \sim F_n}[\mathcal{M}(1^n, s) = 1]| \leq neg(n)$$

2.2 Previous Definitions of Obfuscation

In the subsequent definitions, we consider a family $\mathcal{C}$ of probabilistic polynomial size circuits to be obfuscated. For a length parameter n let $\mathcal{C}_n$ be the circuits in $\mathcal{C}$ with input length n. The size of the circuits in $\mathcal{C}_n$ is polynomial in n. If the obfuscator $\mathcal{O}$ is a polynomial-size circuit, then we say it *efficiently obfuscates* the family $\mathcal{C}$, and that $\mathcal{C}$ is *efficiently obfuscatable*. Whenever we refer to obfuscation, we will mean (efficient) black-box obfuscation unless explicitly noted otherwise. For definitions of obfuscation in the random oracle model, see Section 5.

Definition 2.3 (Black-Box Obfuscation [2])**.** An algorithm $\mathcal{O}$ that takes as input a circuit in $\mathcal{C}$ and outputs a new circuit, is said to be a *black-box obfuscator for the family* $\mathcal{C}$, if it has the following properties:

- Preserving Functionality:

  For any input length n, for any $C \in \mathcal{C}_n$:

  $$Pr[\exists x \in \{0,1\}^n : \mathcal{O}(C)(x) \neq C(x)] \leq neg(n)$$

  where the probability is over $\mathcal{O}$'s coins.

- Polynomial Slowdown:

  There exists a polynomial $p(n)$ such that for all input lengths, for any $C \in \mathcal{C}_n$, the obfuscator $\mathcal{O}$ only enlarges C by a factor of p: i.e., $|\mathcal{O}(C)| \leq p(|C|)$.

- Virtual Black-box:

  For any polynomial size circuit adversary $\mathcal{A}$, there exists a polynomial size simulator circuit $\mathcal{S}$ such that for every input length n and every $C \in \mathcal{C}_n$:

  $$|Pr[\mathcal{A}(\mathcal{O}(C)) = 1] - Pr[|\mathcal{S}^C(1^n) = 1]| \leq neg(n)$$

  where the probability is over the coins of the adversary, the simulator and the obfuscator.

If in addition the obfuscator $\mathcal{O}$ is also efficient (runs in time polynomial in the size of the circuit being obfuscated), then we say $\mathcal{O}$ is an *efficient* black-box obfuscator for the family $\mathcal{C}$.

Definition 2.4 (Indistinguishability Obfuscation [2])**.** An algorithm $\mathcal{O}$, that takes as input a circuit in $\mathcal{C}$ and outputs a new circuit, is said to be a *(computational/statistical/perfect) indistinguishability obfuscator for the family* $\mathcal{C}$, if it has the preserving functionality and polynomial slowdown properties as in Definition 2.3, and also has the following property (*instead of* the virtual black-box property).

- Computationally/Statistically/Perfectly Indistinguishable Obfuscation:

  For all large enough input lengths, for any circuit $C_1 \in C_n$ and for any circuit $C_2 \in \mathcal{C}_n$ *that computes the same function as* C_1 and such that $|C_1| = |C_2|$, the two distributions $\mathcal{O}(\mathbf{C_1})$ and $\mathcal{O}(\mathbf{C_2})$ are (respectively) computationally/statistically/perfectly indistinguishable.

If in addition the obfuscator $\mathcal{O}$ is also efficient (runs in time polynomial in the size of the circuit being obfuscated), then we say $\mathcal{O}$ is an *efficient* indistinguishability obfuscator for the family $\mathcal{C}$.

2.3 New Definition of Obfuscation

In this section we introduce a new definition for an obfuscator that does as well as is possible. This definition (similarly to Definition 2.4) avoids the black-box paradigm.

Informally, best-possible obfuscation, as presented below in Definition 2.5, guarantees that anything that can be learned efficiently from the obfuscated $\mathcal{O}(C_1)$, can also be extracted efficiently (*simulated*) from *any program* C_2 *of similar size for the same function.* Thus, any information that is exposed by $\mathcal{O}(C_1)$ is exposed by *every other* equivalent circuit of a similar size, and we conclude that $\mathcal{O}(C_1)$ is no worse an obfuscation than any of these other circuits (it is, in other words, the best obfuscation possible).

Definition 2.5 (Best-Possible Obfuscation)**.** An algorithm $\mathcal{O}$ that takes as input a circuit in $\mathcal{C}$ and outputs a new circuit, is said to be a *(computationally/statistically/perfectly) best-possible obfuscator for the family* $\mathcal{C}$, if it has the preserving functionality and polynomial slowdown properties as in Definition 2.3, and also has the following property (*instead of* the virtual black-box property).

- Computational/Statistical/Perfect Best-Possible Obfuscation:

  For any polynomial-size learner $\mathcal{L}$, there exists a polynomial size simulator $\mathcal{S}$ such that for every large enough input length n, for any circuit $C_1 \in C_n$ and for any circuit $C_2 \in \mathcal{C}_n$ *that computes the same function as* C_1 and satisfies $|C_2| = |C_1|$, the two distributions $\mathcal{L}(\mathcal{O}(\mathbf{C_1}))$ and $\mathcal{S}(\mathbf{C_2})$ are (respectively) computationally/statistically/perfectly indistinguishable.

If in addition the obfuscator $\mathcal{O}$ is also efficient (runs in time polynomial in the size of the circuit being obfuscated), then we say $\mathcal{O}$ is an *efficient* best-possible obfuscator for the family $\mathcal{C}$.

An Alternative Formulation. When dealing with best-possible obfuscators, we often refer to the *"empty" learner*; this is the learner that simply outputs whatever obfuscation it gets as input. If there exists an efficient simulator $\mathcal{M}$ for the "empty" learner, then there exists an efficient simulator $\mathcal{M}'$ for every efficient learner $\mathcal{L}$: $\mathcal{M}'$ on input C_2 simply computes $\mathcal{M}(C_2)$ and outputs the result of $\mathcal{L}(\mathcal{M}(C_2))$. We know that for circuits C_1, C_2 of identical size and identical functionality the distributions $\mathcal{O}(C_1)$ and $\mathcal{M}(C_2)$ are indistinguishable, and thus also $\mathcal{L}(\mathcal{O}(C_1))$ and $\mathcal{M}'(C_2) = \mathcal{L}(\mathcal{M}(C_2))$ must be indistinguishable.

Thus, an equivalent definition to "Best Possible" could do away with the learner and only require the existence of an *efficient* simulator, i.e., a simulator $\mathcal{S}$ such that for circuits C_1, C_2 of identical size and identical functionality the distributions $\mathcal{O}(C_1)$ and $\mathcal{S}(C_2)$ are indistinguishable.[2] We feel, though, that this looses some of Definition 2.5's intuitive appeal.

Efficient and Inefficient Obfuscation. Note that while it seems that (for any reasonable notion of obfuscation) it would be important for obfuscators to be efficient, we find even the question of inefficient obfuscation to be interesting, both from a complexity-theoretic and possibly also from a practical point of view. In some applications, it may be sufficient to obfuscate a small *kernel* of the program, and the program owner may be willing to invest a huge amount of time in this obfuscation procedure (especially if the obfuscation guarantee is information-theoretic). The main focus of our work, though, is efficient obfuscation. When we refer to black-box, best-possible or indistinguishability obfuscators we mean *efficient* obfuscators unless we explicitly note otherwise.

More on Statistical Guarantees. The notion of computationally best-possible obfuscation gives the guarantee that an whatever an efficient learner can learn from the obfuscation is indistinguishable *to efficient distinguishers* from what can be efficiently learned from any other circuit with the same functionality and of a similar size. This certainly seems sufficient for the software-protection applications of obfuscation, but of course it would be even better if *no distinguisher* (even inefficient) could distinguish the (efficient) learner's output from what can be efficiently learned from any other circuit with the same functionality and of a similar size. This is the guarantee given by perfectly best-possible obfuscation (by perfect indistinguishability, we mean that the

[2] In fact, this shows that efficient indistinguishability obfuscation implies efficient best-possible obfuscation, by taking the indistinguishability obfuscator $\mathcal{O}$ to also be the simulator. See Proposition 3.5 for a full proof.

distributions are identical — statistical distance 0). Statistically best-possible obfuscation bounds the information-theoretic distinguisher's advantage in distinguishing the learner's output from what can be learned from another functionally equivalent circuit. For statistical indistinguishability, unless noted otherwise, we only assume that the distinguisher's advantage (the statistical distance) is smaller than a (specific) constant. This strengthens negative results.

We observe that the existence of an *inefficient* perfectly best-possible obfuscator, implies the existence of an efficient one that uses the simulator to obfuscate. A similar argument also applies to statistically best-possible obfuscation, unless the statistical distance guarantee is very weak.

Note that the "ultimate" information-theoretic guarantee would be that everything that can be learned even by an *inefficient* learner from an obfuscated circuit, can also be simulated perfectly and efficiently by an *efficient* simulator from any functionally equivalent circuit of the same size. In fact, our positive result for best-possible obfuscation (see Proposition 3.2) gives such a guarantee. Our negative results for information-theoretically secure obfuscation are strengthened by considering only efficient learners.

Throughout this work, when we refer to best-possible or indistinguishability obfuscators we always mean *computational* (and efficient, see above) obfuscators unless we explicitly note otherwise.

3 Comparison with Prior Definitions

In this section we compare the new definition of best-possible obfuscation to the black-box and indistinguishability definitions proposed by Barak *et al.* [2].

3.1 Best-Possible vs. Black-Box Obfuscation

Best-possible obfuscation is a relaxed requirement that departs from the black-box paradigm of previous work. We first observe that any black-box obfuscator is also a best-possible obfuscator.

Proposition 3.1. *If $\mathcal{O}$ is an efficient black-box obfuscator for circuit family $\mathcal{C}$, then $\mathcal{O}$ is also an efficient (computationally) best-possible obfuscator for $\mathcal{C}$.*

Proof. Assume for a contradiction that $\mathcal{O}$ is not a best-possible obfuscator for C. This implies that there is no best-possible simulator for the "empty" learner that just outputs the obfuscated circuit it gets. In particular, $\mathcal{O}$ itself is not a good simulator. Thus there exists a polynomial p and a distinguisher $\mathcal{D}$, such that for infinitely many input lengths n, there exist two circuits $C_1, C_2 \in \mathcal{C}_n$, such that $|C_1| = |C_2|$ and C_1 and C_2 are equivalent, but:

$$|Pr[\mathcal{D}(\mathcal{O}(C_1)) = 1] - Pr[\mathcal{D}(\mathcal{O}(C_2)) = 1]| \geq p(n)$$

Now consider $\mathcal{D}$ as a predicate adversary for the black-box obfuscator $\mathcal{O}$. The black-box simulator $\mathcal{S}$ for $\mathcal{D}$ clearly behaves identically on C_1 and C_2 (because they have the same functionality), but $\mathcal{D}$'s behavior on $\mathcal{O}(C_1)$ and $\mathcal{O}(C_2)$ is non-negligibly different. Thus (for infinitely many input lengths) $\mathcal{S}$ is not a black-box simulator for $\mathcal{D}$, a contradiction.

□

Next, we provide a (weak) separation result. We exhibit a natural (low) complexity class, that of languages computable by polynomial size ordered binary decision diagrams (POBDDs), such that best-possible obfuscation *within* the class is achievable, but there are simple functionalities that are provably impossible to black-box obfuscate *within the class*.

Ordered Binary Decision Diagrams (OBDDs). The computational model of ordered binary decision diagrams was introduced by Bryant [5]. An ordered binary decision diagram is a rooted directed acyclic graph with a vertex set V containing non-terminal vertices, each with two children, and terminal vertices (without children), each labeled 0 or 1. Each edge e in the graph is marked with an input literal ℓ_e (e.g. ℓ_e could be x_1, $\overline{x_8}$ etc.). For every non-terminal vertex, the labels of its (two) outgoing edges should be negations of each other (e.g. x_3 and $\overline{x_3}$). An input $x \in \{0,1\}^n$ is accepted by an OBDD if and only if after removing every edge e for which $\ell_e = 0$ there exists a path from the root node to a terminal node labeled by 1. In addition, in an OBDD, on *every* path from the root vertex to a terminal vertex, the indices of the literals of edges on the path must be strictly increasing. We will focus on polynomial-size OBDDs, or POBDDs. We note that another way to view POBDDs is as logarithmic-space deterministic Turing Machines whose input tape head can only move in one direction (from the input's first bit to its last).

Bryant [5] showed that OBDDs have a simple canonical representation. For any function, there exists a *unique* smallest OBDD that is its canonical representation. Moreover, for polynomial-size OBDDs, this canonical representation is efficiently computable.

Note that we defined obfuscation for *circuits*, not OBDDs, but for every OBDD, there exists a boolean circuit (that computes the same functionality) from which it is easy to extract the OBDD. When we refer to obfuscating the family of OBDDs, we are implicitly referring to obfuscating the underlying family of circuits representing OBDDs.

We begin by observing that POBDDs can be perfectly best-possible obfuscated *as POBDDs* (namely the output of the obfuscator is a POBDD itself). This is a corollary of POBDDs having efficiently computable *canonical representations*.

Proposition 3.2. *There exists an efficient perfectly best-possible (and perfectly indistinguishable) obfuscator for POBBDs.*

Proof. The best-possible obfuscator for a POBDD P simply takes P, computes (efficiently) its canonical representation, and outputs that program as the best-possible obfuscation. The canonical representation has the same functionality as P, is no larger than P, and (most significantly) is unique, depending only on the functionality of P. The simulator gets a POBDD P' and also efficiently computes its canonical representation. The canonical representations of P and P' are identical if and only if P and P' compute the same functionality. Thus the obfuscator is indeed a perfectly best-possible obfuscator for the family of POBDDs.

□

We next show that there exists a family of languages computable by POBDDs, that cannot be black-box obfuscated (efficiently or inefficiently) as POBDDs (i.e the resulting program itself being represented as a POBDD). This gives a (weak) separation between best-possible and black-box obfuscation. The weakness is that it remains possible that any input POBDD *can* be black-box obfuscated such that the output circuit is no longer a POBDD but is in some higher complexity class.

Proposition 3.3. *There exists a family of languages computable by POBDDs, that cannot be black-box obfuscated as POBDDs.*

Sketch. Intuitively, because POBDDs have a simple efficiently computable canonical representation, non black-box information can be extracted from a POBDD by reducing it to its "nice" canonical form, and then extracting information from this canonical form.

More formally, consider (for example) the simple family of point functions $\{I_p\}_{p\in\{0,1\}^n}$, where the function I_p outputs 1 on input the point p and 0 everywhere else. Note that point functions are computable by POBDDs. Now observe that any POBDD computing a point function for a point p can be reduced to its canonical representation, from which p is easily extracted. Thus for any supposed obfuscator that obfuscates point functions as POBDDs there exists an adversary that (for *every point*) can extract all the bits of the point from the "obfuscated" POBDD. Clearly, no black-box simulator can successfully extract even a single bit of the point for a non-negligible fraction of point functions. Thus there exists no black-box obfuscator that obfuscates POBDDS computing point functions *as POBDDs*.

□

We note that many other natural languages computable by POBDDs cannot be black-box obfuscated as POBDDs. Black-box obfuscation of POBDDs as more complex circuits remains an intriguing open question.

3.2 Best-Possible vs. Indistinguishability Obfuscation

As mentioned above, the notions of best-possible obfuscation and indistinguishability obfuscation are related, though the guarantees given by these two types of obfuscation are different. In this section we will show that any best-possible obfuscator is also an indistinguishability obfuscator. Furthermore, for *efficient* obfuscation, the two notions are equivalent. For *inefficient* obfuscation (which we still find interesting), however, the notions are *not* equivalent unless the polynomial hierarchy collapses. In fact, inefficient indistinguishability obfuscators exist unconditionally (see [2]). On the other hand, building even inefficient best-possible obfuscators remains an interesting open question. We begin by showing that best-possible obfuscation is in fact at least as strong as indistinguishability obfuscation.

Proposition 3.4. *If $\mathcal{O}$ is a perfectly/statistically/computationally best-possible obfuscator for circuit family $\mathcal{C}$, then $\mathcal{O}$ is also a (respectively) perfect/statistical/computational indistinguishability obfuscator for $\mathcal{C}$.*

Sketch. To prove the claim, consider the "empty" learner $\mathcal{L}$ that just outputs whatever obfuscation it is given, and its simulator $\mathcal{S}$. Let δ be the distinguishability measure (computational or statistical) in the (computational or perfect/statistical) guarantee of the obfuscator. We get that for any two circuits C_1 and C_2 that are of the same size and compute the same functionality: $\delta(\mathcal{L}(\mathcal{O}(C_1)), \mathcal{S}(C_2)) = \delta(\mathcal{O}(C_1), \mathcal{S}(C_2)) \leq \varepsilon$, and $\delta(\mathcal{L}(\mathcal{O}(C_2), \mathcal{S}(C_2))) = \delta(\mathcal{O}(C_2), \mathcal{S}(C_2)) \leq \varepsilon$, and thus (since computational and statistical distinguishabilities are transitive):

$$\delta(\mathcal{O}(C_1), \mathcal{O}(C_2)) \leq 2\varepsilon$$

Note that the perfect/statistical/computational guarantee is preserved.

□

As noted above, if we restrict our attention to efficient obfuscators, indistinguishability obfuscators are also best-possible obfuscators.

Proposition 3.5. *If $\mathcal{O}$ is an* efficient *perfect/statistical/computational indistinguishability obfuscator for a circuit family $\mathcal{C}$, then $\mathcal{O}$ is also an efficient (respectively) perfectly/statistically/ computationally best-possible obfuscator for $\mathcal{C}$.*

Proof. Let $\mathcal{O}$ be an efficient indistinguishability obfuscator. Then for any learner $\mathcal{L}$, let $\mathcal{S}$ be the (efficient) simulator that gets a circuit C_2, runs $\mathcal{O}(C_2)$, and then activates $\mathcal{L}$ on $\mathcal{O}(C_2)$. We get that if $\mathcal{O}$ is a perfect/statistical/computational indistinguishability obfuscator, then for any two circuits C_1 and C_2 that are of the same size and compute the same functions, the two distributions $\mathcal{L}(\mathcal{O}(C_1))$ and $\mathcal{S}(C_2) = \mathcal{L}(\mathcal{O}(C_2))$ are perfectly/statistically/computationally indistinguishable (because $\mathcal{O}$ is an indistinguishability obfuscator). Thus $\mathcal{O}$ is also an efficient best-possible obfuscator.

Note that the efficiency of the indistinguishability obfuscator is *essential* to guarantee the efficiency of the simulator, without which the obfuscator does not meet the best-possible definition.

□

It is important to note that there is no reason to believe that the two notions of obfuscation are equivalent for *inefficient* obfuscation. In fact, whereas [2] design exponential-time indistinguishability obfuscators, there is no known construction for inefficient *best-possible* obfuscators. We find even the construction of *inefficient* best-possible obfuscators to be an interesting problem.

We end this subsection by observing that if $P = NP$ then it is not hard to construct efficient perfect best-possible obfuscators (and indistinguishability obfuscators) for every polynomial-size circuit. In fact this complexity assumption is almost tight. We will show in Theorem 4.1, that if statistically best-possible obfuscators can be built even for very simple circuits, then the polynomial hierarchy collapses to its second level.

Proposition 3.6. *If P=NP then the family of polynomial-sized circuits can be efficiently perfectly best-possible obfuscated.*

Proof. Assume $P = NP$. For any circuit C, it is possible to *efficiently* extract the smallest lexicographically first circuit C_{min} that is equivalent to C (this problem is solvable using a language in the second level of the polynomial hierarchy). As Barak *et al.* [2] note, such an extraction procedure is a perfectly indistinguishable obfuscation of C, and thus there exists an efficient perfect indistinguishability obfuscator for the family of polynomial-size circuits. By Proposition 3.5 it is also an efficient perfectly best-possible obfuscator for the family of polynomial-size circuits. Note that even if $P \neq NP$ then we get an *inefficient* indistinguishability obfuscator. It remains, however, unclear whether we can get an inefficient best-possible obfuscator, as the (always efficient!) simulator can no longer run the "circuit minimization" procedure.

□

4 Impossibility Results for Statistically best-Possible Obfuscation

In this section we present a hardness result for statistically best-possible obfuscation. In Section 3 it was shown that if $P = NP$ then every polynomial-sized circuit can be perfectly best-possible obfuscated, thus we cannot hope for an *unconditional* impossibility result. We show that the condition $P = NP$ is (nearly) tight, and in fact the existence of statistically best-possible obfuscators even for the class of languages recognizable by 3-CNF circuits (a sub-class of AC^0) implies that the polynomial hierarchy collapses to its second level. This result shows the impossibility of statistically best-possible obfuscation for any class that contains 3-CNF formulas (and in particular also for the class of general polynomial sized circuits).

Theorem 4.1. *If the family of 3-CNF formulas can be statistically best-possible obfuscated (not necessarily efficiently), then the polynomial hierarchy collapses to its second level.*

Proof. **A Simple Case.** We begin by considering a simple case: suppose that the family of 3-CNF formulas can be *perfectly* best-possible obfuscated (not necessarily efficiently) while *perfectly* preserving functionality (i.e. the obfuscated circuit never errs). We can use the Simulator $\mathcal{S}$ for the "empty" learner, to construct an NP proof for Co-SAT (a Co-NP -complete problem). To see this, consider an input 3-CNF formula φ of size $|\varphi|$. We would like to find a witness for non-satisfiability of φ. Towards this end, we first construct an unsatisfiable formula ψ of size $|\varphi|$. A witness for the non-satisfiability of φ is a pair of random strings (r, r') such that the output of the simulator $\mathcal{S}$ on φ with randomness r is equal to its output on ψ with randomness r'. This proof system is indeed in NP:

- Efficiently Verifiable. The simulator is efficient, and thus the witness is efficiently verifiable.
- Complete. If φ is unsatisfiable, then φ and ψ compute the same function (the constant 0 function) and are of the same size. We know that $\mathcal{O}$ is a perfect best-possible obfuscator and thus the distributions $\mathcal{O}(\varphi)$, $\mathcal{S}(\varphi)$, $\mathcal{S}(\psi)$, $\mathcal{O}(\psi)$ are all identical. This implies that there must exist (r, r') such that $\mathcal{S}(\varphi, r) = \mathcal{S}(\psi, r')$.
- Sound. If φ is satisfiable, then because the obfuscator perfectly preserves functionality, the distributions $\mathcal{O}(\varphi)$, $\mathcal{O}(\psi)$ are disjoint (they are distributions of circuits with different functionalities). Thus the distributions $\mathcal{S}(\varphi)$, $\mathcal{S}(\psi)$ of the (perfect) simulator's output are also disjoint, and there exist no (r, r') such that $\mathcal{S}(\varphi, r) = \mathcal{S}(\psi, r')$.

Full Proof. The full proof for the case of statistically best-possible obfuscation follows along similar lines, giving a reduction from a Co-NP -complete problem (circuit equivalence) to a problem in AM.[3] By the results of Fortnow [12], Aliello and Håstad [1], and Boppana, Håstad and Zachos [4] (see also Feigenbaum and Fortnow [10]), this collapses the polynomial hierarchy to its second level. First, assume that the statistically best-possible obfuscator guarantees statistical distance at most $\frac{1}{10}$ between the simulator and obfuscator. We begin with some complexity theory background:

Background. The statistical difference (SD) problem was introduced by Sahai and Vadhan [22]. It is a promise problem that receives a pair (C_1, C_2) of polynomial-size circuits, that can each be used to sample a distribution. The yes-instances of the problem are pairs of circuits for which the statistical distance between the distributions generated by the two circuits is at least $\frac{2}{3}$, no-instances are pairs of circuits for which this distance is at most $\frac{1}{3}$. Sahai and Vadhan [22] prove that SD is complete for the complexity class SZK of languages with statistical zero-knowledge protocols. The complement of SD, the problem $\overline{SD}$ is thus also in SZK (and is in fact also complete), by the result of Okamoto [21] that SZK is closed under complement.

The results of Fortnow [12], Aliello and Håstad [1], and Boppana, Håstad and Zachos [4] show that if there is a Co-NP -complete problem in SZK, then the polynomial hierarchy collapses to its second level.

The problem of checking the equivalence of 3-CNF formulas consists of checking, given two 3-CNF formulas over n boolean input variables (say that the formulas must be of the same size), whether they agree on all possible assignments to the n variables. Checking the equivalence of 3-CNF formulas is Co-NP complete (there is a simple reduction to it from the Co-NP complete problem $\overline{3SAT}$).

[3] Actually, this is a problem in statistical zero knowledge: the complement of the Statistical Difference Problem, introduced by Sahai and Vadhan [22].

Outline. We assume that the family of 3-CNF formulas can be statistically best-possible obfuscated (even inefficiently). We will use the simulator to construct an efficient (Karp) reduction from the $CoNP$-compete problem of 3-CNF equivalence testing, to the $\overline{SD}$ problem, which has a statistical zero-knowledge protocol.

In particular, this reduction will imply that if there exists a statistically best-possible obfuscator for the family of 3-CNFs, then there exists a statistical zero-knowledge protocol checking the equivalence of 3-CNFs. This collapses the polynomial hierarchy.

The Reduction. Let $\mathcal{S}$ be the (efficient) simulator for the "empty" learner that just outputs the obfuscated circuit that it is given. The reduction receives two identical size 3-CNF formulas, ϕ_1 and ϕ_2. It proceeds to generate two circuits C_1 and C_2. The circuit C_i is simply the simulator S with the formula ϕ_i hardwired as its input. The two circuits (C_1, C_2) generate the *distributions* $\mathcal{S}(\phi_1), \mathcal{S}(\phi_2)$ respectively, and they will be the input to $\overline{SD}$.

- Completeness: If $\phi_1 \equiv \phi_2$, then by the properties of the best-possible obfuscator $\Delta(\mathcal{O}(\phi_1), \mathcal{S}(\phi_1)) \leq \frac{1}{10}$, and $\Delta(\mathcal{O}(\phi_1), \mathcal{S}(\phi_2)) \leq \frac{1}{10}$. Thus $\Delta(\mathcal{S}(\phi_1), \mathcal{S}(\phi_2)) < \frac{1}{3}$, and $(C_1, C_2) \in \overline{SD}$
- Soundness: If $\phi_1 \not\equiv \phi_2$, then by the preserving functionality property of the obfuscator, the two distributions $\mathcal{O}(\phi_1)$ and $\mathcal{O}(\phi_2)$ are very far (at statistical distance almost 1). On the other hand, $\Delta(\mathcal{O}(\phi_1), \mathcal{S}(\phi_1)) \leq \frac{1}{10}$, and $\Delta(\mathcal{O}(\phi_2), \mathcal{S}(\phi_2)) \leq \frac{1}{10}$, and thus we conclude that $\Delta(\mathcal{S}(\phi_1), \mathcal{S}(\phi_2)) > \frac{2}{3}$, and $(C_1, C_2) \notin \overline{SD}$.

Note that the reduction is efficient even if the obfuscator is not, as it only uses the code of the simulator (which is always efficient).

□

Proposition 3.2 and Theorem 4.1 give examples of circuit classes that can and cannot be statistically best-possible obfuscated. The proofs give characterizations of circuit classes that can be statistically best-possible obfuscated. A *sufficient* condition for statistically best-possible obfuscation of a class of circuits is having an efficiently computable canonical representation, a *necessary* condition is having a statistical zero knowledge proof for the equivalence problem.

Finally, a corollary of this theorem is that the same class of 3-CNF formulas cannot be statistically *indistinguishability* obfuscated in polynomial time unless the polynomial hierarchy collapses. This is the first impossibility result for indistinguishability obfuscation in the standard model.

Corollary 4.2. *If the family of 3-CNF formulas can be* efficiently *statistically indistinguishability obfuscated, then the polynomial hierarchy collapses to its second level.*

Proof. By Proposition 3.5, if there exists an *efficient* statistical indistinguishability obfuscator for the family of 3-CNFs, then there also exists an efficient statistically best-possible obfuscator for the same family. This, in turn, implies (by Theorem 4.1) that the polynomial hierarchy collapses to its second level.

□

5 Best-Possible Obfuscation in the Random Oracle Model

In this section we present an impossibility result for (efficient) computationally best-possible obfuscation in the (programmable) random oracle model. We begin by describing the model and recasting the definitions of obfuscation in the presence of random oracles.

5.1 Definitions, Revisited

The Random Oracle Model. In the random oracle model we assume that all parties (the circuits, obfuscator, adversary etc.) have access to a random oracle and can make oracle queries. All oracle queries are answered by a single function $\mathcal{R}$, that is selected uniformly and at random from the set of all functions. Specifically, for each input length n, $\mathcal{R}$ will be a function from $\{0,1\}^n$ to $\{0,1\}^{p(n)}$ for some polynomial p. For simplicity, we will assume throughout this work that for all n's the function $\mathcal{R}$ is a random *permutation* [4] on $\{0,1\}^n$. Circuits access the random oracle by making oracles queries using a special *oracle gate*. It is important that we assume that calls to these oracle gates are clearly visible when running the circuit.

Obfuscation in the Random Oracle Model. When considering obfuscation in the random oracle model, all circuits are allowed oracle access (*including the circuits to be obfuscated*), and all probabilities are taken over the selection of a random oracle.

In all definitions of obfuscation (Definitions 2.3, 2.4, 2.5), in the the *preserving functionality* requirement, the probability that there exists an input for which the obfuscated circuit gives the wrong answer is taken also over the selection of the random oracle.

In the *Virtual Black-box* requirement of Definition 2.3, we require that for any polynomial size circuit adversary $\mathcal{A}$, there exists a polynomial size simulator circuit $\mathcal{S}$ such that for every input length n and every $C \in \mathcal{C}_n$:

$$|Pr[\mathcal{A}(\mathcal{O}(C)) = 1] - Pr[|\mathcal{S}^C(1^n) = 1]| \leq neg(n)$$

where the probability is over the random oracle, the coins of the adversary, the simulator and the obfuscator.

In the *Indistinguishability Obfuscation* requirement of Definition 2.4 and the *Best-Possible Obfuscation* requirement of Definition 2.5, all distributions are taken over the random oracle (this also affects the circuits being obfuscated, which may include values that depend on the random oracle).

5.2 The Impossibility Result

We show how to use a random oracle to build circuits that cannot be best-possible obfuscated for point functions. We note that the use of the random oracle both strengthens and weakens this result. The result is strengthened because a random oracle could conceivably help obfuscation (*a la* [18]), but weakened because the random oracle is used to build a circuit that cannot be obfuscated. Moreover, in the proof we need to assume that a distinguisher can see the obfuscated circuit's oracle calls and that it can access the random oracle itself. It is still possible that all circuits that do not use the random oracle can be best-possible obfuscated.

[4] Note that results hold for random function oracles (as long as the function's range is significantly larger than its domain, say at least twice as large).

We show that a specific family of circuits for computing point functions cannot be obfuscated in the presence of a random oracle $\mathcal{R}$. A point function I_p is the function that outputs 1 on input p and 0 on all other inputs. We begin by presenting the family of point function circuits for which we will show impossibility of obfuscation.

Definition 5.1 (The circuit family $\{C_p^r\}$)**.** For any input length n, the family of circuits $\{C_p^r\}_n$ defines a set of circuits on inputs of length n. Each point $p \in \{0,1\}^n$ and *pad* $r \in \{0,1\}^n$ define a circuit C_p^r that computes the point function I_p. The data contained in the circuit C_p^r is:

- The pad r is included in C_p^r "in the clear".
- The point p is "hidden", the only information that is given about it is $y = \mathcal{R}(p \circ r)$.

For an input $x \in \{0,1\}^n$, to compute the point function I_p, the circuit C_p^r outputs 1 if and only if $\mathcal{R}(x \circ r) = y$ (recall $y = \mathcal{R}(p \circ r)$). Otherwise the circuit outputs 0.

We claim that the family of point function circuits $\{C_p^r\}$ cannot be best-possible obfuscated.

Theorem 5.1. *The circuit family $\{C_p^r\}$ cannot be efficiently computationally best-possible obfuscated.*

Proof Intuition. Observe that any obfuscator $\mathcal{O}$ must preserve the functionality of a circuit C_p^r. Furthermore, the only information the obfuscator has about the point p is the value $\mathcal{R}(p \circ r)$. To preserve functionality, for any input x, the obfuscated circuit $\mathcal{O}(C_p^r)$ needs to find out whether $x = p$. Now, because the only information available to the obfuscator and the obfuscated circuit about p is the value $\mathcal{R}(p \circ r)$, for most inputs x, the obfuscated circuit must ask the random oracle for the value $\mathcal{R}(x \circ r)$. Thus, for many x's, one of the (polynomially many) oracle calls of $\mathcal{O}(C_p^r)(x)$ should be to $\mathcal{R}(x \circ r)$.

In the proof we construct a distinguisher between obfuscated circuits and the output of the "empty" learner's simulator. For a random pad $s \in \{0,1\}^n$, we examine the distinguisher $\mathcal{D}_s$ that is given an obfuscation $\mathcal{O}(C_p^r)$ of a circuit C_p^r computing a (randomly chosen) point function I_p. The distinguisher $\mathcal{D}_s$ runs the circuit $\mathcal{O}(C_p^r)$ on a random input x, and tries to "guess" whether the circuit C_p^r has s for its pad, i.e. whether $r = s$. To do this, the distinguisher runs $\mathcal{O}(C_p^r)$ on a random input $x \in \{0,1\}^n$ and outputs 1 if and only if $\mathcal{O}(C_p^r)$ queried the random oracle on $x \circ s$.

Let $r \in \{0,1\}^n$ be a second random pad. Recall that we concluded above that for a random input x we expect $\mathcal{O}(C_p^s)(x)$ to query the random oracle on $x \circ s$. On the other hand, for a random pad r, the probability (over the independently random s) that $\mathcal{O}(C_p^r)$ queries the random oracle on $x \circ s$ is very small. Thus, the distinguisher $\mathcal{D}_s$ behaves very differently on $\mathcal{O}(C_p^s)$ than on $\mathcal{O}(C_p^r)$, even though both these circuits have exactly the same functionality! This means that $\mathcal{O}$ is not a best-possible obfuscator.

Proof. Throughout this proof we restrict our attention to the "empty" learner that just outputs the obfuscation it gets, and the simulator $\mathcal{S}$ for this "empty" learner. We begin by describing the distinguisher (family) we will use.

The Distinguisher $\mathcal{D}_s$. Every pad $s \in \{0,1\}^n$ defines a distinguisher $\mathcal{D}_s$. This distinguisher gets as input a (probabilistic) oracle circuit C with n-bit inputs (we will consider input circuits that are obfuscations of circuits in the family $\{C_p^r\}$). The distinguisher chooses a random input $x \in \{0,1\}^n$ and random coins r for the circuit C, and then runs C on input x with randomness r. If in this execution C makes the oracle query $x \circ s$ then D_s outputs 1, otherwise D_s outputs 0.

Claim 5.1. *Let $\mathcal{O}$ be any efficient obfuscator.*

$$Pr\left[\mathcal{D}_s(\mathcal{O}(C_p^s)) = 1\right] \geq 1 - neg()$$

where the probability is over random $s, p \in \{0,1\}^n$, the coins of $\mathcal{O}$ and $\mathcal{D}_s$, and the selection of the random oracle $\mathcal{R}$.

Proof. We show that for random $s, p, x \in \{0,1\}^n$, with high probability over the coins of $\mathcal{O}$ and $\mathcal{O}(C_p^s)$ and the random oracle $\mathcal{R}$, during its execution $\mathcal{O}(C_p^s)(x)$ queries the random oracle on $x \circ s$.

To see this, first assume that with probability a (over random $x, p, s \in \{0,1\}^n$, $\mathcal{O}$, $\mathcal{O}(C_p^s)$ and $\mathcal{R}$) neither $\mathcal{O}(C_p^s)$ nor $\mathcal{O}(C_p^s)(x)$ query the random oracle on $p \circ s$ or $x \circ s$. Now we claim that also with probability a, neither $\mathcal{O}(C_p^s)$ nor $\mathcal{O}(C_p^s)(p)$ queries the random oracle on $p \circ s$ or $x \circ s$. This is because as long as neither $\mathcal{O}(C_p^s)$ nor $\mathcal{O}(C_p^s)(z)$ queries either $p \circ s$ or $x \circ s$, their views (over the random variables and the selection of a random oracle) are identically distributed regardless of whether $z = x$ or $z = p$. We conclude that with probability a the behavior of $\mathcal{O}(C_p^s)(x)$ and $\mathcal{O}(C_p^s)(p)$ are identical, in particular their outputs are identically distributed, even though $C_p^s(x)$ is 0 (with all but negligible probability) and $C_p^s(p)$ is 1! Thus, by the preserving functionality requirement, a must be negligible.

For a random $s, p, x \in \{0,1\}^n$, the probability that the obfuscator $\mathcal{O}$ on input C_p^s queries $p \circ s$ or $x \circ s$ is negligible (over its coins and $\mathcal{R}$). The probability that $\mathcal{O}(C_p^s)(x)$ queries $p \circ s$ is also negligible (recall x is random). But we just saw that to maintain functionality either $\mathcal{O}(C_p^s)$ or $\mathcal{O}(C_p^s)(x)$ must make one of the oracle queries $p \circ s$, $x \circ s$! We conclude that with all but negligible probability $\mathcal{O}(C_p^s)(x)$ must query $x \circ s$.

In particular, this implies that for random $s, p \in \{0,1\}^n$, when $\mathcal{D}_s$ chooses a random $x \in \{0,1\}^n$ and runs $\mathcal{O}(C_p^s)(x)$ (with random coins), with all but negligible probability (over all the random variables, $\mathcal{O}$'s coins and the random oracle) $\mathcal{O}(C_p^s)(x)$ queries $x \circ s$ and $\mathcal{D}_s(\mathcal{O}(C_p^s))$ outputs 1.

□

Claim 5.2. *$\mathcal{O}$ be any efficient obfuscator.*

$$Pr\left[\mathcal{D}_s(\mathcal{O}(C_p^r)) = 1\right] \leq neg()$$

where the probability is over random $s, r, p \in \{0,1\}^n$, the coins of $\mathcal{O}$ and $\mathcal{D}_s$, and the selection of the random oracle $\mathcal{R}$.

Proof. For any input x, the oracle queries made by $\mathcal{O}(C_p^r)(x)$ are completely independent of s. Since $\mathcal{O}(C_p^r)(x)$ only makes a polynomial number of queries, the probability that one of those queries is to $x \circ s$ is negligible.

□

Putting together Claims 5.1 and 5.2, we conclude that for any efficient obfuscator, for a random $s, r, p \in \{0,1\}^n$, the distinguisher $\mathcal{D}_s$ distinguishes $\mathcal{O}(C^s_p)$ from $\mathcal{O}(C^r_p)$, a contradiction to the best-possible obfuscation requirement. We conclude that the circuit family $\{C^r_p\}$ cannot be best-possible obfuscated.

The pad s is a source of non-uniformity in the distinguisher. We have shown that a random s makes for a good distinguisher with high probability, and thus for every alleged obfuscator there certainly exists a good (non-uniform) distinguisher that breaks it.

□

The family of circuits that we show cannot be obfuscated is a family that computes point functions. This may seem contradictory, as Lynn, Prabhakaran and Sahai [18] showed that a class of circuits computing point functions *can* be obfuscated in the random oracle model. The source of this disparity is that they (as well as all other known positive results on obfuscating point functions) only consider obfuscators that get the point *in the clear*, whereas the family of point function circuits that we present ($\{C^r_p\}$) hides information about the point. Malkin [19], was the first to ask whether *any* point function implementation can be black-box obfuscated.

Theorem 5.1 shows impossibility for simpler and more natural functionalities than those considered in previous results, but does so using circuits with random oracle gates.

Extensions. We note that this impossibility result applies also to black-box obfuscation (because black-box obfuscation also implies best-possible obfuscation, see Proposition 3.1).

One possible objection to this impossibility result, is that the information revealed by obfuscation of circuits in the family $\{C^r_p\}$ (namely the pad r) is not related to the point p. We note, though, that it seems essential that an obfuscator strip programs of *all* non black-box information. Indeed, we view leaking information that is unrelated to the functionality as a serious problem, as one of the goals of obfuscation for software protection is "stripping" programs of non-essential information (e.g. embarrassing code comments, indications of software bugs etc.).

Implications for a world without random oracles. We conclude with an informal discussion of the ways in which our proof uses the random oracle model, and how one could hope to remove this assumption. Our construction uses the random oracle $\mathcal{R}$ in two ways. First, $\mathcal{R}$ is used to hide information about p in the circuit family $\{C^r_p\}$. Essentially, we use $\mathcal{R}$ to obfuscate a point function (where the point is $p \circ r$). Intuitively, since we know how to (black-box) obfuscate point functions without using random oracles, we could use (strong) cryptographic assumptions in place of the random oracle for this.

The second place in our proof where we use the properties of random oracles is when we assume a distinguisher can see the points on which the obfuscated circuit queries the random oracle. If we want to get rid of the random oracles, this is a more troubling assumption. The issue is that even if we could use some other method to hide information about the point p in the standard model, there is no reason to assume we could identify any internal computation of the obfuscated circuit. For example, consider using Canetti's point function obfuscation and giving the obfuscator a circuit C that hides some information on p by exposing only $f_z(p,r) = (z, z^{p \circ r})$ [5]. Even if on any input x the obfuscated circuit *always* computes the value $f_z(x,r) = (z, z^{x \circ r})$, there is no guarantee

[5] Here z is a randomly chosen member of Z^*_b (for b selected randomly by choosing a random prime a such that $a = 2b+1$), and $p \circ r$ corresponds to some member of the group.

that a distinguisher can identify these computations! Thus $\mathcal{O}(C)$ may not expose any information on r. We note, however, that to *prove* that an obfuscator can obfuscate *any* circuit computing a point function, one would have to construct an obfuscator that indeed hides internal computations. Thus it seems that even for achieving the (seemingly modest) goal of best-possible obfuscation for polynomial-size point-function circuits, one would have to present a method for hiding complex internal computations of a circuit. Such a method, in and of itself, seems to require significant progress on the problem of obfuscation.

6 Concluding Remarks and Discussions

We conclude with a discussion of best-possible obfuscation and issues raised in this work.

Input/Output Representation. Several of our results highlight the issue of the *representation* of an obfuscator's input and output. At times (in Section 3) we restrict the *representation* of both the obfuscator's input and output functionality to be "simple" circuits representing POBDDs. At other times (in the proof of Theorem 5.1), we construct complex circuits that hide information about their functionality from the obfuscators. In general, restricting the input representation makes the task of obfuscation easier (see discussion in Section 5), whereas restricting the output representation makes the task of obfuscation harder, and we use this in Proposition 3.3 to show that point functions cannot be black-box obfuscated as POBDDs. Previous positive results on obfuscation considered obfuscators that get a particular representation of the functionality (e.g. the point p for the point function I_p). Future work on black-box (and non black-box) obfuscation should consider the question of which *representations* of the desired functionality are obfuscated.

This issue was also raised by Malkin [19], who asked whether *any* point function implementation can be black-box obfuscated in the standard model. An relaxed (but related[6]) formulation of this question is whether the family of polynomial-size circuits computing point functions can be best-possible obfuscated. The proof of Theorem 5.1 answers this question negatively in the presence of random oracles, but either an impossibility proof or a provably secure obfuscator (in a world without random oracles) may have interesting consequences.

Circuit Sizes. In our definition of best-possible obfuscation (Definition 2.5) we compare the obfuscated circuit $\mathcal{O}(C_1)$ with circuits C_2 of the same size as C_1 (and computing the same functionality). This definition requires that the obfuscation of C_1 leak as little information as any equivalent circuit of a specific (polynomially) smaller size. We could make stronger requirements, such as leaking less information than an equivalent circuit C_2 that is as large as $\mathcal{O}(C_1)$, twice as large as C_1, etc. (all results would still hold). In general, the larger the circuit used as a benchmark (C_2), the stronger the definition. The important point is guaranteeing that $\mathcal{O}(C_1)$ leaks as little information as any other functionally equivalent circuit of a related size.

Auxiliary Input. Goldwasser and Kalai [14] augment the virtual black-box requirement of obfuscation to hold in the presence of auxiliary input. They note that this is an important requirement for any obfuscation that is used in practice, as auxiliary input comes into play in the real world.

[6]This formulation is equivalent to the original question raised by Malkin under the assumption that point functions can indeed be obfuscated when the point is given in the clear. In this case, a best-possible obfuscation leaks as little information as the black-box obfuscated point function circuits, and is thus also a black-box obfuscation.

Following this argument, we could extend the best-possible obfuscation requirement to hold in the presence of auxiliary input. This is a strengthening of the definition, and thus all negative results clearly still hold. The positive result of Proposition 3.2 (obfuscating POBDDs) also holds even in the presence of (dependent) auxiliary input.

Weaker Variants. In light of the negative results of Theorems 4.1 and 5.1 it is interesting to consider *weaker* variants of best-possible obfuscation (Definition 2.5). While the variants below lose some of the appealing intuitive "garbling" guarantee of Definition 2.5, meeting any of them would all give at least *some* indication that the obfuscator truly garbles circuits.

- *Hiding Less Information.* One natural approach is to follow in the footsteps of Barak *et al.* [2], and consider best-possible *predicate* obfuscators: an obfuscation is predicate best-possible if any *predicate* of the original circuit that can be learned from the obfuscation, could also be learned from any other circuit of a similar size computing the same functionality. While this definition is weaker than computationally best-possible obfuscation, the proof of Theorem 5.1 rules out even general-purpose predicate best-possible obfuscation in the random oracle model (and perhaps gives some intuition that this type of obfuscation would be hard to achieve in the standard model).

- *Weaker Indistinguishability.* Canetti [6] and Wee [23] relax the virtual black-box requirement, requiring only *polynomially* small indistinguishability between the output of an adversary and its simulator. Moreover, they allow the simulator's size to depend (polynomially) on this indistinguishability parameter. We note that negative results in this work (Theorems 4.1 and 5.1) hold even if we require only polynomially small indistinguishability and allow the simulator's size to depend (polynomially) on the indistinguishability parameter.

- *Weaker Functionality.* Definition 2.5 requires that with all but negligible probability, the obfuscated circuit *perfectly* preserves the functionality of the original circuit. We could relax this, and require only that for every input, with all but a small constant error probability, the obfuscated circuit outputs the same output as the original circuit. Our negative results apply even under this weakened preserving functionality requirement. The positive result on best-possible obfuscation of POBDDs (Proposition 3.2) gives an obfuscator that *perfectly* preserves the functionality of the circuit it obfuscates.

7 Acknowledgements

We thank Yael Tauman Kalai and Tali Kaufman for helpful and enjoyable discussions. Thanks also to anonymous reviewers for their insightful comments which much improved (or so we hope) the presentation. We particularly thank an anonymous reviewer for suggesting a simplification to construction of un-obfuscatable circuits in the random oracle model and the proof of Theorem 5.1.

References

[1] William Aiello, Johan Håstad. *Statistical Zero-Knowledge Languages can be Recognized in Two Rounds*. Journal of Computer and System Sciences 42(3): 327-345 (1991)

[2] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, Ke Yang. *On the (Im)possibility of Obfuscating Programs*. CRYPTO 2001: 1-18

[3] Mihir Bellare, Phillip Rogaway. *Random Oracles are Practical: A Paradigm for Designing Efficient Protocols*. ACM Conference on Computer and Communications Security 1993: 62-73

[4] Ravi B. Boppana, Johan Håstad, Stathis Zachos. *Does co-NP Have Short Interactive Proofs?* Information Processing Letters 25(2): 127-132 (1987)

[5] Randal E. Bryant. *Graph-Based Algorithms for Boolean Function Manipulation*. IEEE Transactions on Computers, C(35), No. 8, August, 1986: 677-691

[6] Ran Canetti. *Towards Realizing Random Oracles: Hash Functions That Hide All Partial Information*. CRYPTO 1997: 455-469

[7] Ran Canetti, Oded Goldreich, Shai Halevi. *The random oracle methodology, revisited*. Journal of the ACM 51(4): 557-594 (2004)

[8] Ran Canetti, Daniele Micciancio, Omer Reingold. *Perfectly One-Way Probabilistic Hash Functions (Preliminary Version)*. STOC 1998: 131-140

[9] Yevgeniy Dodis, Adam Smith. *Correcting errors without leaking partial information*. STOC 2005: 654-663

[10] Joan Feigenbaum, Lance Fortnow. *Random-Self-Reducibility of Complete Sets*. SIAM Journal on Computing 22(5): 994-1005 (1993)

[11] Amos Fiat, Adi Shamir. *How to Prove Yourself: Practical Solutions to Identification and Signature Problems*. CRYPTO 1986: 186-194

[12] Lance Fortnow. *The complexity of perfect zero-knowledge*. In S. Micali, editor, Advances in Computing Research, 5: 327-343. JAI Press, Greenwich, 1989

[13] Shafi Goldwasser, Yael Tauman Kalai. *On the (In)security of the Fiat-Shamir Paradigm*. FOCS 2003: 102-113

[14] Shafi Goldwasser, Yael Tauman Kalai. *On the Impossibility of Obfuscation with Auxiliary Input*. FOCS 2005: 553-562

[15] Shafi Goldwasser and Silvio Micali. *Probabilistic Encryption and How to Play Mental Poker Keeping Secret All Partial Information*. STOC 1982: 365-377

[16] Satoshi Hada. *Zero-Knowledge and Code Obfuscation*. Asiacrypt 2000: 443-457

[17] Susan Hohenberger, Guy N. Rothblum, abhi shelat, Vinod Vaikuntanathan. *Securely Obfuscating Re-encryption*. TCC 2007: 233-252

[18] Ben Lynn, Manoj Prabhakaran, Amit Sahai. *Positive Results and Techniques for Obfuscation*. EUROCRYPT 2004: 20-39

[19] Tal Malkin. *Personal Communication* (2006).

[20] Arvind Narayanan, Vitaly Shmatikov. *Obfuscated databases and group privacy.* ACM Conference on Computer and Communications Security 2005: 102-111

[21] Tatsuaki Okamoto. *On Relationships between Statistical Zero-Knowledge Proofs.* Journal of Computer and System Sciences 60(1): 47-108 (2000)

[22] Amit Sahai, Salil P. Vadhan. *A complete problem for statistical zero knowledge.* Journal of the ACM 50(2): 196-249 (2003)

[23] Hoeteck Wee. *On obfuscating point functions.* STOC 2005: 523-532

[24] Andrew Chi-Chih Yao. *Theory and Applications of Trapdoor Functions (Extended Abstract).* FOCS 1982: 80-91