On Obfuscating Point Functions

Hoeteck Wee*
Computer Science Division
University of California, Berkeley

Jan 2005

Abstract

We study the problem of obfuscation in the context of point functions (also known as delta functions). A point function is a Boolean function that assumes the value 1 at exactly one point. Our main results are as follows:

1. We provide a simple construction of efficient obfuscators for point functions for a slightly relaxed notion of obfuscation - wherein the size of the simulator has an inverse polynomial dependency on the distinguishing probability - which is nonetheless impossible for general circuits. This is the first known construction of obfuscators for a non-trivial family of functions under general computational assumptions. Our obfuscator is based on a probabilistic hash function constructed from a very strong one-way permutation, and does not require any set-up assumptions. Our construction also yields an obfuscator for point functions with multi-bit output.
2. We show that such a strong one-way permutation - wherein any polynomial-sized circuit inverts the permutation on at most a polynomial number of inputs - can be realized using a random permutation oracle. We prove the result by improving on the counting argument used in [GT00]; this result may be of independent interest. It follows that our construction yields obfuscators for point functions in the non-programmable random permutation oracle model (in the sense of [N02]). Furthermore, we prove that an assumption like the one we used is necessary for our obfuscator construction.
3. Finally, we establish two impossibility results on obfuscating point functions which indicate that the limitations on our construction (in simulating only adversaries with single-bit output and in using non-uniform advice in our simulator) are in some sense inherent. The first of the two results is a consequence of a simple characterization of functions that can be obfuscated against general adversaries with multi-bit output as the class of functions that are efficiently and exactly learnable using membership queries.

We stress that prior to this work, what is known about obfuscation are negative results for the general class of circuits [BGI+01] and positive results in the random oracle model [LPS04] or under non-standard number-theoretic assumptions [C97]. This work represents the first effort to bridge the gap between the two for a natural class of functionalities.

*`hoeteck@cs.berkeley.edu`. Work supported by US-Israel BSF Grant 2002246.

1 Introduction

1.1 Background

The elegant cryptographic framework for obfuscation was established in the seminal work of Barak, et al. in [BGI+01]. The heart of the framework is the definition of the virtual black-box property, which formalizes the obfuscation requirement using the simulation paradigm: anything one can efficiently compute given an obfuscated program (or circuit), one could also efficiently compute from observing the input/output behavior of the program (or circuit). In addition, [BGI+01] also showed that efficient circuit obfuscators do not exist (unconditionally). However, the work did not rule out the existence of efficient obfuscators for restricted but nonetheless useful classes of circuits, nor did it provide any evidence of the existence of such obfuscators, except for a remark that the constructions of Canetti et al. in [CMR98] can be viewed as some form of obfuscators for point functions.

An obfuscator for point functions (which may also be viewed as obfuscating the equality functionality) is exactly what we need for the Unix password hashing algorithm [WG00] – the virtual black-box property guarantees that any adversary upon seeing the encrypted password cannot do much better than the dictionary attack. Our work provides a means of understanding the security of the password hashing algorithm wherein the underlying cryptographic hash function is only assumed to be a very strongly one-way permutation, and not a random oracle. In fact, the recent work of Lynn, et al. in [LPS04] demonstrated that obfuscators for point functions do exist in the random oracle model. The obfuscation algorithm applies the random oracle to the secret value (namely the value at which the point function assumes the value 1) and stores the output into a program. The program upon receiving an input applies the random oracle to the input and outputs 1 if the oracle output is equal to the value that is stored and 0 otherwise. Our approach is similar, except that we replace the random oracle with a probabilistic hash function constructed from a very strong one-way permutation.

1.2 Other Related Work

In work preceding [CMR98], Canetti [C97] presented several equivalent notions of obfuscation in the specific context of point functions, one of which is the same as the relaxed notation of virtual black-box used in this paper. In addition, Canetti constructed obfuscators for point functions based on a very strong variant of the Decisional Diffie-Hellman problem, that the problem remains hard if one of the inputs comes from any distribution of superlogarithmic min-entropy (instead of the uniform distribution)[1]. We note that this assumption has a flavor of pseudorandomness, whereas we start with a hardness assumption from which we derive pseudorandomness.

The paper of [CMR98] generalizes the work of [C97] to construct hash functions satisfying a different notion of secrecy under standard cryptographic assumptions. Combined with the assumption and analysis in this paper, the construction of [CMR98] would yield an obfuscator with a non-uniform simulator of super-polynomial size (even for inverse polynomial distinguishing probability). Moreover, the hash function is only computationally collision-resistant, thus the resulting obfuscator only achieves a weaker notion of "computational functionality" (which is weaker than approximate functionality). On the other hand, our construction is simpler than that of [CMR98], and yields a hash function that is statistically collision-resistant, and an obfuscator with a non-uniform simulator of polynomial size (for inverse polynomial distinguishing probability).

[1]The precise assumption is as follows: for any well-spread distribution ensemble $\{X_q\}$ with (superlogarithmic) min-entropy $\omega(\log \log q)$ where the domain of X_q is Z_q^*, for a drawn from X_q and for $b, c \in_R Z_q^*$, we have $\{g^a, g^b, g^{ab}\}$ and $\{g^a, g^b, g^c\}$ computationally indistinguishable.

1.3 Contributions and Perspective

On the whole, our work constitutes an effort towards bridging the gap between positive and negative results for program obfuscation.

1.3.1 Positive results for obfuscation

Our main positive result (Theorem 3.5) is a new construction of a probabilistic hash function starting from a very strong one-way permutation π as follows:

$$h(x; \tau_1, \ldots, \tau_{3n}) = (\tau_1, \ldots, \tau_{3n}, \langle x, \tau_1 \rangle, \langle \pi(x), \tau_2 \rangle, \ldots, \langle \pi^{3n-1}(x), \tau_{3n} \rangle)$$

This hash function simultaneously achieves statistical collision-resistance and very strong pseudorandom properties, and can be used to replace the random oracle in the work of [LPS04] to yield the first construction of obfuscators under general computational assumptions. This is also one of the few instances in cryptography (another bring [GHR99]) where a random oracle can be replaced by a cryptographic construction. Both the construction and the analysis are largely inspired by the techniques used in [BM84, GL89, C97, CMR98, F99], but is much simpler and self-contained than [CMR98, F99] in that we do not require the [GGM84] construction of pseudorandom functions. Unlike [BM84, GL89], we do not append $\pi^{3n}(x)$ to the output of our hash function; the reason will be clear from examining the hybrid argument used in the proof for proving pseudorandomness.

The weak simulator for our obfuscator construction works as follows: it has as non-uniform advice a set L that depends on the adversary and specifies the point functions for which the obfuscation reveals "too much" information to the adversary. On oracle access to a point function, the simulator checks if it corresponds to a function in L, and if so, it can simulate the adversary on a random obfuscation of that function. Otherwise, it simulates the adversary on a random string (which may not even correspond to a valid hash), and this works because the hash function satisfies a very strong pseudorandomness property. The construction of the simulator and L is inspired by ideas from [C97], but the analysis is much simpler.

1.3.2 The one-way permutation assumption

Our construction uses a non-standard assumption: that there exists a strong one-way permutation wherein any polynomial-sized circuit inverts the permutation on at most a polynomial number of inputs (Assumption 3.1). Standard cryptographic assumptions assert hardness with respect to "work" (i.e., time over success probability) $n^{\omega(1)}$. Here, we require "work" $2^{n-O(\log n)}$, whereas each function can be inverted with work 2^n. We stress that we do not have a strong intuition about whether this assumption holds. However, that we do not know any algorithm that does better than brute force for the circuit satisfiability problem (CSAT) (i.e., a probabilistic $2^{n-\omega(\log n)} \cdot \mathrm{poly}(\mathsf{circuitsize})$-time algorithm where n is the number of variables)[2] and that even a collapse of BQP to BPP only offers a quadratic speed-up (i.e., a $2^{n/2} \cdot \mathrm{poly}(\mathsf{circuitsize})$ running time due to Grovers) offer some evidence that the assumption is not entirely unreasonable. We also show that an assumption of this kind is in fact necessary for the public-coin obfuscators that we construct (Theorem 4.4), and that such a strong one-way permutation can be realized with random permutation oracle (Theorem 4.1). The latter yields an obfuscator construction in the non-programmable random permutation oracle model.

Our positive results suggest that it may in fact be possible to modify existing cryptographic constructions and protocols proven secure in the random oracle model to obtain new ones that are provably secure under an assumption similar to the one we made. While our assumption may not be deemed reasonable and has the "flavor" of a random oracle, it is still weaker as we only assume for that a specific task - that of inverting the given permutation at a random input - no adversary can do much better than treating the permutation as a black-box. Note that we could potentially provide empirical observations supporting the existence of such

[2]Note that reducing the CSAT instance to a $\mathsf{3SAT}$ instance and then running Schöning's algorithm does not help, since the number of variables in the $\mathsf{3SAT}$ instance depends on the circuit size.

a strong one-way permutation, unlike a random oracle, which in itself already constitutes an obfuscation of a non-trivial and extremely powerful functionality. In fact, it is not even clear what it means to provide empirical evidence for the existence of a function that behaves like a random oracle [CGH98].

1.3.3 Limitations of our simulator construction

1. (weak simulator) We allow the size of the simulator to have an inverse polynomial dependency on the distinguishing probability, as in the definition of ϵ-knowledge in [DNS98]. We argue that this is sufficient to capture an intuitive and appealing notion of obfuscation: an adversary can approximate whatever he learns from seeing an obfuscated circuit by any inverse polynomial function with a polynomial blow-up in its running time and advice. Furthermore, we note that [BGI+01] rules out obfuscation of general circuits even under this weaker simulator requirement.
2. (simulator with non-uniform advice) Note that most cryptographic works use uniformly black-box reductions in their proof of security, so that the proof yields a uniform simulator for uniform adversaries. This is not the case in our work. Our simulator is inherently non-uniform, as we hard-wire into it non-uniform advice about the adversary which we do not know how to efficiently compute even given the description of the circuit A; instead, we know that such advice exists and has a succinct description based on our assumption about the underlying one-way permutation. We feel that the use of non-uniformity is not a major short-coming in our simulator, since we can interpret our obfuscator as having the property that "anything an adversary can efficiently compute given the obfuscated circuit, he could also efficiently compute from observing the input/output behavior of the circuit, when given a small amount of help". We also prove that using non-uniform advice (about the adversary) is in fact necessary for constructions using black-box access to the adversary. Unfortunately, the state-of-the-art non-black-box techniques [B01] require interaction with the adversary.
3. (general adversaries) Our simulator only applies to an adversary that computes a $\{0,1\}$-valued function on its input; that is, an adversary that is trying to decide some property of the original circuit. Note that simulating such an adversary already encapsulates semantic security. As pointed out in [BGI+01], we would like (for positive results) for the simulator to satisfy a stronger requirement, namely to simulate the view of the adversaries. We show that this is impossible to achieve for point functions via a relativizing argument. In fact, we show a stronger result, that every family of circuits that can be obfuscated against general adversaries is efficiently and exactly learnable using membership queries. Note that this does not contradict [LPS04] which constructs obfuscators for point functions satisfying this stronger requirement, because they allow the simulator to choose the coin tosses of the random oracle (refer to [P03] for a more extensive discussion of the issues associated with simulation in the random oracle model).

2 Preliminaries

We will adopt the standard way of modeling efficient adversary strategies as a family of probabilistic polynomial-sized circuits. Therefore, a polynomial-sized adversary A or a PPT A will in fact refer to a family of probabilistic polynomial-sized circuits. Similarly, computationally indistinguishable refers to indistinguishability by non-uniform polynomial-sized adversaries.

2.1 Notation and definitions

We write U_n to denote the uniform distribution over $\{0,1\}^n$, and $\text{neg}(n)$ to denote a function of the form $n^{-\omega(1)}$. In the context of describing probability distributions, we write $x \in U_n$ to denote choosing x at random from U_n; we also use $x \in L$ to denote choosing an element x from the set L uniformly at random. For a probabilistic function f, we use $f(x;R)$ to denote the output of f on input x and internal coin tosses R, and we say that f is *public-coin* [L96] if it publishes its internal coin tosses as part of its output.

- A *point function* $I_x : \{0,1\}^n \to \{0,1\}$, where $|x| = n$, is defined by $I_x(y) = 1$ if $y = x$ and 0 otherwise. We would also use I_x to denote a circuit that hardwires the value x and computes I_x.

- Two distributions X and Y over $\{0,1\}^n$ are (s, ϵ)*-indistinguishable* if for all probabilistic circuits A of size s, $|\Pr[A(X) = 1] - \Pr[A(Y) = 1]| < \epsilon$.

- A function $f : \{0,1\}^n \to \{0,1\}^*$ is (s, ϵ)*-one-way* if f is efficiently computable and $\Pr_{x \in U_n}[A(f(x)) \in f^{-1}(f(x))] < \epsilon(n)$ for all circuits A of size s.

- A probabilistic function $h : \{0,1\}^n \to \{0,1\}^*$ is *statistically collision-resistant* if

$$\Pr_{R \in U_{r(n)}} \Big[\exists\, x \neq y \in \{0,1\}^n : h(x; R) = h(y; R)\Big] \leq 2^{-n}$$

2.2 Obfuscation

Definition 2.1. *[BGI+01, LPS04] A probabilistic polynomial-time algorithm $\mathcal{O}$ is an* obfuscator *for the* family of circuits $\mathcal{C} = \cup_n \mathcal{C}_n$ *(where $\mathcal{C}_n$ is the subset of circuits in $\mathcal{C}$ that take inputs of length n) if the following three conditions hold:*

- *(approximate functionality) There exists a negligible function α such that for all n, for all $C \in \mathcal{C}_n$, with probability $1 - \alpha(n)$ over the internal coin tosses of the obfuscator, $\mathcal{O}(C)$ describes a circuit that computes the same function as C.*

- *(polynomial slowdown) There is a polynomial p such that for every circuit $C \in \mathcal{C}$, $|\mathcal{O}(C)| \leq p(|C|)$.*

- *("virtual black-box" property) For any PPT A, there is a PPT S_A and a negligible function α such that for all circuits $C \in \mathcal{C}$,*

$$\Big|\Pr\left[A(\mathcal{O}(C)) = 1\right] - \Pr\left[S_A^C(1^{|C|}) = 1\right]\Big| \leq \alpha(|C|)$$

The obfuscators that we construct satisfy a weaker variant of the "virtual black-box" property (similar to the notion ϵ-knowledge in [DNS98]):

> *("virtual black-box" property with a weak simulator) For every family of polynomial-sized circuits $\{A_n\}$ and every function $\epsilon(n) = 1/n^{O(1)}$, there exists a family of probabilistic circuits $\{S_n\}$ of size* $\mathrm{poly}(n, 1/\epsilon)$ *such that for all sufficiently large n, for all circuits $C \in \mathcal{C}_n$:*
>
> $$\Big|\Pr\left[A_n(\mathcal{O}(C)) = 1\right] - \Pr\left[S_n^C(1^{|C|}) = 1\right]\Big| \leq \epsilon(n)$$

We stress that [BGI+01] also rules out obfuscation for general circuits under this definition. In addition, as pointed out in [BGI+01], this definition is equivalent to that which asks that for every predicate P (not necessarily efficiently computable), the probability that $A_n(\mathcal{O}(C)) = P(C)$ is at most the probability that $S_n^C(1^{|C|}) = P(C)$ plus ϵ. In the rest of this paper, we will use this notion of "virtual black-box", unless otherwise specified. In particular, our results in Sections 4 and 5 pertain to this weaker notion of obfuscation. For concrete parameters, we have the following definition:

Definition 2.2. *An obfuscator $\mathcal{O}$ for a set of circuits $\mathcal{C}_n$ is (K, s, ϵ)*-virtual black-box *if it satisfies the following: for any probabilistic circuit A of size s, there exists a probabilistic circuit S_A of size K such that for all circuits $C \in \mathcal{C}_n$,*

$$\Big|\Pr\big[A(\mathcal{O}(C)) = 1\big] - \Pr\big[S_A^C(1^{|C|}) = 1\big]\Big| < \epsilon$$

3 Constructing Obfuscators for Point Functions

A natural approach would be to replace the oracle hash function used in [LPS04] with a standard cryptographic primitive; we begin by explaining why two such primitives do not work. The first is a non-interactive perfectly (or statistically) binding and computationally hiding commitment scheme. The main problem here is that there is no efficient procedure to verify the secret value without revealing the randomness used in the commitment scheme. The same problem arises if we were to use a public-key encryption scheme secure against plain-text attacks. Another approach is to use the secret value as the seed to a pseudorandom generator and storing the output of the generator. This does not work because pseudorandomness is only guaranteed when the secret value underlying the point function is randomly chosen from the uniform distribution, whereas we require security for every possible point function in the obfuscation setting. We begin by presenting the assumption we use for our construction, along with the underlying hash function.

Assumption 3.1. *There exists an efficiently computable permutation $\{\pi_n : \{0,1\}^n \to \{0,1\}^n\}_{n\in\mathbb{N}}$ such that for every polynomial $p(n)$ and every family of circuits $\{A_n\}$ of size $p(n)$, there exists a polynomial q such that:* $\Pr_{x\in U_n}[A_n(\pi(x)) = x] \leq q(n)/2^n$.

Construction 3.2. *Let $\pi : \{0,1\}^n \to \{0,1\}^n$ be a permutation. We define a (public-coin) probabilistic function $h : \{0,1\}^n \times \{0,1\}^{3n^2} \to \{0,1\}^{3n^2+3n}$ as follows:*

$$h(x;\tau_1,\dots,\tau_{3n}) = (\tau_1,\dots,\tau_{3n},\langle x,\tau_1\rangle,\langle\pi(x),\tau_2\rangle,\dots,\langle\pi^{3n-1}(x),\tau_{3n}\rangle)$$

Proposition 3.3. *Suppose there exists a $\big(\mathrm{poly}(n,1/\epsilon)s, \frac{\epsilon K}{16n}\cdot\frac{1}{2^n})\big)$-one-way permutation, then there exists a public-coin obfuscator for the family of point function $\{I_x\}_{x\in\{0,1\}^n}$ which is $(K\,\mathrm{poly}(n), s, \epsilon)$-virtual black-box.*

Proof. Let π be a $\big(\mathrm{poly}(n,1/\epsilon)s, \frac{\epsilon K}{16n}\cdot\frac{1}{2^n})\big)$-one-way permutation, and h be the hash function from Construction 3.2 based on π. Then, define $\mathcal{O}(I_x;R)$ to be the circuit that stores the value $h(x;R)$ (which contains R as a substring), and on input y, checks whether $h(y;R) = h(x;R)$. If so, output 1, and 0 otherwise. Clearly, $\mathcal{O}$ satisfies polynomial slow-down and public-coin.

Approximate functionality: Fix $x \neq y \in \{0,1\}^n$. Then,

$$\begin{aligned}&\Pr_{\tau_1,\dots,\tau_{3n}}[h(x;\tau_1,\dots,\tau_{3n}) = h(y;\tau_1,\dots,\tau_{3n})]\\ =\ &\Pr_{\tau_1,\dots,\tau_{3n}}[\forall\, j=1,2,\dots,3n : \langle\pi^{j-1}(x),\tau_j\rangle = \langle\pi^{j-1}(y),\tau_j\rangle] = \frac{1}{2^{3n}}\end{aligned}$$

because π is a permutation, so $\pi^{j-1}(x) \neq \pi^{j-1}(y)$ for all $j = 1,2,\dots,3n$. Taking a union bound over all $x,y \in \{0,1\}^n$ shows that h is statistically collision-resistant, from which approximate functionality follows.

"Virtual black-box": Let A be a probabilistic circuit of size s, and define

$$L = \{x \in \{0,1\}^n : \big|\Pr_R[A(h(x;R)) = 1] - \Pr[A(U_{3n^2+3n}) = 1]\big| \geq \epsilon\}$$

Claim 3.4. $|L| \leq K-1$.

Note that the "virtual black-box" property follows readily from Claim 3.4 as follows: consider the simulator S_A (a probabilistic circuit of of size $K\,\mathrm{poly}(n)$) that has L hardwired into it. On input 1^n and oracle access to I_x, S_A queries I_x on each element of L. If $x \in L$, S_A picks R' at random and outputs $A(h(x;R'))$. Otherwise (that is, $x \notin L$), S_A picks a random string $R'' \in \{0,1\}^{\ell(n)}$ and outputs $A(R'')$.

Now, suppose Claim 3.4 does not hold, namely that $|L| \geq K$. We show how to construct from A a circuit of size $\mathrm{poly}(n,1/\epsilon)s$ that inverts π on $\epsilon K/16n$ inputs, which contradicts our assumption about π. We may assume (by replacing A with its negation if necessary) that there exists a subset L' of L of size at least $K/2$

such that for all $x \in L'$:

$$\Pr_R[A(h(x;R)) = 1] - \Pr[A(U_{3n^2+3n}) = 1] \geq \epsilon$$

Averaging over $x \in L'$, we have

$$\Pr_{x \in L',R}[A(h(x;R)) = 1] - \Pr[A(U_{3n^2+3n}) = 1] \geq \epsilon$$

Consider the following $3n+1$ hybrid distributions: where $x \in L', \tau_1, \ldots, \tau_{3n} \in U_n$ and $b_1, \ldots, b_{3n} \in \{0,1\}$:

$$\begin{array}{rcl} D_0 & = & \{\pi^{3n}(x), \tau_1, \ldots, \tau_{3n}, \langle \pi^{3n-1}(x), \tau_{3n}\rangle, \ldots, \langle x, \tau_1\rangle\} \\ D_1 & = & \{\pi^{3n}(x), \tau_1, \ldots, \tau_{3n}, \langle \pi^{3n-1}(x), \tau_{3n}\rangle, \ldots, b_1\} \\ \vdots & & \vdots \quad\quad \cdots \quad \vdots \quad \cdots \quad \vdots \\ D_{3n} & = & \{\pi^{3n}(x), \tau_1, \ldots, \tau_{3n}, b_{3n}, \ldots, b_1\} \end{array}$$

A standard hybrid argument yields some $j \in \{1, 2, \ldots, 3n-1\}$ and a next-bit predictor that on input $\pi^{3n}(x), \tau_1, \ldots, \tau_{3n}$ and $\langle \pi^{3n-1}(x), \tau_{3n}\rangle, \ldots, \langle \pi^j(x), \tau_{j+1}\rangle$, guesses $\langle \pi^{j-1}(x), \tau_j\rangle$ with probability $1/2 + \epsilon/3n$, where the probability is taken over $x \in L'$, $\tau_1, \ldots, \tau_{3n}$ and the coin tosses of P. We can then fix $\tau_{j'}$, for all $j' \neq j$ as well as the coin tosses of the predictor to obtain a deterministic predictor P' of size $O(s)$ satisfying:

$$\Pr_{x \in L', \tau_j}[P'(\pi^j(x), \tau_j) = \langle \pi^{j-1}(x), \tau_j\rangle] \geq 1/2 + \epsilon/3n$$

since P' can on input $\pi^j(x), \tau_j$ compute $\pi^{j+1}(x), \ldots, \pi^{3n}(x)$. By an averaging argument, it follows that for a $\epsilon/6n$ fraction of x in L' (and call such x "good"):

$$\Pr_{\tau_j}[P'(\pi^j(x), \tau_j) = \langle \pi^{j-1}(x), \tau_j\rangle] \geq 1/2 + \epsilon/6n$$

We can then apply the list-decoding algorithm for Hadamard code from [GL89] to recover "good" x's. Now, we have a probabilistic circuit of size $\text{poly}(n, 1/\epsilon)s$ that recovers each "good" x with probability 3/4. We can then hardwire the randomness into the circuit to obtain one of size $\text{poly}(n, 1/\epsilon)s$ that inverts π on a $\epsilon/16n$ fraction of the values in $\pi^j(L')$, which contradicts the $\big(\text{poly}(n, 1/\epsilon)s, \frac{\epsilon K}{16n} \cdot \frac{1}{2^n}\big)$-one-way property of π. □

The following follows readily from Proposition 3.3:

Theorem 3.5. *Under Assumption 3.1, Construction 3.2 applied to π yields an efficient public-coin obfuscator with a weak simulator for the family of point functions.*

We may in fact relax the assumption in Theorem 3.5 to requiring a strongly one-way permutation ensemble, instead of a specific permutation. Roughly speaking, a strongly one-way permutation ensemble is a family of permutations indexed by strings, where most permutations are strongly one-way (in the sense of Assumption 3.1) with respect to non-uniform adversaries that get its index as an additional input. Unlike the setting of standard one-way permutations, Assumption 3.1 is not an immediate consequence of Assumption 3.6.

Assumption 3.6. *There exists a collection of efficiently computable permutation ensemble $\mathcal{F} = \{\mathcal{F}_n\}_{n \in \mathbb{N}}$ such that for every polynomial $p(n)$ and every family of circuits $\{A_n\}$ of size $p(n)$, there exists a polynomial q and a negligible function α such that for all sufficiently large n, with probability $1 - \alpha(n)$ over $\pi \in \mathcal{F}_n$, $\Pr_{x \in U_n}[A_n(\pi, \pi(x)) = x] \leq q(n)/2^n$.*

Theorem 3.7. *Under Assumption 3.1, there exists an efficient public-coin obfuscator with a weak simulator for the family of point functions with multi-bit output, namely $I_{x,\beta} : \{0,1\}^n \to \{0,1\}^{3n}$, where $|x| = n$ and $|\beta| = m = \mathrm{poly}(n)$ is defined by $I_x(y) = \beta$ if $y = x$ and 0^m otherwise.*

Proof. (sketch) We use the obfuscator construction from [LPS04], but replacing the oracle with a generalization of the hash function from Construction 3.2, namely: $h_{n,m} : \{0,1\}^n \times \{0,1\}^{3n^2+n} \to \{0,1\}^{3n^2+6n+m}$ as follows:

$$h(x;\tau_1,\ldots,\tau_{3n+1}) = (\tau_1,\ldots,\tau_{3n},\langle x,\tau_1\rangle,\ldots,\langle \pi^{3n-1}(x),\tau_{3n}\rangle,\langle \pi^{3n}(x),\tau_{3n+1}\rangle,\ldots,\langle \pi^{3n+m-1},\tau_{3n+1}\rangle)$$

Specifically, $\mathcal{O}(I_{x,\beta};R)$ is the circuit that stores the value $z = h(x;R) \oplus (0^{3n^2+6n} \circ \beta)$, and on input y, checks whether the first $3n^2 + 6n$ bits of $h(y;R)$ and z agree. If so, output the last m bits of $z \oplus h(y;R)$; otherwise, output 0^m. $\square$

4 On the One-Way Permutation Assumption

In the previous section, we provide a construction of obfuscators under a very strong assumption. Here, we show that the assumption is plausible, and can be realized with random permutation oracle, and that a similar assumption is also necessary.

4.1 Very strongly one-way permutation from a random permutation

Gennaro and Trevisan proved that a random permutation is one-way against nonuniform adversaries [GT00]. We extend their argument to show that a random permutation is also very strongly one-way in the sense of Assumption 3.1 (slightly stronger in fact, as $q(n)$ here is bounded by a fixed polynomial in $p(n)$). Clearly, our analysis is tight up to polynomial factors. To simplify the exposition, we include the input gates in determining the size of a circuit, so that any circuit has size at least that of its input. We use Π_n to denote the set of all permutations over $\{0,1\}^n$.

Theorem 4.1. *For all sufficiently large n, a random $\pi \in \Pi_n$ is $(s, s^4/2^n)$-one-way for all $n \le s \le 2^{n/5}$ with probability at least $1 - 2^{-n^3}$.*

The approach is the same as that in [GT00]: showing that any permutation π for which there is a circuit A such that A inverts π on "many" inputs has a "short" description (given A). Hence there cannot be too many such permutations. Refer to Appendix A for the proofs of the theorem and the following claim.

Claim 4.2. *Let A be a circuit that makes (at most) $s-1$ queries to a permutation $\pi : \{0,1\}^n \to \{0,1\}^n$, and for which $\Pr_y[A^\pi(y) = \pi^{-1}(y)] \ge s^4/2^n$. Then, π can be described using at most*

$$\log \binom{2^n}{s^3} + s^3 \log s + \log[2^n]_{s^4-s^3} + \log(2^n - s^4)!$$

bits given A (where $[n]_k$ denotes the quantity $n(n-1)\cdots(n-k+1)$).

Remark(s): The main quantitative difference between the analysis in [GT00] and our analysis is in the length of the description in Claim 4.2: the former would yield $2\log\binom{2^n}{s^3} + \log(2^n - s^3)! \approx 2^n n + s^3 n - \theta(s^3 \log n)$ bits whereas our analysis yields approximately $2^n n - \Omega(s^3 \log s)$ bits. In terms of "work" (discussed in Section 1.3.2), the analysis in [GT00] cannot provide a lower bound better than $2^{n/2}$ whereas our analysis yields a lower bound of $2^{n-\theta(\log n)}$.

Consider a relativization of Definition 2.1 wherein all parties (the obfuscation algorithm, the adversary and the simulator) have access to a random permutation oracle, and the simulator must simulate the adversary's (single-bit) output with respect to the same oracle. We refer to this as the non-programmable random permutation oracle model [N02]:

("virtual black-box" property with a non-programmable random permutation oracle) With probability $1 - \mathrm{neg}(n)$ *over* $\pi \in \Pi_n$*, for every family of probabilistic polynomial-sized circuits* $\{A_n\}$ *and every function* $\epsilon(n) = 1/n^{O(1)}$*, there exists a family of probabilistic circuits* $\{S_n\}$ *of size* $\mathrm{poly}(n, 1/\epsilon)$ *such that for all sufficiently large* n*, for all circuits* $C \in \mathcal{C}_n$*:*

$$\left| \Pr\left[A_n^{\pi}(\mathcal{O}^{\pi}(C)) = 1\right] - \Pr\left[S_n^{C,\pi}(1^{|C|}) = 1\right] \right| \leq \epsilon(n)$$

Corollary 4.3. *There exists an efficient public-coin obfuscator for point functions in the non-programmable random permutation oracle model.*

Our obfuscator here differs from the obfuscator for point functions in [LPS04] in that the simulator for latter gets to choose the coin tosses of the random oracle and therefore simulates the view of the adversary with respect to a possibly different (random) oracle. However, the [LPS04] construction does not have any of the limitations described in Section 1.3.3.

4.2 The necessity of strong computational assumptions

In this section, we explore the necessity of Assumptions 3.1 and 3.6.

Proposition 4.4. *Suppose that public-coin obfuscators exist. Then, there exists an efficiently computable function ensemble* $\mathcal{F} = \{\mathcal{F}_n\}$ *satisfying the following properties:*

- *(mostly injective) With probability* $1 - \mathrm{neg}(n)$ *over* $f \in \mathcal{F}_n$*,* f *is injective.*
- *(somewhat strongly one-way) For all polynomials* $p(n)$*, for all family of circuits* $\{A_n\}$ *of size* $p(n)$*, there exists a constant* $c > 0$ *such that for all sufficiently large* n*,* $|Q_n| < n^c$*, where* Q_n *is the set:*

$$\{x \in \{0,1\}^n \mid \Pr_{f \in \mathcal{F}_n}[A_n(f, f(x)) \in f^{-1}(f(x))] \geq 1/p(n)\}$$

Note that the main qualitative differences between the function ensemble herein and that stipulated in Assumption 3.6 are mostly injective functions vs permutations and a reversal of quantifiers between x and f in the specification of the one-wayness property.

Proof. Let $\mathcal{O}$ be a public-coin obfuscator for point functions using $r(\cdot)$ random bits. We define

$$\mathcal{F}_n = \{f \mid (f(x); f) = \mathcal{O}(x; R), R \in \{0,1\}^{r(n)}\}$$

that is, every $R \in \{0,1\}^{r(n)}$ is the index[3] of a function $f \in \mathcal{F}_n$, and $f(x)$ is the string $\mathcal{O}(x; R)$, excluding R (which is part of $\mathcal{O}(x; R)$ since $\mathcal{O}$ is public-coin). It is clear that functionality for $\mathcal{O}$ implies $\mathcal{F}$ is mostly injective.

Next, for each n, let x_n denote the lexicographic mid-point of Q_n, and let P_n denote the predicate "$\succ x_n$". Consider the family of (probabilistic) circuits $\{B_n\}$, where B_n is the circuit that on input (f, y)

i. Compute $x' = A_n(f, y)$.

ii. If $f(x) = f(x')$, output $P_n(x')$; otherwise, output a random bit.

It is easy to see that

$$\Pr_{f \in \mathcal{F}_n}[B_n(f, f(x)) = P_n(x)] \geq \tfrac{1}{2} + \tfrac{1}{2p(n)} - \mathrm{neg}(n)$$

[3]To avoid introducing additional notation, we use f to denote both the function and its index.

where the neg(n) term captures the fraction of f which is not injective. Let $\{S_n\}$ be the weak simulator for $\{B_n\}$ with distinguishing probability $1/4p(n)$ given by the "virtual black-box" property of $\mathcal{O}$. Therefore, for all sufficiently large n, for all $x \in Q_n$, we have:

$$\left| \Pr_{f \in \mathcal{F}_n} \left[B_n(f, f(x)) = P_n(x) \right] - \Pr \left[S_n^{I_x}(1^n) = P_n(x) \right] \right| \leq 1/4p(n)$$

On the other hand, since S_n is polynomial-size, we have:

$$\Pr_{x \in Q_n} [S_n^{I_x}(1^n) = P_n(x)] \leq \tfrac{1}{2} + \tfrac{\mathrm{poly}(n)}{|Q_n|}$$

Combining the 3 inequalities yields polynomial bound on $|Q_n|$. □

Next, we consider a complexity assumption about circuit satisfiability that is necessary for the existence of obfuscators for point functions (not necessarily public-coin ones); it follows that some sort of exponential lower bound is indeed necessary to obtain any kind of positive results for obfuscating point functions. We defer the proof to Appendix A.

Proposition 4.5. *[Tre04] If there exists a nontrivial (i.e., a probabilistic $2^{o(\#\mathsf{variables})} \cdot \mathrm{poly}(\mathsf{circuitsize})$-time) algorithm for the* CSAT *problem, then obfuscating point functions is impossible.*

It is generally believed that there is no nontrivial algorithm for CSAT. It would be interesting to find out if we could rule out obfuscating point functions assuming just the existence of a probabilistic $2^{\#\mathsf{variables} - \omega(\log \#\mathsf{variables})} \cdot \mathrm{poly}(\mathsf{circuitsize})$-time algorithm for CSAT.

5 Impossibility Results for Obfuscating Point Functions

Here, we outline two impossibility results relating to uniformly black-box simulators and simulators for general adversaries. Refer to Appendix B for the (omitted) proofs.

5.1 Ruling out obfuscators with uniformly black-box simulators

Recall that in the proof that indistinguishability implies semantic security for public-key encryptions schemes, the simulator picks a random encryption c of a random bit, and runs the adversary on c. Note that the simulator is uniform and uses only black-box access to the adversary. This is often the case for proving equivalence between a simulator-based and a distinguisher-based characterizations of cryptographic notions. Here, we rule out obfuscators with such uniformly black-box simulators for point functions.

> *("virtual black-box" property with a uniformly black-box simulator) For every function $\epsilon(n) = 1/n^{O(1)}$ and every polynomial $p(n)$, there exists an oracle PPT S such that for every family of circuits $\{A_n\}$ of size $p(n)$, we have: for all sufficiently large n, for all circuits $C \in \mathcal{C}_n$*
>
> $$\left| \Pr\left[A(\mathcal{O}(C)) = 1 \right] - \Pr\left[S^{A,C}(1^{|C|}) = 1 \right] \right| \leq \alpha(|C|)$$

The motivation for this result comes from the following simple observation: Fix a uniformly black-box simulator S of size s, and consider an adversary A_L that has hardwired into it a random subset L of $\{0,1\}^n$ of size s^2. On input an obfuscated circuit C of a point function, A outputs 1 if $C(x) = 1$ for any $x \in L$ and 0 otherwise. Intuitively, S given oracle access to A_L cannot learn more than s elements of L by simply evaluating A_L on obfuscations of different point functions. This naive argument fails precisely because S may evaluate A_L on circuits that do not correspond to valid obfuscations of point functions. In fact, the queries S may correspond to arbitrary subset membership queries, and for the case $|L| = 1$, it is plausible that S learns the single element of L using n non-adaptive queries, one to learn each bit of the n-bit string.

Theorem 5.1. *Uniformly black-box obfuscators for point functions do not exist.*

5.2 Ruling out obfuscators for general adversaries

As pointed out in [BGI+01], we may want to consider a stronger notion of "virtual black-box", where we do not restrict the nature of what the adversary is trying to compute. Informally, we require that the simulator, given just oracle access to a circuit C, produce an output distribution that is computationally indistinguishable from what the adversary computes when given $\mathcal{O}(C)$. In this setting, it suffices to consider adversary that computes the identity function, that is, output $\mathcal{O}(C)$.

> *("virtual black-box" property with a general adversary) For every polynomial $s(n)$ and every function $\epsilon(n) = 1/n^{O(1)}$, there exists a (possibly uniform) family of probabilistic circuits $\{S_n\}$ of size $\mathrm{poly}(s, n, 1/\epsilon)$ such that for all sufficiently large n, for all circuits $C \in \mathcal{C}_n$: $\{\mathcal{O}(C)\}$ and $\{S_n^C(1^{|C|})\}$ are (s, ϵ)-indistinguishable.*

Note that in this definition, we allow the size of the simulator S to depend on the size of the distinguisher. The difference between this definition and that with a weak simulator presented in Section 2.2 is the order of quantifiers (so that the simulator S_n may or may not depend on the distinguisher A_n): informally, upon fixing n, s, ϵ,

- (weak simulator) for all A_n of size s, there exists S_n of size $poly(s, n, 1/\epsilon)$ such that for all $C \in \mathcal{C}_n$: $|\Pr[A_n(\mathcal{O}(C)) = 1] - \Pr[S_n^C(1^{|C|}) = 1]| \leq \epsilon$.
- (general adversary) there exists S_n of size $poly(s, n, 1/\epsilon)$ such that for all A_n of size s, for all $C \in \mathcal{C}_n$: $|\Pr[A_n(\mathcal{O}(C)) = 1] - \Pr[A_n(S_n^C(1^{|C|})) = 1]| \leq \epsilon$.

Proposition 5.2. *A family of circuits $\mathcal{C} = \cup_n \mathcal{C}_n$ is efficiently obfuscatable against general adversaries iff $\mathcal{C}$ is efficiently and exactly learnable using membership queries (and possibly non-uniformity).*

The learning algorithm for an efficiently obfuscatable against general adversaries merely outputs a circuit computing the majority of n^2 independent copies of the simulator. Note that the learning algorithm will be non-uniform if the simulator is non-uniform; however, the non-uniformity only depends on the input length n. Moreover, many lower bounds for learnability are lower bounds on query complexity and may therefore be used to rule out non-uniform learning algorithms and thus obfuscation against general adversaries. Obfuscating exactly learnable functions against general adversaries is straight-forward and was observed in [LPS04]: the obfuscator simply takes the input circuit C and outputs the circuit produced by the learning algorithm given oracle access to C; the simulator does essentially the same thing and thus its size will not in fact depend on the size of the distinguisher, even though we allow for that in the definition. In fact, the result relativizes in the sense that if we allow the obfuscation algorithm, the obfuscated circuit, the simulator and the distinguisher access to some oracle, we obtain an efficient and exact learning algorithm with respect to the same oracle.

The next result follows from the fact that point functions are not exactly learnable (since a uniformly chosen point function is statistically indistinguishable from the all-zeroes function given polynomially many membership queries).

Theorem 5.3. *Obfuscating point functions against general adversaries is impossible. Furthermore, the proof relativizes and hence the result extends to the non-programmable random oracle model.*

6 Conclusion

We presented a positive result for program obfuscation in this paper under a very strong but nonetheless plausible assumption, along with some evidence that the assumption we used and the limitations of our construction are in fact inherent. We point out several interesting directions for future work:

- Obfuscating point functions in the original [BGI+01] definition (with negligible distinguishing probability): is that possible?

- Obfuscating multi-point functions (or self-composability [LPS04]): a natural extension of function points are multi-point functions $\{I_{x_1,\ldots,x_k} : \{0,1\}^n \to \{0,1\}^k\}$, for $(x_1, \ldots, x_k) \in (\{0,1\}^n)^k$ where $I_{x_1,\ldots,x_k}(y) = (I_{x_1}(y), \ldots, I_{x_k}(y))$. Simply hard-writing $h(x_1), \ldots, h(x_k)$ into the obfuscated circuit and trying to establish that there is only a small subset $L \subseteq (\{0,1\}^n)^k$ such that $\{h(x_1), \ldots, h(x_k)\}_{(x_1,\ldots,x_k)\in L}$ is computationally indistinguishable from $(U_\ell)^k$ runs into problems.

- Obfuscating AC_0: obfuscating TC_0 is impossible [BGI+01], whereas we can obfuscate NC_0 against general adversaries (since NC_0 is exactly learnable). Note that the techniques of [BGI+01] do not extend to AC_0 since pseudorandom functions do not exist in AC_0 [LMN93]. In addition, AC_0 can implement point functions, so the results of Section 4 do apply in this setting.

- Explore the possibility of replacing constructions in the random oracle model with a real-world construction under Assumption 3.6.

7 Acknowledgements

I am especially grateful to David Molnar, Luca Trevisan, David Wagner and Ke Yang for their encouragement and for illuminating and inspiring discussions, especially in the early stages of this research, and to Amit Sahai, Salil Vadhan and an anonymous referee for their insightful and constructive feedback on this work. In addition, I would like to thank Andrej Bogdanov, Dan Boneh, Ran Canetti, Iordanis Kerenidis, Daniele Micciancio, Jason Waddle, David Woodruff and the anonymous referees for helpful pointers, suggestions and discussions.

References

[B01] Boaz Barak, "How to Go Beyond the Black-box Simulation Barrier", *Proceedings of FOCS '01.*

[BGI+01] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil Vadhan, Ke Yang, "On the (Im)possibility of Obfuscating Programs" *Proceedings of Crypto '01.*

[BM84] Manuel Blum, Silvio Micali, "How to Generate Cryptographically Strong Sequences of Pseudo-Random Bits", *SIAM Journal of Computing*, Vol 13, pages 850-864, 1984.

[C97] Ran Canetti, "Towards realizing random oracles: Hash functions that hide all partial information", *Proceedings of Crypto '97.*

[CGH98] Ran Canetti, Oded Goldreich, and Shai Halevi, "The Random Oracle Methodology, Revisited", *Proceedings of STOC 1998.*

[CMR98] Ran Canetti, Daniele Micciancio and Omer Reingold, "Perfectly One-Way Probabilistic Hash Functions", *Proceedings of STOC 1998.*

[DNS98] Cynthia Dwork, Moni Naor and Amit Sahai, "Concurrent Zero-Knowledge", *Proceedings of STOC 1998*

[F99] Marc Fischlin, "Pseudorandom Function Tribe Ensembles Based on One-Way Permutations: Improvements and Applications", *Proceedings of Eurocrypt 1999.*

[GGM84] Oded Goldreich, Shafi Goldwasser and Silvio Micali, "How to Construct Random Functions", *Journal of the ACM*, 33(4), 1984, 792-807.

[GHR99] Rosario Gennaro, Shai Halevi and Tal Rabin, "Secure Hash-and-Sign Signatures without the Random Oracle", *Proceedings of Eurocrypt 1999.*

[GL89] Oded Goldreich, Leonid Levin, “Hard-Core Predicates for Any One-Way Function”, *Proceedings of STOC 1989.*

[GT00] Rosario Gennaro and Luca Trevisan, “Lower Bounds on Efficiency of Generic Cryptographic Constructions”, *Proceedings of FOCS 2000.*

[L96] Michael Luby, “Pseudorandomness and Cryptographic Applications”, Princeton University Press, 1996.

[LPS04] Benjamin Lynn, Manoj Prabhakaran, Amit Sahai, “Positive Results and Techniques for Obfuscation”, *Proceedings of Eurocrypt 2004.*

[LMN93] Nathan Linial, Yishay Mansour, Noam Nissan, “Constant Depth Circuits, Fourier Transform, and Learnability”, *Journal of the ACM*, 40(3), 1993, 607-620.

[N02] Jesper Buss Nielsen, “Separating Random Oracle Proofs from Complexity Theoretic Proofs: The Non-committing Encryption Case”, *Proceedings of Crypto 2002.*

[P03] Rafael Pass, “On Deniabililty in the Common Reference String and Random Oracle Models”, *Proceedings of Crypto 2003.*

[Tre04] Luca Trevisan, private communication, 2004.

[WG00] David Wagner, Ian Goldberg, “Proofs of security for the Unix password hashing algorithm”, *Proceedings of Asiacrypt 2000.*

A Omitted proofs from Section 4

A.1 Proof of Claim 4.2

Proof. Let $N = 2^n$. WLOG, assume that A makes distinct queries to π, and always queries π on the value it is going to output. Consider the set I of s^4 points on which A inverts π, after making s queries to π. We define a set Y and another set W via the following process: initially W is $\{0,1\}^n$ whereas Y is empty, and all the elements of I are candidates for inclusion in Y. Take the lexicographically first element y from I, and place it in Y. Next, simulate the computation of $A^\pi(y)$ and let $x_1, \dots, x_{q(y)}$ be the distinct queries made by A to π, with $q(y) \le s$ and $\pi(x_{q(y)}) = y$. Let $y_1, \dots, y_{q(y)}$ be the corresponding answers (that is, $y_i = \pi(x_i)$). We add the answers $y_1, \dots, y_{q(y)-1}$ (in order) and the number $q(y)$ to our description of π, and remove the strings $x_1, \dots, x_{q(y)}$ from I. In addition, we remove the strings $x_1, \dots, x_{q(y)}$ from W. At any step of the construction, one element is added to Y and at most $s-1$ elements is removed from I. Since I initially contains s^4 elements, in the end, we have $|Y| \ge s^3$. In fact, we will stop the process when Y reaches exactly s^3 elements.

We claim that given the set Y, the descriptions $y_1, \dots, y_{q(y)-1}$ and the number $q(y)$ for each $y \in Y$, the values of π on W and the circuit A, it is possible to compute π everywhere. The values of π^{-1} on Y can be reconstructed sequentially for all $y \in Y$, taken in lexicographic order, as follows: Simulate the computation of $A^\pi(y)$. Our description allows us to answer the first $q(y)-1$ queries, and the $q(y)$th query is exactly $\pi^{-1}(y)$. Note that while reconstructing π^{-1} on Y, we have also reconstructed the set W and π on all of $\{0,1\}^n - W$.

Let $Q = \sum_{y \in Y} q(y)$, so $Q \le s^3(s-1)$ and $|W| \le 2^n - Q - s^3$. Describing Y requires $\log \binom{N}{s^3}$ bits. The descriptions $y_1, \dots, y_{q(y)-1}$ and the number $q(y)$ for all $y \in Y$ require at most $s^3 \log s + \sum_{i=1}^{Q} \log(N - i + 1) = s^3 \log s + \log[N]_Q$. Once we have constructed the set W, π on $\{0,1\}^n - W$ can be described using $\log(N - |W|)!$ bits. The total description requires at most $\log \binom{N}{s^3} + s^3 \log s + \log[N]_Q + \log(N - |W|)! \le \log \binom{N}{s^3} + s^3 \log s + \log[2^n]_{s^4 - s^3} + \log(N - s^4)!$ bits. □

A.2 Proof of Theorem 4.1

Proof. Fix a circuit A of size s, where $s \le 2^{(n-1)/4}$. It follows from the above claim that the fraction of permutations $\pi \in \Pi_n$ such that $\Pr_x[A^\pi(\pi(x)) = x] \ge s^4/2^n$ is at most

$$\binom{N}{s^3} \cdot s^{s^3} \cdot [N]_{s^4 - s^3} \cdot (N - s^4)! \cdot \frac{1}{N!} = \frac{s^{s^3}}{s^3!} \cdot \frac{[N]_{s^4 - s^3}}{[N - s^3]_{s^4 - s^3}}$$

which is upper bounded by

$$\left(\frac{es}{s^3}\right)^{s^3} \cdot \exp\left(\frac{s^3}{N - s^3} + \dots + \frac{s^3}{N - s^4}\right) < \left(\frac{e^2}{s^2}\right)^{s^3}$$

Since there are at most $2^{sn \log s}$ circuits of size s, a union bound shows that the probability over a random choice of $\pi \in \Pi_n$ that there exists a circuit A of size s, for some s satisfying $n \le s \le 2^{n/5}$, where $\Pr_x[A^\pi(\pi(x)) = x] \ge s^4/2^n$ is at most

$$\sum_{s=n}^{2^{n/5}} s^{ns} \left(\frac{e^2}{s^2}\right)^{s^3} < 2^{n/5} \left(\frac{e^2}{n}\right)^{n^3} < 2^{-n^3} \quad \square$$

A.3 Proof of Prop 4.5

Proof. Let $t(\#\mathsf{variables}, \mathsf{circuitsize})$ denote the running time of the CSAT algorithm, and suppose on the contrary that there exists an obfuscator $\mathcal{O}$ for point functions. Let $\ell(n) = \omega(\log n)$ such that $t(\ell(n), \mathrm{poly}(n)) =$

poly(n), and let L denote the set of strings $\{0,1\}^n$ whose last $n-\ell(n)$ bits are 0's (so that $|L| = n^{\omega(1)}$). Let A denote the polynomial-time algorithm that uses the CSAT algorithm to decide on input a circuit C of size poly(n) on $\ell(n)$ variables, whether there is a satisfying assignment for C whose first bit is 1. Then,

$$\Pr_{x \in L}[A(\mathcal{O}(I_x)) = x_1] \geq 1 - \text{neg}(n)$$

(where x_1 denotes the first bit of x) whereas for any probabilistic polynomial-time algorithm S, possibly non-uniform,

$$\Pr_{x \in L}[S^{I_x}(1^n) = x_1] \leq 1/2 + \text{neg}(n)$$

This yields the required contradiction to the "virtual black-box" property. $\square$

B Omitted proofs from Section 5

B.1 Proof of Theorem 5.1

Proof. Suppose on the contrary that there exists an obfuscator $\mathcal{O}$ for point functions with a uniformly black-box obfuscator S of size $s = poly(n)$ for a fixed value of ϵ, say 1/2. For any $L \subseteq \{0,1\}^n$, A_L to be the adversary that has L hardwired into it, and on input a circuit C, and outputs $\bigvee_{z \in L} C(z)$. Now, fix L to be a subset containing exactly s^2 elements (and since the result holds for each such set L, we can pick a random set L if we want a PPT adversary). We want to show that:

$$\mathrm{E}_{y \in L}\left[|\Pr[S^{A_L, I_y} = 1] - \Pr[S^{A_{L \setminus y}, I_y} = 1]|\right] \leq \frac{1}{s-1} \tag{B.1}$$

By Yao's minimax theorem, it suffices to prove this for deterministic S.[4] Pick a y at random in L and fix y. Consider the first oracle query q_1 that S makes, and we have two cases:

1. q_1 is a circuit C to the adversary oracle. Let $\Gamma_C = \{z \in \{0,1\}^n \mid C(z) = 1\}$. If $\Gamma_C \cap L = \emptyset$, then the answer from both adversaries A_L and $A_{L \setminus y}$ would be 0, independent of y. Similarly, if $|\Gamma_C \cap L| \geq 2$, then the answer from both adversaries would be 1, also independent of y. For the case $|\Gamma_C \cap L| = 1$ (and in fact, we may assume WLOG that all simulator queries to the adversary oracle are of this form), then the answer from both adversaries would be 1, unless $\Gamma_C \cap L = \{y\}$.

2. q_1 is a string to the point function oracle. Then, if the query is not y, then the answer is always 0.

We define query to be "useful" if it is either the query y to the point function oracle, or a circuit C to the adversary oracle satisfying $\Gamma_C \cap L = \{y\}$. Observe that for a random y, the first query is "useful" with probability $1/s^2$. For each $i \geq 1$, it is easy to see that conditioned upon the first i queries of S being not useful, y is equally likely to be any string in a set of size at least $s^2 - i$, so the probability that the $(i+1)$-th query is useful is at most $1/(s^2 - i)$. This is true even if the queries are adaptive. Then, (B.1) follows from:

$$\Pr_{y \in L}\left[\text{at least one of } s \text{ queries is useful}\right] \leq \sum_{i=1}^{s} \frac{1}{s^2 - i + 1} < \frac{1}{s-1}$$

On the other hand, $\Pr[A_L(\mathcal{O}(I_y)) = 1] \geq 1 - \text{neg}(s)$ and $\Pr[A_{L \setminus y}(\mathcal{O}(I_y)) = 1] \leq \text{neg}(s)$ which together with (B.1) contradict the fact that $\mathcal{O}$ is a uniformly black-box obfuscator. $\square$

[4]Once we fix the random coin tosses of S, both $\Pr[S^{A_L, I_y} = 1]$ and $\Pr[S^{A_L, I_y} = 1]$ are $0, 1$ values.

B.2 Proof of Prop 5.2

Proof. (sketch) For completeness, we outline the analysis of the learning algorithm for a family of circuits obfuscatable against general adversaries. Let $s(n)$ denote an upper bound of the size of the circuits produced by the obfuscator, and S the simulator for distinguishers of size s and $\epsilon = 1/4$. For each $y \in \{0,1\}^n$, consider the non-uniform distinguisher eval_y of size s that evaluates its input (which is a circuit) on y. Then, we have: for all $C \in \mathcal{C}_n$ and for all $y \in \{0,1\}^n$:

$$\Pr[\mathsf{eval}_y(S^C(1^n)) = (\mathcal{O}(C))(y) \text{ and } (\mathcal{O}(C))(y) = C(y)] \geq 3/4 - \mathrm{neg}(n)$$

Taking the majority of n^2 independent evaluations of S allows us to take union over all $y \in \{0,1\}^n$ so that for all $C \in \mathcal{C}_n$, with overwhelming probability, the learning algorithm produces a circuit that agrees with C everywhere on $\{0,1\}^n$.

□