J. Cryptol. (2014) 27: 480–505
DOI: 10.1007/s00145-013-9151-z

Journal of
CRYPTOLOGY

On Best-Possible Obfuscation*

Shafi Goldwasser[†]
Weizmann Institute of Science, Rehovot 76100, Israel
and
CSAIL, MIT, Cambridge, MA 02139, USA
shafi@theory.csail.mit.edu

Guy N. Rothblum[‡]
Microsoft Research, Silicon Valley, Mountain View, CA 94043, USA
rothblum@alum.mit.edu

Communicated by Oded Goldreich

Received 27 July 2007
Online publication 22 May 2013

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, pp. 1–18, 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 information that cannot be obtained from a black box. 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

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

[†] S. Goldwasser was supported by NSF grant CNS-0430450, NSF grant CFF-0635297 and a Cymerman–Jakubskind award.

[‡] Work of G.N. Rothblum was done while the author was a graduate student at CSAIL, MIT, and supported by NSF grant CNS-0430450 and NSF grant CFF-0635297.

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-Time 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.

Key words. Obfuscation.

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 guarantee 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. Loosely speaking, 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] also considered 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, which computes (inefficiently, by exhaustive search) a small circuit that is equivalent to the one being obfuscated. For some functionalities, and some choices of the revealed circuit, this is a great deal of information to give away. For example, when obfuscating a circuit computing an encryption functionality, the obfuscated circuit may always reveal a secret key that allows decryption. Such an obfuscator does not satisfy our intuitive notion of obfuscation (e.g. releasing only a key that allows encryption would give a more secure obfuscation), but this (inefficient) obfuscator could still satisfy the indistinguishability obfuscation definition of [2].

1.1. *This Work*

We propose a new notion of obfuscation, *best-possible obfuscation*, which avoids the black-box paradigm, and 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* information that cannot be obtained from a black-box, we require only that the (best-possible) obfuscated program leak as little information as possible, i.e. does not leak anything that cannot be obtained from any other code of an equivalent program. 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 information that cannot be obtained from a black-box (e.g. the code of a hard-to-learn function), as long as whatever it leaks is efficiently computable or *"learnable"*[1] 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 2.6, 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 computational restriction on the learner strengthens negative results (while our positive result on perfectly best-possible obfuscation applies also to unbounded *learners*).

In Proposition 2.9 we show that, in fact, the existence of an *inefficient* best-possible obfuscator implies the existence of an *efficient* one. Using this fact, some of our negative results for efficient best-possible obfuscation actually extend to inefficient best-possible obfuscation (but whenever we refer to obfuscation we mean efficient obfuscation unless we explicitly state otherwise).

[1] We note that previous work [2] used the term "learning from a program" to refer to information that can be computed from black-box access, whereas throughout this work we (extensively) use the term "learning from a program" to refer to information that can be computed from direct access to a circuit computing the program's functionality.

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 Sect. 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 itself a POBDD.

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). We note that Barak et al. showed that *efficient* indistinguishability obfuscators have a "competitiveness" property: For any pair of circuits, no obfuscator can make that pair of circuits "more indistinguishable" than an indistinguishability obfuscator (up to polynomial blowups in the security parameter). In particular, this also implies efficient best-possible obfuscation (as in Proposition 3.5).

For inefficient obfuscation, the definitions are very different: 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-Time Hierarchy collapses to its second level (see Theorem 4.1).

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 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-Time Hierarchy collapses to its second level. Corollary 4.2 (which follows from Theorem 4.1 and the aforementioned equivalence) states that also if there exists an *efficient* statistically secure indistinguishability obfuscator for the same simple circuit family, then the Polynomial-Time 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 [19] 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 5.2 we present a simple family of circuits with access to the random oracle that are provably hard to 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 Sect. 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) any circuit implementing a 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 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 [19] 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 [24] 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 does not 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 [7] presented a scheme that uses more general assumptions than those used by [6] (however, their solution is not 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 [18] gave a positive result for obfuscating a very natural and more complex cryptographic functionality (i.e., re-encryption) under a security-oriented definition. Hofheinz, Malone-Lee, and Stam [17] suggested a different security-oriented definition of obfuscation, argued its suitability for security applications, and provided positive and negative results.

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, 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 [10], 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 [8], 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 [10], which uses a random oracle for obtaining digital signatures from identification schemes. This method was shown to have the potential of leading to insecure signature schemes regardless of the possible implementation of the random oracle.

As mentioned above, in the context of obfuscation, Lynn, Prabhakaran and Sahai [19] explored which circuits could be obfuscated in the random oracle model. In the random oracle model, the view generated by the black-box simulator is indistinguishable when taken over a randomly selected oracle (Sect. 5 of this work considers the same model). We briefly review one of their constructions. They used the random oracle $\mathcal{R}$ to obfuscate a point function I_p (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 information about I_p (beyond what can be obtained via black-box access). 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 [21] gave a positive result for obfuscating databases in the random oracle model. In Sect. 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 Sect. 2. We compare our new definition of obfuscation with previous definitions in Sect. 3. In Sect. 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 Sect. 6.

2. Definitions and Discussion

2.1. *Notation and Preliminaries*

Let $[n]$ be the set $\{1, 2, \dots n\}$. 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; that is, for any positive polynomial $p()$, there exists some n_0 such that for all $n \geq n_0$ we have $f(n) < \frac{1}{p(n)}$.

Distributions, Ensembles, and Indistinguishability An ensemble $D = \{D_n\}_{n \in N}$ is a sequence of random variables, each ranging over $\{0,1\}^{\ell(n)}$ for some $l : \mathbb{N} \to \mathbb{N}$. 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

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

Definition 2.2 (Statistical Indistinguishability). Two probability ensembles D and F are $\varepsilon(n)$*-statistically close* or $\varepsilon(n)$*-statistically indistinguishable* if for every n we have $\Delta(D_n, F_n) \leq \varepsilon(n)$. We say that D and F are *statistically indistinguishable* if they are $\varepsilon(\cdot)$-statistically indistinguishable for a negligible function $\varepsilon(\cdot)$.

Definition 2.3 (Computational Indistinguishability [15,25]). Two probability ensembles D and F are computationally indistinguishable if for any polynomial-size circuit ensemble $\mathcal{C}$, where $\mathcal{C}_n$ takes as input 1^n and one sample s from either D_n or F_n, and outputs 0 or 1, the difference

$$\big| Pr_{s \sim D_n}\big[\mathcal{C}_n\big(1^n, s\big) = 1\big] - Pr_{s \sim F_n}\big[\mathcal{C}_n\big(1^n, s\big) = 1\big] \big|$$

is negligible in n.

2.2. *Previous Definitions of Obfuscation*

In the subsequent definitions, we consider a family $\mathcal{C}$ of deterministic 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. For all the notions of obfuscation discussed below, if the obfuscator $\mathcal{O}$ is a probabilistic polynomial-time Turing machine, 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 Sect. 5.

Black-Box Obfuscation The first notion of obfuscation we present is black-box obfuscation, due to [2]. The intuition behind the security guarantee is that the obfuscator makes it difficult to compute predicates (boolean functions) of the circuit being obfuscated. For any predicate $\pi : \mathcal{C} \to \{0, 1\}$ on circuits in the family $\mathcal{C}$, and any circuit $C \in \mathcal{C}$, computing $\pi(C)$ from an obfuscation of C should be no easier than computing $\pi(C)$ from black-box access to C. Formally this is captured by a simulation requirement (see below).

Definition 2.4 (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 $C \in \mathcal{C}_n$:
$$Pr\big[\exists x \in \{0, 1\}^n : \mathcal{O}(C)(x) \neq C(x)\big] \leq neg(n)$$
where the probability is over $\mathcal{O}$'s coins.[2]
- *Polynomial Slowdown*: There exists a polynomial $p(\cdot)$ such that for all input lengths n, 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 $C \in \mathcal{C}_n$:
$$\big|Pr\big[\mathcal{A}\big(\mathcal{O}(C)\big) = 1\big] - Pr\big[\mathcal{S}^C\big(1^n\big) = 1\big]\big| \leq neg(n)$$
where the probability is over the coins of the adversary, the simulator and the obfuscator.

Indistinguishability Obfuscation The second notion we present in this section is that of *indistinguishability obfuscation*, due also to [2]. Intuitively, the requirement here is that the obfuscations of functionally equivalent circuits should be indistinguishable (in a computational or information-theoretic sense).

Definition 2.5 (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 2.4, and also has the following property (*instead of* the virtual black-box property).

- *Computationally/Statistically/Perfectly Indistinguishable Obfuscation*: For all large enough input lengths n, for any circuit $C_1 \in \mathcal{C}_n$ 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 $\mathcal{O}(C_1)$ and $\mathcal{O}(C_2)$ are (respectively) computationally/statistically/perfectly indistinguishable.

[2] This is in fact a relaxation of the requirement in [2], which asked for a 0 probability of error.

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 $\mathcal{C}$. In the remainder of this work, when we refer to indistinguishability obfuscators we mean *efficient* indistinguishability obfuscators unless we explicitly note otherwise.

2.3. *New Definition of Obfuscation and Its Properties*

In this section we introduce a new definition for an obfuscator that does as well as is possible. This definition (similarly to Definition 2.5) avoids the black-box paradigm.

Informally, best-possible obfuscation, as presented in Definition 2.6 below, 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 2.6 (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 2.4, and also has the following property (*instead of* the virtual black-box property).

- *Computational/Statistical/Perfect Best-Possible Obfuscation*: For any PPTM learner $\mathcal{L}$, there exists a PPTM simulator $\mathcal{S}$ such that for any $poly(n)$-bit long auxiliary input z, and any two circuits $C_1, C_2 \in C_n$ *that compute the same function* and satisfy $|C_2| = |C_1|$, the two distributions $\mathcal{L}(z, \mathcal{O}(C_1))$ and $\mathcal{S}(z, C_2)$ are (respectively) computationally/statistically/perfectly indistinguishable. If the two distributions are always $\varepsilon(n)$-statistically indistinguishable, then we say the obfuscator is $\varepsilon(n)$-statistically best-possible.

Note that, unlike in Definition 2.4, here the simulator gets the explicit code of a circuit, and not just black-box access to its functionality. For statistical indistinguishability, unless noted otherwise, we only assume that the distinguisher's advantage $\varepsilon(n)$ (the statistical distance) is smaller than the constant $1/3$. Statistical obfuscation with $\varepsilon(n) = 1/3$ may be a weak security guarantee, but this strengthens negative results. The positive result of Proposition 3.2 yields a perfectly best-possible obfuscator (i.e. $\varepsilon(n) = 0$).

An Alternative Formulation We also present an equivalent formulation of best-possible obfuscation, which is often easier to work with for technical reasons. For this formulation (and elsewhere in this work) we often refer to the *"empty" learner*; this is the learner that simply outputs whatever obfuscation it gets as input. The alternative formulation focuses only on this empty learner, doing away with the need to consider general learners. It only requires the existence of an *efficient* simulator for the empty learner. That is, a simulator $\mathcal{S}$ such that for circuits C_1, C_2 of identical size and functionality, the distributions $\mathcal{O}(C_1)$ and $\mathcal{S}(C_2)$ are indistinguishable.[3] We feel, though, that this formulation (provided next) loses some of the intuitive appeal of Definition 2.6.

Definition 2.7 (Computational/Statistical/Perfect Best-Possible Obfuscation, alternative formulation). There exists a PPTM simulator $\mathcal{S}$, such that for any two circuits $C_1, C_2 \in C_n$ *that compute the same function* and satisfy $|C_2| = |C_1|$, the two distributions $\mathcal{O}(C_1)$ and $\mathcal{S}(C_2)$ are (respectively) computationally/statistically/perfectly indistinguishable.

Proposition 2.8. *The best-possible obfuscation properties of Definitions* 2.6 *and* 2.7 *are equivalent.*

Proof. The best-possible obfuscation property of Definition 2.6 immediately implies that of Definition 2.7 by examining the simulator for the "empty learner" (the learner that simply outputs its input).

In the other direction, if the obfuscator satisfies the property of Definition 2.8, then there exists an efficient simulator $\mathcal{S}$ for the "empty" learner. This implies an efficient simulator $\mathcal{S}'$ for every efficient learner $\mathcal{L}$ and auxiliary input z: On input C_2, the simulator $\mathcal{S}'$ computes $\mathcal{S}(C_2)$ and outputs the result of $\mathcal{L}(z, \mathcal{S}(C_2))$. Since 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 (perfectly, statistically or computationally), also $\mathcal{L}(z, \mathcal{O}(C_1))$ and $\mathcal{S}'(C_2) = \mathcal{L}(z, \mathcal{S}(C_2))$ must be indistinguishable. □

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

For the general question of obfuscation (not necessarily in the best-possible sense), in applications we would want an efficient obfuscator. Still, 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.

We observe that for the best-possible notion of obfuscation, the existence of an *inefficient* best-possible obfuscator (perfect, statistical or computational), implies the existence of an *efficient* one that uses the simulator to obfuscate (albeit with some possible loss in the security parameters). This is formalized in the next proposition.

Proposition 2.9. *Let $\mathcal{O}$ be an inefficient best-possible obfuscator for a circuit family $\mathcal{C}$, where $\mathcal{O}$ can be a perfect, $\varepsilon(n)$-statistical or computational. Then, there exists an* efficient *best-possible obfuscator $\mathcal{O}'$ for $\mathcal{C}$, where $\mathcal{O}'$ is perfect, $(\varepsilon(n) \cdot polylog(n))$-statistical, or computational (respectively).*

[3] In fact, this shows that efficient indistinguishability obfuscation implies efficient best-possible obfuscation, by taking the indistinguishability obfuscator $\mathcal{O}$ to be both the best-possible obfuscator and its simulator (i.e. circuit C_1 will be obfuscated by $\mathcal{O}(C_1)$, whereas this code will be simulated by $\mathcal{O}(C_2)$). See Proposition 3.5 for a full proof.

Proof. In all three cases (perfect, statistical, and computational), we construct $\mathcal{O}'$ using the (efficient) simulator $\mathcal{S}$ of $\mathcal{O}$ for the "empty learner" (i.e. the simulator of Definition 2.7).

The Case of Perfectly Best-Possible We construct $\mathcal{O}'$ to simply run the (efficient) simulator $\mathcal{S}$ on its input, and simulate $\mathcal{O}'$ by the same simulator $\mathcal{S}$. The simulator's output on two equivalent circuits of the same size is perfectly indistinguishable (i.e., identical) by the best-possible obfuscation guarantee, and it clearly satisfies polynomial slowdown. The main concern is maintaining functionality, but $\mathcal{O}'$ almost always maintains the circuit's functionality (because $\mathcal{O}$ almost always does).

The Case of $\varepsilon(n)$-Statistically Best-Possible The simulator's output is $\varepsilon(n)$-statistically close to the obfuscators. With all but negligible probability, the obfuscator's output maintains functionality, and so the simulator maintains functionality with probability at least $1-(\varepsilon(n)+neg(n))$, where we assume $\varepsilon(n) \leq 1/3$.[4] We construct $\mathcal{O}'$ that runs the simulator $polylog(n)$ times on its input. Each of these executions outputs a circuit, and $\mathcal{O}'$'s output is a circuit that runs all these circuits on its input, and outputs the majority answer. Each execution outputs a circuit that maintains functionality with probability at least $3/5$. Thus with all but at negligible probability, a majority of the $polylog(n)$ outputs maintain functionality, and so $\mathcal{O}'$'s output maintains functionality. Since S is a simulator for the best-possible obfuscator $\mathcal{O}$, and by a hybrid argument, $\mathcal{O}'$ is an $(\varepsilon(n) \cdot polylog(n))$-best-possible obfuscator.

The Case of Computationally Best-Possible We begin by observing that while in the computational case the simulator's output may *never* fully maintain functionality of the input circuit, it does satisfy a weaker guarantee. For any computationally best-possible obfuscator, for every input circuit, for every input, with all but negligible probability the simulator's output maintains functionality *on that input*. This is because otherwise the "bad" inputs, those for which with non-negligible probability functionality is not maintained, can be "hard-wired" into a (non-uniform) distinguisher that distinguishes the obfuscator's and simulator's outputs (recall that with overwhelming probability the obfuscator maintains functionality for *every input*).

We construct the obfuscator $\mathcal{O}'$, as we did in the statistical case, by running the simulator many times and outputting the circuit that takes the majority answer. In this case, we take $O(n)$ copies. This means that for each input, functionality is maintained with probability 2^{-2n}, and by a union bound with probability at least 2^{-n} the output of $\mathcal{O}'$ maintains functionality for all inputs. By a hybrid argument $\mathcal{O}'$ is also a computationally best-possible obfuscator. □

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].

[4] We do not know whether the result of Proposition 2.9 applies to all functions $\varepsilon(n)$ bounded away from 1.

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. By Proposition 2.8, 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

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

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, i.e. with the output itself being a POBDD, is achievable, but there are simple functionalities that are provably impossible to black-box obfuscate *within the class* (i.e. black-box obfuscation with a POBDD output is impossible).

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, including a "root" vertex, and terminal vertices (without children). Each non-terminal vertex v is labeled with the name of an input bit ℓ_v (e.g. $\ell_v = x_1$ or $\ell_v = x_5$), the terminal vertices are labeled 0 or 1, and for each non-terminal vertex, one of its outgoing edges is labeled 0 and the other is labeled 1. An input $x \in \{0,1\}^n$ is accepted by an OBDD if and only if after removing, for every vertex v with label $\ell_v = x_i$, its outgoing edge labeled by $1 - x_i$ (i.e. removing the outgoing edge whose label is not equal to the value of the input bit labeling the vertex), 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 input bits labeling vertices 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 (which 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.

Best-Possible Obfuscation of POBDDs 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 on input a POBDD P, computes (efficiently) P's 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. □

Note this construction gives a very strong information-theoretic obfuscation: Anything that can be learned from the obfuscation even by an *inefficient* learner, can also be simulated perfectly by an *efficient* simulator (from any functionally equivalent circuit of the same size). This is stronger than the perfectly best-possible obfuscation requirement we made in Definition 2.6 (the weaker requirement made there strengthens negative results).

No Black-Box Obfuscation for 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.*

Proof. By the results of [5], from any POBDD (obfuscated or not) we can extract a canonical smallest equivalent POBDD. We will show, however, that for some (simple and natural) functions that are computable by POBDDs, computing a canonical smallest equivalent POBDD is impossible *from black-box access*. This implies that POBDDs that compute such functions cannot be black-box obfuscated, the details follow.

We proceed by considering the 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. Point functions are computable by (linear size) POBDDs. Using Bryant's algorithm [5], from any POBDD computing the point function I_p, we can learn the canonical POBDD P_p that computes the function I_p. On examination this canonical P_p is a very simple POBDD: It has $n+1$ vertices, two of which are terminal vertices labeled 0 and 1. The POBDD P_p has a path of length n from the root to terminal vertex 1, where the edges in the path are labeled by all n bits of the input x, i.e. $(x_1, x_2, \dots, x_n)$ in increasing order. The literal labeling the ith edge is negated if and only if $p_i = 0$. This path accepts if $x = p$. In addition, every vertex along this path also has an edge leading to the terminal vertex 0 (the rejecting vertex). This rejecting edge from vertex i is labeled with the ith input bit (just like the edge from vertex i that is on the "accepting path"), and the literal is negated if and only if $p_i = 1$.

Now since the canonical POBDD P_p can be computed from *any* POBDD that computes I_p, it can also always be computed from the obfuscator's output. If we analyze P_p we can easily extract every bit of the point p by examining whether or not the literals labeling edges on the accepting path are negated. Thus, for any $p \in \{0,1\}^n$, from any (allegedly obfuscated) POBDD computing I_p, we can extract *all the bits* of p.

On the other hand, no polynomial time black-box simulator that only gets black box access to I_p for a random point p can extract any bit of p with probability significantly greater than $1/2$. We conclude that there exists no black-box obfuscator that obfuscates POBDDS computing point functions *as POBDDs*. □

We mention that many other natural languages computable by POBDDs cannot be black-box obfuscated as POBDDs. Black-box obfuscation of POBDDs by 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-Time Hierarchy collapses. In fact, inefficient indistinguishability obfuscators exist unconditionally (see [2]). On the other hand, building inefficient best-possible obfuscators (for arbitrary circuits) remains a fascinating open question, and in particular it would lead (up to some loss in the security parameters) to a construction of *efficient* best-possible obfuscators (see Sect. 2.3). 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 $\mathcal{C}$.*

Proof. The intuition is that (by definition) the best-possible simulator's output distributions on any two functionally equivalent circuits (of the same size) are indistinguishable. Since the best-possible obfuscator's output on any circuit is indistinguishable from the simulator's, this means that the obfuscator's output distributions on two equivalent circuits (of the same size) will also be indistinguishable.

More formally, let δ be a bound on the distinguishability (computational or statistical) of the best-possible obfuscator's output and the simulator's output. Let $\mathcal{S}$ be the simulator for the "empty learner" $\mathcal{L}$ (the learner that just outputs its input) as guaranteed Definition 2.7. Then by a hybrid argument, the distinguishability (computational or statistical) of the distributions $\mathcal{O}(C_1)$ and $\mathcal{O}(C_2)$ is at most the sum of the distinguishability (computational or statistical) of the two distributions $\mathcal{O}(C_1)$ and $\mathcal{S}(C_2)$ and the distinguishability (computational or statistical) of the two distributions $\mathcal{O}(C_2)$ and $\mathcal{S}(C_2)$. When the circuits C_1 and C_2 are of the same size and compute the same functionality, the distributions $\mathcal{O}(C_1)$ and $\mathcal{S}(C_2)$ are δ-indistinguishable (computationally or statistically) by the best-possible obfuscation guarantee. The same indistinguishability holds for the distributions $\mathcal{O}(C_2)$ and $\mathcal{S}(C_2)$. We conclude that $\mathcal{O}$ is an indistinguishability obfuscator, with a 2δ bound on the (computational or statistical) distinguishability. □

As noted above, if we restrict our attention to efficient obfuscators, indistinguishability obfuscators are also best-possible obfuscators. This is shown in Proposition 3.5 below.

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 perfectly/statistically/computationally* (*respectively*) *best-possible obfuscator for $\mathcal{C}$.*

Proof. Following the alternative formulation of Definition 2.7 (best-possible obfuscation with an "empty learner"), the intuition is that the (efficient) indistinguishability obfuscator itself can be used as a best-possible simulator. The obfuscator's and the simulator's outputs are indistinguishable (on any pair of functionally equivalent circuits) by the indistinguishability obfuscation guarantee.

More formally, let $\mathcal{O}$ be an efficient indistinguishability obfuscator. We construct an efficient simulator $\mathcal{S}$, s.t. when it gets a circuit C_2 it runs $\mathcal{O}(C_2)$. 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. 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 $\mathcal{O}(C_1)$ and $\mathcal{S}(C_2) = \mathcal{O}(C_2)$ are perfectly/statistically/computationally indistinguishable (because $\mathcal{O}$ is an indistinguishability obfuscator). Thus $\mathcal{O}$ is also a best-possible obfuscator. □

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 end this subsection by observing that if $P = NP$, then it is possible 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-Time 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. Assuming $P = NP$, we show that 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-Time Hierarchy). To see this, fix a binary encoding for circuits (i.e., a way of representing circuits as binary strings). Observe that a co-NP machine can test whether two circuits are equivalent, rejecting if the two circuits are *not* equivalent (i.e. there exists an input on which their outputs differ). Now, given an input circuit C, a prefix $p \in \{0, 1\}^*$, and a target size s, an NP machine with a co-NP oracle can find whether there *exists* a circuit whose binary encoding begins with prefix p, is of size s, and is equivalent to C. This is done by "guessing" an encoding with prefix p of size s, testing its equivalence to C using the co-NP oracle, and accepting if the two circuits are equivalent. This problem is thus in the second level of the Polynomial-Time Hierarchy, and if $P = NP$ it can be solved in polynomial time.

Given a polynomial-time algorithm, we can now *find* the smallest lexicographically first equivalent circuit. First, find the size of the smallest equivalent circuit by trying all possible (polynomially many) target sizes. For each target size, test whether there exists an equivalent circuit beginning with the empty prefix. Once the minimal size is known, we can find the lexicographically first equivalent circuit of this size. This is done by fixing the bits of the circuit one-by-one and using them as the prefix (i.e., we first compute whether there exists an equivalent circuit with prefix 0, if so we look for one with prefix 00, if such a circuit does not exist then we know one exists with prefix 01, now we try prefix 010 etc.). We terminate when we have found the encoding of the lexicographically first equivalent circuit of this minimal size. If $P = NP$, then this is a polynomial-time procedure.

The smallest lexicographically first circuit is unique, and depends only on the circuit's functionality. Thus, as Barak et al. [2] note, the above extraction procedure is a perfectly indistinguishable obfuscation of C. We conclude that 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. □

We note once more that, as observed in [2], even if $P \neq NP$, the procedure described in the proof of Proposition 3.6 gives an *inefficient* indistinguishability obfuscator. It remains unclear, however, whether we can get an inefficient best-possible obfuscator, as the simulator, which is required to be efficient, can no longer run the above "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 Sect. 3 it was shown that if $P = NP$ then every polynomial-sized circuit can be perfectly best-possible obfuscated, and thus we cannot hope for an *unconditional* impossibility result. We show that the condition $P = NP$ is (somewhat) 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-Time Hierarchy collapses to its second level. This result gives evidence that statistically best-possible obfuscation is impossible 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*, *then the Polynomial-Time 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 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 (and denote its size by $|\varphi|$). 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 $\mathcal{O}(\varphi)$ and $\mathcal{S}(\varphi)$, as well as $\mathcal{S}(\psi)$ and $\mathcal{O}(\psi)$, are 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 $\mathcal{O}(\varphi)$ and $\mathcal{O}(\psi)$ are disjoint (they are distributions of circuits with different functionalities). Thus, the distributions $\mathcal{S}(\varphi)$ and $\mathcal{S}(\psi)$ of the (perfect) simulator's output are also disjoint, and there exist no (r, r') such that $\mathcal{S}(\varphi, r) = \mathcal{S}(\psi, r')$.

In conclusion, we find that a perfectly best-possible obfuscator for 3-CNF formulas that perfectly preserves functionality implies that *NP* equals co-NP.

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*. By the results of Fortnow [11], Aliello and Håstad [1], and Boppana, Håstad and Zachos [4], this collapses the Polynomial-Time 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 [23]. It is a promise problem that refers to pairs (C_1, C_2) of polynomial-size circuits, which 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 $\frac{2}{3}$, whereas no-instances are pairs of circuits for which this distance is at most $\frac{1}{3}$. Sahai and Vadhan [23] prove that *SD* is complete for the complexity class SZK (consisting of languages with statistical zero-knowledge interactive proofs). The complement of *SD*, the problem $\overline{SD}$ is thus also in SZK (and is in fact also complete), by the result of Okamoto [22] (see also Goldreich and Vadhan [12] that SZK is closed under complement.

The results of Fortnow [11], 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-Time 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 $\overline{3SAT}$).

Outline We assume that the family of 3-CNF formulas can be statistically best-possible obfuscated. 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, which collapses the Polynomial-Time 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* $\mathcal{S}(\phi_1)$ and $\mathcal{S}(\phi_2)$, 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 (non-matching) sufficient and necessary conditions for 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, whereas a *necessary* condition is having a statistical zero knowledge proof for the equivalence problem.

Finally, a corollary of Theorem 4.1 is that the same class of 3-CNF formulas cannot be statistically *indistinguishability* obfuscated in polynomial time, unless the Polynomial-Time 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-Time 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-Time 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, the function $\mathcal{R}$ maps $\{0,1\}^n$ 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*[5] 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 2.4, 2.5, 2.6), in 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.

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

In the *Virtual Black-box* requirement of Definition 2.4, 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 $C \in \mathcal{C}_n$:

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

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 2.5 and the *Best-Possible Obfuscation* requirement of Definition 2.6, all distributions are taken over 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 [19]), 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.

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 5.2. *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 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 $\mathcal{R}(x \circ r)$. In fact, with all but negligible probability over p, r, x and the coins of the obfuscator and the obfuscated circuit, the obfuscated circuit $\mathcal{O}(C_p^r)$, when activated on input x, queries the random oracle on the point $(x \circ r)$. This is shown in Claim 5.3.

On the other hand, for a uniformly random pad s, the probability that $\mathcal{O}(C_p^r)$, when activated on input x, queries the random oracle on the point $(x \circ s)$ is negligible. This is simply because the number of oracle calls made by $\mathcal{O}(C_p^r)$ is polynomial. See Claim 5.4.

We conclude that for uniformly random p, r, s, the obfuscations $\mathcal{O}(C_p^r)$ and $\mathcal{O}(C_p^s)$ can be distinguished. The distinguisher $\mathcal{D}_s$ runs its input (obfuscated) circuit on a random input x, and accepts iff the circuit queries the random oracle at point $(x \circ s)$. By the above, with high probability over p, r, s and the coins of the distinguisher and obfuscated circuit, it will accept $\mathcal{O}(C_p^s)$ but reject $\mathcal{O}(C_p^r)$. This is a contradiction to the best-possible obfuscation property. See the full details below.

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 for the circuit C, and then runs C on input x with the randomness it chose. If in this execution C makes the oracle query $x \circ s$ then D_s outputs 1, otherwise D_s outputs 0.

Claim 5.3. *Let $\mathcal{O}$ be any efficient obfuscator.*

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

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

$$Pr\big[\mathcal{D}_s\big(\mathcal{O}\big(C_p^r\big)\big) = 1\big] \le neg()$$

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 5.3 and 5.4, we conclude that for any efficient obfuscator, for a random $s, r, p \in \{0,1\}^n$, the distinguisher $\mathcal{D}_s$ distinguishes $\mathcal{O}(C_p^s)$ from $\mathcal{O}(C_p^r)$, a contradiction to the best-possible obfuscation requirement. We conclude that the circuit family $\{C_p^r\}$ 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 [19] 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 ($\{C_p^r\}$) hides information about the point. Malkin [20], was the first to ask whether *any* point function implementation can be black-box obfuscated.

Theorem 5.2 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 $\{C_p^r\}$ (namely the pad r) is not related to the point p. We believe, however, that it is 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 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 $\{C_p^r\}$. 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 $f_z(p, r) = (z, z^{p \circ r})$.[6] Even if on every input x the obfuscated circuit *always* computes the value $f_z(x, r) = (z, z^{x \circ r})$, there is no guarantee 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 Sect. 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 5.2), 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 Sect. 5), whereas restricting the output representation makes the task of

[6] 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.

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 [20], who asked whether *any* point function implementation can be black-box obfuscated in the standard model. A relaxed (but related[7]) formulation of this question is whether the family of polynomial-size circuits computing point functions can be best-possible obfuscated. The proof of Theorem 5.2 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 2.6) 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.

Weaker Variants In light of the negative results of Theorems 4.1 and 5.2 it is interesting to consider *weaker* variants of best-possible obfuscation (Definition 2.6). While the variants below lose some of the appealing intuitive appeal of Definition 2.6, 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 5.2 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 [24] relax the virtual black-box requirement, by requiring only *polynomially* small indistinguishability between the output of an adversary and its simulator. Moreover, they allow the simulator's size

[7] 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.

to depend (polynomially) on this indistinguishability parameter. We note that negative results in this work (Theorems 4.1 and 5.2) hold even if we require only polynomially small (or, in some of the results, even constant) indistinguishability, and allow the simulator's size to depend (polynomially) on the indistinguishability parameter.

- *Weaker Functionality.* Definition 2.6 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. The negative result of Theorem 5.2 (impossibility in the random oracle model) applies 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.

Acknowledgements

We thank Yael Tauman Kalai and Tali Kaufman for helpful and enjoyable discussions. Thanks also to the anonymous TCC 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 5.2. Finally, we are very grateful to Oded Goldreich for his many helpful and insightful comments on this work and its presentation.

References

[1] W. Aiello, J. Håstad, Statistical zero-knowledge languages can be recognized in two rounds. *J. Comput. Syst. Sci.* **42**(3), 327–345 (1991)

[2] B. Barak, O. Goldreich, R. Impagliazzo, S. Rudich, A. Sahai, S.P. Vadhan, K. Yang, On the (Im)possibility of obfuscating programs, in *CRYPTO 2001* (2001), pp. 1–18

[3] M. Bellare, P. Rogaway, Random oracles are practical: a paradigm for designing efficient protocols, in *ACM Conference on Computer and Communications Security 1993* (1993), pp. 62–73

[4] R.B. Boppana, J. Håstad, S. Zachos, Does co-NP have short interactive proofs? *Inf. Process. Lett.* **25**(2), 127–132 (1987)

[5] R.E. Bryant, Graph-based algorithms for boolean function manipulation. *IEEE Trans. Comput. C* **35**(8), 677–691 (1986)

[6] R. Canetti, Towards realizing random oracles: hash functions that hide all partial information, in *CRYPTO 1997* (1997), pp. 455–469

[7] R. Canetti, D. Micciancio, O. Reingold, Perfectly one-way probabilistic hash functions (Preliminary version), in *STOC 1998* (1998), pp. 131–140

[8] R. Canetti, O. Goldreich, S. Halevi, The random oracle methodology, revisited. *J. ACM* **51**(4), 557–594 (2004)

[9] Y. Dodis, A. Smith, Correcting errors without leaking partial information, in *STOC 2005* (2005), pp. 654–663

[10] A. Fiat, A. Shamir, How to prove yourself: practical solutions to identification and signature problems, in *CRYPTO 1986* (1986), pp. 186–194

[11] L. Fortnow, The complexity of perfect zero-knowledge, in *Advances in Computing Research*, vol. 5, ed. by S. Micali (JAI Press, Greenwich, 1989), pp. 327–343

[12] O. Goldreich, S.P. Vadhan, Comparing entropies in statistical zero knowledge with applications to the structure of SZK, in *IEEE Conference on Computational Complexity 1999* (1999), p. 54

[13] S. Goldwasser, Y. Tauman Kalai, On the (In)security of the Fiat–Shamir paradigm, in *FOCS 2003* (2003), pp. 102–113

[14] S. Goldwasser, Y. Tauman Kalai, On the impossibility of obfuscation with auxiliary input, in *FOCS 2005* (2005), pp. 553–562

[15] S. Goldwasser, S. Micali, Probabilistic encryption and how to play mental poker keeping secret all partial information, in *STOC 1982* (1982), pp. 365–377

[16] S. Hada, Zero-knowledge and code obfuscation, in *Asiacrypt 2000* (2000), pp. 443–457

[17] D. Hofheinz, J. Malone-Lee, M. Stam, Obfuscation for cryptographic purposes, in *TCC 2007* (2007), pp. 214–232

[18] S. Hohenberger, G.N. Rothblum, A. Shelat, V. Vaikuntanathan, Securely obfuscating re-encryption, in *TCC 2007* (2007), pp. 233–252

[19] B. Lynn, M. Prabhakaran, A. Sahai, Positive results and techniques for obfuscation, in *EUROCRYPT 2004* (2004), pp. 20–39

[20] T. Malkin, Personal communication (2006)

[21] A. Narayanan, V. Shmatikov, Obfuscated databases and group privacy, in *ACM Conference on Computer and Communications Security 2005* (2005), pp. 102–111

[22] T. Okamoto, On relationships between statistical zero-knowledge proofs. *J. Comput. Syst. Sci.* **60**(1), 47–108 (2000)

[23] A. Sahai, S.P. Vadhan, A complete problem for statistical zero knowledge. *J. ACM* **50**(2), 196–249 (2003)

[24] H. Wee, On obfuscating point functions, in *STOC 2005* (2005), pp. 523–532

[25] A.C.-C. Yao, Theory and applications of trapdoor functions (extended abstract), in *FOCS 1982* (1982), pp. 80–91