On the Impossibility of Obfuscation with Auxiliary Input

Shafi Goldwasser[*]
The Weizmann Institute
shafi@theory.lcs.mit.edu

Yael Tauman Kalai[†]
MIT
yael@theory.lcs.mit.edu

Abstract

Barak et al. formalized the notion of obfuscation, and showed that there exist (contrived) classes of functions that cannot be obfuscated. In contrast, Canetti and Wee showed how to obfuscate point functions, under various complexity assumptions. Thus, it would seem possible that most programs of interest can be obfuscated even though in principle general purpose obfuscators do not exist.

We show that this is unlikely to be the case. In particular, we consider the notion of obfuscation w.r.t. auxiliary input, *which corresponds to the setting where the adversary, which is given the obfuscated circuit, may have some additional a priori information. This is essentially the case of interest in any usage of obfuscation we can imagine. We prove that there exist many* natural *classes of functions that cannot be obfuscated w.r.t. auxiliary input, both when the auxiliary input is dependent of the function being obfuscated and even when the auxiliary input is* independent *of the function being obfuscated.*

We also give a positive result. In particular, we show that any obfuscator for the class of point functions is also an obfuscator w.r.t. independent *auxiliary input.*

1 Introduction

The problem of program obfuscation, which practitioners have been engaged in for many years, has recently received attention in the theoretical community as well. This was initiated by the work of Barak *et al.* [BGI+01], who formulated the problem of circuit (program) obfuscation. Loosely speaking, the goal of circuit obfuscation is to make a circuit "unintelligible" while preserving its functionality. This was formalized via the "virtual black box" property, which asserts that any predicate that can be computed (in polynomial time) from the obfuscated circuit can also be computed from the input-output behavior of the circuit (i.e., given black-box access to the circuit).

[BGI+01] showed the *existence* of (contrived) classes of functions that are not obfuscatable. In contrast, the works of Canetti and Wee [C97, W05] show, under various complexity assumptions, how to obfuscate the particular class of point functions, which consists of all boolean functions of the form $I_x(y) = 1$ if and only if $x = y$ (one may think of x as a password and the obfuscation of I_x as a public program that checks whether y is a valid password or not). Thus, it would seem completely possible that most functions of interest can be obfuscated even though in principle general purpose obfuscators do not exist.

In this work we show that this is unlikely to be the case. We consider the definition of *obfuscation w.r.t. auxiliary input*. Namely, we modify the "virtual black box" property, to require that any bit that can be computed from the obfuscated circuit and (polynomial size) auxiliary input z, can also be computed from z and black-box access to the circuit. We first argue that any useful positive result about the possibility of obfuscation must satisfy this extended definition (see Sections 1.1 and 1.2 for examples). We then show that there are many *natural* circuit classes that are not obfuscatable w.r.t. auxiliary input. Notice that, as obfuscation w.r.t. auxiliary input is harder to satisfy than obfuscation without auxiliary input, the result of [BGI+01] already implies the existence of circuits that cannot be obfuscated w.r.t. auxiliary input. Our emphasis is to show that this is actually true for wide and natural classes of circuits.

The idea of requiring security to hold even when an auxiliary input is available to the adversary is not new, and has been present since the early work on auxiliary-input zero-knowledge protocols [GO94]. In the context of zero-knowledge, the requirement is that for every $x \in L$ and for every auxiliary input z, whatever can be learned by a polynomial time verifier that is given (x, z) and interacts with a prover on input x, can also be learned by a polynomial time simulator that is given only (x, z). Intuitively, one may think of z as the *history* observed by the verifier in previous executions. Without this requirement, it is impossible to show secure (even sequential) composition of

[*]This work was supported in part by NSF CNS-0430450, a Minerva Grant 8495, and a Cymerman-Jakubskind award.

[†]This work was supported by an IBM Ph.D. fellowship.

zero-knowledge protocols. Thus, by now, the terms zero-knowledge [GMR88] and auxiliary-input zero-knowledge [GO94] have become one and the same.

In the context of obfuscation, we incorporate auxiliary-input in a very similar manner into the definition. We require that for every auxiliary input z whatever can be learned by a polynomial time non-uniform adversary that is given z and an obfuscated circuit, can be learned by a polynomial time non-uniform simulator that is given z and input/output access to the circuit. We distinguish between two types of obfuscation w.r.t. auxiliary input: *obfuscation w.r.t. dependent auxiliary input* and *obfuscation w.r.t. independent auxiliary input*.[1]

1.1 Dependent Auxiliary Input

Let $\mathcal{C} = \{\mathcal{C}_n\}_{n\in\mathbb{N}}$ be a class of circuits. We say that $\mathcal{O}$ is an *obfuscator w.r.t. dependent auxiliary input* of the class $\mathcal{C}$ if for every function $C \in \mathcal{C}$ the virtual black-box property holds even when the adversary (and the simulator) are given an additional auxiliary input z (this should hold for any z, including one that possibly depends on C).[2]

Example of Obfuscation w.r.t. Dependent Input: A primary usage of obfuscation, pointed out in [BGI+01], is to delegate cryptographic ability. Consider the task of decryption where $D_{SK}(c)$ stands for the decryption algorithm with secret key SK applied to the ciphertext c. Say Alice wants to delegate to her assistant Bob the ability to decrypt all documents which pertain to travel matters. This is easily achieved using an obfuscator $\mathcal{O}$ as follows. Define the function $D1_{SK}(c) \triangleq$ "compute $m = D_{SK}(c)$; output m if and only if m starts with 'subject:travel'," and give Bob the obfuscated program $\mathcal{O}(D1_{SK})$. Say that next month, Alice wants to delegate to Bob the ability to decrypt all documents which pertain to recruiting matters. This is achieved in the same manner: let $D2_{SK}(c) \triangleq$ "compute $m = D_{SK}(c)$; output m if and only if m starts with 'subject:recruiting'," and give Bob the obfuscated program $\mathcal{O}(D2_{SK})$.

Thus, Bob is given obfuscations of two functions $\mathcal{O}(D1_{SK})$ and $\mathcal{O}(D2_{SK})$. We should obviously require that whichever predicate can be learned (in polynomial time) from $\mathcal{O}(D1_{SK})$ *and* $\mathcal{O}(D2_{SK})$, can also be learned (in polynomial time) from input/output access to $D1_{SK}$ and *dependent* auxiliary input $\mathcal{O}(D2_{SK})$.

1.2 Independent Auxiliary Input

The issue of auxiliary input may seem superfluous in those applications of obfuscation designed carefully so that no dependent input is available. Still, we argue that auxiliary inputs which are *independent* of the obfuscated circuit are likely to always exist, and should be considered as well.

To capture the independence of the auxiliary input from the obfuscated circuit C, we fix the auxiliary input z given to the adversary before C is chosen from the class $\mathcal{C}$. Formally, we require that for all auxiliary inputs z given to the adversary, the black-box property should hold for a *randomly* chosen circuit in $\mathcal{C}$.[3]

Whereas requiring the black box property to hold when an auxiliary input is given, is a strengthening of the requirement made by the original [BGI+01] definition, the fact that we require the black box property to hold for a *random* circuit in the class rather than for *every* circuit, is a weakening of the requirements of [BGI+01]'s definition.[4] We emphasize that weakening the definition of obfuscation, strengthens any impossibility results on obfuscation which is the focus of this work. More importantly, we believe that this weakening is meaningful and sufficient for many positive applications of obfuscation, where the particular obfuscated circuit is chosen at random. This is how obfuscation is used in most of the examples of [BGI+01], where generally speaking a class of circuits $\mathcal{C}$ corresponds to a class of cryptographic algorithms (e.g. a class of circuits each performing RSA decryption for a different secret key, or a class of circuits each computing digital signatures for a different signing key, or a class of circuits each computing a pseudo random function for a different seed) and a random choice of $C \in \mathcal{C}$ corresponds to choosing a particular secret key for the cryptographic algorithm at hand. Let us illustrate this with an example.

Example of Obfuscation w.r.t Independent Input: Take any secure digital signature $SIG = (G, S, V)$. Say a signer Alice with secret signing key SK_1 and a signer Carol with an independent signing key SK_2 share an assistant Bob. Alice delegates to Bob the ability to sign on

[1]This distinction was not done in the context of zero-knowledge.

[2]This definition follows the lines of the definition of auxiliary-input zero-knowledge, which also allows the auxiliary input z to depend on the statement x being proven.

[3]At first it may seem that fixing the independent auxiliary input z to the adversary before choosing the circuit to be obfuscated, is equivalent to hard-wiring z to the adversary, and thus that an impossibility result for obfuscation w.r.t. independent auxiliary input implies an impossibility result for obfuscation w.r.t. [BGI+01]'s original definition. However this intuition is misleading, as the following illustrates. Consider a z that satisfies the following three requirements: (1) given z and an obfuscation of $C \in_R \mathcal{C}$ it is easy to compute some predicate $\pi(C, z)$; (2) given z and black-box access to $C \in_R \mathcal{C}$ it is hard to compute the predicate $\pi(C, z)$; (3) given z and black-box access to $C \in_R \mathcal{C}$ *and* the value of some hard computation on z (say $s = f(z)$, for a hard function f) it is easy to compute the predicate $\pi(C, z)$. Then, certainly, requirements (1) and (2) imply that obfuscating C w.r.t. auxiliary input is impossible. In contrast, no such impossibility is implied for the [BGI+01]'s definition, since by requirements (1) and (3), if z is hard-wired to the adversary which is given z and an obfuscation of C, then $f(z)$ can be hard-wired to the simulator which is only given z and black-box access to C. This will enable the simulator to compute π. We note that both in our impossibility results and in our examples we use an auxiliary input of this form.

[4]The negative results of [BGI+01] hold with respect to this weakening.

her behalf the documents which pertain to personal matters, by giving Bob the obfuscated circuit $\mathcal{O}(S1_{SK_1})$, where $S1_{SK_1}(m) \triangleq$ "output $S_{SK_1}(m)$ if and only if m starts with 'subject:personal'." Carol delegates to Bob the ability to sign on her behalf all documents which pertain to student affairs matters, by giving Bob the obfuscated circuit $\mathcal{O}(S2_{SK_2})$, where $S2_{SK_2}(m) \triangleq$ "output $S_{SK_2}(m)$ if and only if m starts with 'subject:student affairs'."

Thus, Bob is given two obfuscations $\mathcal{O}(S1_{SK_1})$ and $\mathcal{O}(S2_{SK_2})$. In this case, in contrast with the example given in Section 1.1, $\mathcal{O}(S1_{SK_1})$ is an auxiliary input *independent* of $\mathcal{O}(S2_{SK_2})$. Naturally, we wish to require that whichever predicate can be learned from $\mathcal{O}(S1_{SK_1})$ and $\mathcal{O}(S2_{SK_2})$, can also be learned from input/output access to $S2_{SK_2}$ and the independent auxiliary input $\mathcal{O}(S1_{SK_1})$.

1.3 Our results

We give separate impossibility results for obfuscation w.r.t. independent auxiliary input and obfuscation w.r.t. dependent auxiliary input. We stress that our impossibility results are unconditional and do not require any intractability assumptions such as the existence of one-way functions. We note that [BGI+01] assume that one-way functions exist in order to obtain a class of circuits that is unobfuscatable.

1.3.1 Impossibility of Obfuscation w.r.t. Independent Auxiliary Input

Our first result considers obfuscation w.r.t. independent auxiliary input. Loosely speaking, we show that many natural *filter functions* (defined below) cannot be obfuscated w.r.t. independent auxiliary input.

Filter functions

Loosely speaking, each filter function is associated with a circuit C and an $\mathcal{NP}$ language L, and is denoted by C^L. The filter function C^L on input (x, w) checks whether $(x, w) \in R_L$ (where R_L is the $\mathcal{NP}$ relation that corresponds to L), and outputs $C(x)$ if and only if w is a valid witness. Thus, C^L gives the value of $C(x)$ only to whoever knows a witness corresponding to x.

Formally, each class of filter functions is associated with a class of circuits $\mathcal{C}$ and an $\mathcal{NP}$ language L. The class of *filter functions* $\mathcal{C}^L \triangleq \{C^L : C \in \mathcal{C}\}$ is the class of circuits where each circuit $C^L \in \mathcal{C}^L$ is defined as follows: $C^L(x, w) = C(x)$ for every input $(x, w) \in R_L$, and $C^L(x, w) = \perp$ for every input $(x, w) \notin R_L$. For example, one may think of $\mathcal{C} = \mathcal{SIG} = \{SIG\}$ as any class of secure digital signature circuits, L as the set $\{(N, y) : y \in QR_N\}$ (the set of quadratic residues mod N), and SIG^L as computing $SIG(N, y)$ only for those users who supply the pair (N, y) together with a square-root of y modulo N. An analogy may be taken from the setting of certification authority: a user's identity corresponds to a pair (N, y) for which only the legal user knows a square root of y mod N, and when he presents this square-root to the trusted center, he gets from the authority a signed certificate of (N, y).

The first result on the impossibility of obfuscation w.r.t. independent auxiliary input is the following:

Result 1 (Informal): Let L be *any* $\mathcal{NP}$-complete language, and let $\mathcal{C}$ be any *any* class of pseudo-random functions, secure encryption algorithms, or randomized digital signature algorithms (where the coins used by the algorithms are replaced by a pseudo random function). Then the class of filter functions $\mathcal{C}^L$ cannot be obfuscated w.r.t. independent auxiliary input.

To express our result in its full generality, we need to introduce the concept of circuits with *super-polynomial pseudo entropy*.[5]

Circuits with super-polynomial pseudo entropy

Pseudo entropy can be thought of as a relaxation of pseudo-randomness. A class of circuits $\mathcal{C}$ is pseudo-random if it is hard to distinguish between having oracle access to $C \in_R \mathcal{C}$ and having oracle access to a totally random function. The pseudo-randomness requirement is very strong: C needs to look *truly random* on *every* element in the domain which is polynomial time computable. Pseudo entropy requires the pseudo-randomness to hold only on a subset of the domain. Moreover, the function need not look truly random on this subset; rather we require the function values on this subset to have "high pseudo entropy."

Specifically, we say that a class of circuits $\mathcal{C}$ has *pseudo entropy at least* $p(\cdot)$ if there exist polynomial size subsets $I_n \subseteq \{0,1\}^n$ such that for $C \in_R \mathcal{C}_n$, the random variable $\{C(x)\}_{x \in I_n}$ cannot be distinguished from a random variable which has statistical min-entropy $p(n)$.[6] We say that $\mathcal{C}$ has *super-polynomial pseudo entropy* if it has pseudo entropy at least $p(\cdot)$, for every polynomial $p(\cdot)$.

We claim that the class of circuits with super-polynomial pseudo entropy is a very broad class, and in particular we show (Claim 4.0.1) that *any* pseudo random function [GGM86], *any* secure encryption algorithm [GM84], and *any* secure probabilistic digital signature algorithm [GMR88] (where the coins used by the algorithms are replaced by a pseudo random function), are examples of classes of circuits with super-polynomial pseudo entropy.

Result 1 can be stated now more generally.

Result 1 (General): The class of filter functions $\mathcal{C}^L$ cannot be obfuscated (in the presence of independent auxiliary inputs), where L is *any* $\mathcal{NP}$-complete language and $\mathcal{C}$ is any class of circuits that satisfies the following two properties:

[5]The term "pseudo entropy" was introduced by [HILL99] in the context of random variables. We extend the use of this term to the context of classes of circuits.

[6]We refer the reader to Section 4 for the precise definition.

1. $\mathcal{C}$ is *strongly unpredictable*. Namely, for every x and for a random $C \in_R \mathcal{C}$ it is hard to predict $C(x)$, given oracle access to C everywhere except x .

2. $\mathcal{C}$ has *super-polynomial pseudo entropy* on inputs in L (i.e., where the polynomial size subsets I_n are contained in $L \cap \{0,1\}^n$).[7,8]

1.3.2 Impossibility of Obfuscation w.r.t. Dependent Auxiliary Input

We next consider obfuscation w.r.t. dependent auxiliary input. We show that if *point-filter functions* (defined below) can be obfuscated then *every* class of circuits with super-polynomial pseudo entropy cannot be obfuscated (in particular, every pseudo random function, every secure encryption algorithm, and every randomized digital signature algorithm cannot be obfuscated).[9] Using a separate proof we show that this condition also implies that the decryption algorithm of *any* secure encryption scheme cannot be obfuscated.

Point-filter functions: Loosely speaking, each point-filter function is associated with an $\mathcal{NP}$ language L, a point x, and a secret bit b. On input w, it outputs its secret bit b if and only if w is a valid witness of x (with respect to the language L). We stress that the language L and the point x are public, whereas the output bit b is secret. Intuitively, the reason that it may be hard to compute the secret bit b is that it may be hard to find a valid witness w for x.[10]

More precisely, every class of point-filter functions is associated with some language $L \in \mathcal{NP}$ and is of the form $\Delta^L = \{\Delta_n^L\}_{n\in\mathbb{N}}$. Every function $\delta_{x,b} \in \Delta_n^L$ is associated with a public string $x \in \{0,1\}^n$ and a secret bit $b \in \{0,1\}$. The function $\delta_{x,b}$ reveals its secret bit b only on inputs w such that $(x,w) \in R_L$ (where R_L is the $\mathcal{NP}$ relation corresponding to the language L). In order to emphasize that x is public, we append x to each output. Formally, $\delta_{x,b}(w) = (x,b)$ if $(x,w) \in R_L$, and $\delta_{x,b}(w) = x$ otherwise. Examples of point-filter classes are $\Delta^{SIG} = \{\delta_{vk,b}\}$, where $\delta_{vk,b}$ reveals its secret bit b only on inputs which are valid signatures w.r.t. the verification key vk, and the class $\Delta^{SAT} = \{\delta_{\phi,b}\}$, where $\delta_{\phi,b}$ reveals its secret bit b only on inputs which satisfy the formula ϕ.

We can now state the result, on the impossibility of obfuscation w.r.t. dependent auxiliary input.

Result 2: *Every* class of circuits with super-polynomial pseudo entropy cannot be obfuscated w.r.t. dependent auxiliary input, or for *every* $\mathcal{NP}$-complete language L, the class of point-filter functions Δ^L cannot be obfuscated w.r.t. dependent auxiliary input.

Thus, we exhibit two classes of functions and show that at least one of them cannot be obfuscated. An alternative, and interesting, conditional formulation of this result, is that if one could obfuscate a single point-filter class Δ^L for an $\mathcal{NP}$-complete language L, then every class of circuits with super-polynomial pseudo entropy cannot be obfuscated.

The question of whether there exists an $\mathcal{NP}$-complete language L for which the point-filter class Δ^L is obfuscatable is thus a worthy future direction to pursue. We show that this question is related to a beautiful fundamental question on the existence of a *hard core predicate for $\mathcal{NP}$ languages* (see Section 6.1 for details).

1.4 On Obfuscating Point Functions

As mentioned earlier, there are two main positive results about the obfuscation of point functions under complexity assumptions.

The work of Canetti [C97] considers the existence of an auxiliary input (in the context of perfect one-way hash-functions), and shows that point functions are weakly obfuscatable w.r.t. dependent auxiliary input, under a strong variant of the DDH Assumption (known as the Strong-DDH Assumption). Canetti does not distinguish between the dependent and independent case. In contrast, the work of Wee [W05] does not consider the existence auxiliary inputs, and it seems unlikely that his obfuscator is robust against dependent auxiliary input (see Section 7 for further discussion).

We show that the question of obfuscating point functions in the presence of *independent* auxiliary input, is equivalent to asking whether obfuscating point functions is possible at all. More formally, we prove the following.

Result 3: If the class of point functions is (weakly) obfuscatable without auxiliary inputs, then it is also (weakly) obfuscatable w.r.t. independent auxiliary inputs.

In [C97], Canetti suggested that one of the usages of obfuscation of point functions is to obtain *content concealing signatures*; i.e., to sign documents m in such a way that if m is known then the signature can be verified as usual, but

[7]We note that every class of pseudo-random functions, secure encryption algorithms and secure probabilistic digital signature algorithms (that use pseudo-random functions to replace their randomness), satisfies these two properties.

[8]Although it may seem that strong unpredictability implies super-polynomial pseudo entropy, and thus that item 2 is superfluous, this is not the case. For example an unpredictable function may be verifiable (i.e., hard to compute but easy to verify), and thus have pseudo entropy 0.

[9]We stress that [BGI+01] presents particular (contrived) examples of classes of pseudo-random functions, encryption algorithms and digital signature algorithms which are not obfuscatable.

[10]Note the contrast between point-filter functions and point functions of [C97, W05]. While point functions are zero everywhere except for one point, a point-filter function can be non-zero on exponentially many inputs (x may have exponentially many witnesses). Moreover, in a point-filter function, x is public and may yield information about the points w which yield the secret value b, whereas there is absolutely no information about which point yields a non-zero value in the point function case.

otherwise the signature of m hides all partial information about m. This proposal is useful for signing more than one message (even for signing random and independent messages) only when the obfuscator of point functions is w.r.t. auxiliary input. Result 3 implies that, when the messages to be signed are chosen from the uniform distribution independently of each other, it suffices to use an obfuscator of point functions (without auxiliary inputs). See Section 7 for elaboration.

1.5 Other Related Work

Interestingly, even prior to the work of [BGI+01], Hada [H00] considered the question of whether pseudo random functions can be obfuscated. He also considered a "virtual black box" type definition. However, he did not restrict the output to be a boolean predicate. Instead, he defined the notion of obfuscation with respect to a given adversary. Namely, an obfuscator which is designed to work against a specific PPT adversary $\mathcal{A}$ (who may be given an auxiliary input). He shows a negative result for obfuscating *any* class of pseudo-random functions against the following adversary $\mathcal{A}$: $\mathcal{A}$ fixes a language $L \in \mathcal{NP}$, fixes a zero-knowledge proof (P, V) for L, and fixes a sequence $\{x_n\}_{n\in\mathbb{N}}$ such that $x_n \notin L$. Given any "obfuscated" circuit C, it outputs an accepting view of $(P, V^*)(x_n)$, where V^* computes its messages by applying C to all previous messages. This can be done using the simulator of the zero-knowledge proof.[11]

Other related works include [CMR98] and [LPS04]. The work of [CMR98] generalizes the work of [C97], in the sense that it relies on weaker and more general assumptions. However, their work yields an obfuscation for the class of point functions, which is weaker in two aspects: (1) the obfuscation is *not* w.r.t. auxiliary inputs. (2) The "virtual black box" property holds only w.r.t. distributions with high min-entropy (i.e., where the point function I_x, to be obfuscated, is chosen according to some high min-entropy distribution over the class of all point functions). The work of [LPS04] is in the Random Oracle Model (which assumes black box access to a truly random function). It generalizes the works of [C97, W05] in the sense that it shows that (in the Random Oracle Model) many access control functions (not only point functions) can be obfuscated.

2 Preliminaries

We adopt the standard way of modelling an efficient adversary as a family of probabilistic polynomial-size circuits. In this paper, A PPT adversary will in fact refer to a family of probabilistic polynomial-size circuits. Similarly, computational indistinguishability refers to indistinguishability by non-uniform poly-size adversaries.

For any language $L \in \mathcal{NP}$ we denote by R_L the $\mathcal{NP}$-relation corresponding to L. Namely $x \in L$ if and only if there exists w such that $(x, w) \in R_L$.

Throughout this paper we distinguish between a family of circuits (which has one circuit for every input size) and a class of circuits (which has many circuits for every input size). A class of circuits is of the form $\mathcal{C} = \{\mathcal{C}_n\}_{n\in\mathbb{N}}$ (denoted by calligraphic letters), where for every $n \in \mathbb{N}$, $\mathcal{C}_n$ is a set of many circuits, each on inputs of size n.

Definition 1 *[BGI+01]: A probabilistic algorithm $\mathcal{O}$ is an* obfuscator *for a class of circuits $\mathcal{C} = \{\mathcal{C}_n\}_{n\in\mathbb{N}}$ if the following three conditions are satisfied:*

- (Functionality): *There exists a negligible function $\mu(\cdot)$ such that for every $n \in \mathbb{N}$ and every $C \in \mathcal{C}_n$, with probability $1 - \mu(n)$ over the internal coin tosses of the obfuscator, $\mathcal{O}(C)$ describes a circuit that computes the same function as C.*[12]
- (Polynomial blowup): *There is a polynomial $l(\cdot)$ such that for every $C \in \mathcal{C}$, $|\mathcal{O}(C)| \le l(|C|)$.*
- ("Virtual black-box" property): *For every* PPT $\mathcal{A}$ *there exists a* PPT $\mathcal{S}$ *and a negligible function $\mu(\cdot)$, such that for every $n \in \mathbb{N}$, every $C \in \mathcal{C}_n$, and every predicate $\pi(\cdot)$,*

$$|Pr[\mathcal{A}(\mathcal{O}(C)) = \pi(C)] - Pr[\mathcal{S}^C(1^n) = \pi(C)]| < \mu(n).$$[13]

Throughout this paper, we restrict our attention to efficient obfuscators, defined as follows.

Definition 2 *An obfuscator $\mathcal{O}$ is said to be* efficient *if it runs in probabilistic polynomial time.*

A few positive results for obfuscation (in the plain model) exist in the literature [C97, CMR98, W05]. All these positive results are for *weak obfuscators* which have the following weaker variant of the "virtual black-box" property:

For every PPT $\mathcal{A}$ *and every polynomial $p(\cdot)$ there exists a* PPT $\mathcal{S}$ *such that for every $n \in \mathbb{N}$, every $C \in \mathcal{C}_n$, and every predicate $\pi(\cdot)$,*

$$|Pr[\mathcal{A}(\mathcal{O}(C)) = \pi(C)] - Pr[\mathcal{S}^C(1^n) = \pi(C)]| < \frac{1}{p(n)}.$$

[11] The result of Hada was conditioned on the existence of a constant-round public-coin zero knowledge proofs, which was later shown to exist by Barak [B01].

[12] The original definition in [BGI+01] considered a slightly stronger functionality property: they required that $\mathcal{O}(C)$ *always* computes the same function as C. This was relaxed as above in [W05].

[13] [BGI+01] formalized the "virtual black-box" property in a different, yet equivalent, way. They required that for every PPT $\mathcal{A}$ there exists a PPT $\mathcal{S}$ and a negligible function $\mu(\cdot)$, such that for every $n \in \mathbb{N}$, and every $C \in \mathcal{C}_n$,

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

We note that the negative results of [BGI+01] hold also for weak obfuscators. Similarly, our negative results hold also for weak obfuscators w.r.t. auxiliary input.

3 Obfuscation w.r.t. Auxiliary Input

We consider two definitions of obfuscation w.r.t. auxiliary input. In the first definition we allow the auxiliary input to *depend* on the function being obfuscated, whereas in the second definition we require the auxiliary input to be *independent* of the function being obfuscated. Both definitions follow the spirit of the original definition of obfuscation given in [BGI+01].

Definition 3 (Obfuscation w.r.t. dependent auxiliary input): *A probabilistic algorithm $\mathcal{O}$ is an* obfuscator w.r.t. dependent auxiliary input *for a class of circuits $\mathcal{C} = \{\mathcal{C}_n\}_{n\in\mathbb{N}}$ if it satisfies the functionality property and the polynomial blowup property as in Definition 1, and in addition it satisfies the following "virtual black box" property:*

For every PPT $\mathcal{A}$ *there exists a* PPT $\mathcal{S}$ *and a negligible function $\mu(\cdot)$, such that for every polynomial $q(\cdot)$, every $n \in \mathbb{N}$, every $C \in \mathcal{C}_n$, every auxiliary input z of size $q(n)$ (z may depend on C), and every predicate $\pi(\cdot)$, the following quantity is smaller than $\mu(n)$:*

$$|Pr[\mathcal{A}(\mathcal{O}(C), z) = \pi(C, z)] - Pr[\mathcal{S}^C(1^n, z) = \pi(C, z)]|.$$

Definition 4 (Obfuscation w.r.t. independent auxiliary input): *A probabilistic algorithm $\mathcal{O}$ is an* obfuscator w.r.t. independent auxiliary input *for a class of circuits $\mathcal{C} = \{\mathcal{C}_n\}_{n\in\mathbb{N}}$ if it satisfies the functionality property and the polynomial blowup property as in Definition 1, and in addition it satisfies the following "virtual black box" property:*

For every PPT $\mathcal{A}$ *there exists a* PPT $\mathcal{S}$ *and a negligible function $\mu(\cdot)$, such that for every polynomial $q(\cdot)$, every $n \in \mathbb{N}$, every auxiliary input z of size $q(n)$, and every predicate $\pi(\cdot)$, the following quantity is smaller than $\mu(n)$:*

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

where the probabilities are over $C \in_R \mathcal{C}_n$, and over the random coin tosses of $\mathcal{A}$, $\mathcal{S}$, and $\mathcal{O}$.

Notice that Definition 4 is weaker than Definition 3, not only because the auxiliary input is independent of the function being obfuscated, but also because in Definition 4 the simulator $\mathcal{S}$ is required to succeed only for random $C \in_R \mathcal{C}_n$ (whereas in Definition 3 the simulator $\mathcal{S}$ is required to succeed for every $C \in \mathcal{C}_n$). As was noted in the Introduction, considering only randomly chosen circuits seems to suffice for most cryptographic applications. Moreover, even if Definition 4 does seem to be too weak, this is not a concern to us, since we are proving impossibility results, and impossibility of achieving a weak definition implies impossibility of achieving a stronger one.

Our negative results hold also for the notion of *weak obfuscation w.r.t. auxiliary input*, which is defined analogously to weak obfuscation (without auxiliary input). In Section 5 we show negative results for weak obfuscation w.r.t. independent auxiliary input, and in Section 6 we show negative results for weak obfuscation w.r.t. dependent auxiliary input. In order to state formally each of these results we need to define the notion of *pseudo entropy of a class of circuits*.

4 High Pseudo Entropy Circuits

Loosely speaking, we say that a class of circuits $\mathcal{C}$ has pseudo entropy at least $p(\cdot)$ if there exist polynomial size sets $I_n \subseteq \{0,1\}^n$ such that the set $C(I_n)$ looks as if it has min-entropy at least $p(n)$, even given oracle access to C on $\bar{I}_n \triangleq \{0,1\}^n \setminus I_n$. This is formalized as follows.

Definition 5 (Pseudo entropy of a class of circuits:) *We say that a class of circuits $\mathcal{C} = \{\mathcal{C}_n\}_{n\in\mathbb{N}}$ has* pseudo entropy *at least $p(\cdot)$ if there exists a polynomial $t(\cdot)$ and sets $I_n \subseteq \{0,1\}^n$ of size $t(n)$, and for every $C \in \mathcal{C}_n$ there is a random variable $\vec{Y}^C = (Y_1, \ldots, Y_{t(n)})$ such that the following holds:*

1. *$\vec{Y}^C$ has (statistical) min entropy at least $p(n)$.*[14]
2. *For every* PPT *oracle machine $\mathcal{D}$ there is a negligible function $\mu(\cdot)$ such that for every $n \in \mathbb{N}$,*

$$|Pr[\mathcal{D}^{C|_{\bar{I}_n}}(\vec{Y}^C) = 1] - Pr[\mathcal{D}^{C|_{\bar{I}_n}}(C(I_n)) = 1]| \leq \mu(n),$$

where the probability is over $C \in_R \mathcal{C}_n$, $\vec{Y}^C$, and the random coin tosses of $\mathcal{D}$. The circuit $C|_{\bar{I}_n}$ agrees with C on every $x \notin I_n$ and outputs $\perp$ on every $x \in I_n$.

There is a slight abuse of notations here. We use I_n to denote both a set and a list (or a vector). For $I_n = (x_1, \ldots, x_{t(n)})$ we let $C(I_n) = (C(x_1), \ldots, C(x_{t(n)}))$.

Definition 6 *We say that a class of circuits $\mathcal{C} = \{\mathcal{C}_n\}_{n\in\mathbb{N}}$ has* super-polynomial pseudo entropy *if it has pseudo entropy at least $p(\cdot)$, for every polynomial $p(\cdot)$.*

We give a few examples of natural classes of circuits that have super-polynomial pseudo entropy.

Claim 4.0.1 *The following classes of circuits all have super-polynomial pseudo entropy:*

1. Every *class of pseudo-random functions.*

[14]A random variable X over some set S is said to have (statistical) min-entropy at least k if for every $x \in S$, $Pr[X = x] \leq 2^{-k}$.

2. Every *secure randomized digital signature algorithm,*[15] *in which the signer replaces the randomness by applying a (secret) pseudo-random function to the message to be signed.*

3. Every *secure secret-key and public-key encryption algorithm, in which the randomness is replaced by applying a (secret) pseudo-random function to the message to be encrypted.*

Proof: Due to lack of space we prove only item 2. Let SIG be *any* secure randomized digital signature algorithm, and let SIG' be the deterministic signature algorithm obtained by taking any pseudo-random function ensemble $\mathcal{F} = \{f_s\}$, and modifying the (randomized) signing algorithm $SIG_{SK}(\cdot)$ by appending a (random) seed s of $\mathcal{F}$ to its signing key, and by setting $SIG'_{SK,s}(M) \triangleq SIG_{SK}(M; f_s(M))$. For every set of $t(n)$ messages $I_n = (M_1, \ldots, M_{t(n)})$, let $\vec{Y}^{SK} = (Y_1, \ldots, Y_{t(n)})$ be a sequence of $t(n)$ independent random variables, where each Y_i is identically distributed to $SIG_{SK}(M_i)$. The fact that Y_i has (statistical) min-entropy at least 1, implies that $\vec{Y}^{SK}$ has (statistical) min-entropy at least $t(n)$. It remains to notice that the pseudo-randomness of $\mathcal{F}$ implies that every PPT oracle machine $\mathcal{D}^{SIG'_{SK,s}|_{\bar{I}_n}}$ cannot distinguish between the random variable $SIG'_{SK,s}(I_n) = (SIG'_{SK,s}(M_1), \ldots, SIG'_{SK,s}(M_{t(n)}))$ and the random variable $\vec{Y}^{SK} = (Y_1, \ldots, Y_{t(n)})$ (for randomly chosen SK, s), implying that SIG' has super-polynomial pseudo entropy. ■

Remark. The reason circuits with super-polynomial pseudo entropy are useful in the context of obfuscation, is that when proving impossibility results, we exploit the following distinction:

1. An obfuscation $\mathcal{O}(C)$ is a small (polynomial size) circuit that agrees with C on I_n.

2. Given black-box access to C, it is hard to construct a small circuit that agrees with C on I_n.

When arguing for (2), we use the fact that $\mathcal{C}$ has super-polynomial pseudo entropy. If $\mathcal{C}$ has super-polynomial pseudo entropy, then given black box access to $C|_{\bar{I}_n}$, it is hard to distinguish (for $C \in_R \mathcal{C}_n$) between the pair $(I_n, C(I_n))$ and the pair $(I_n, \vec{Y}^C)$, where $\vec{Y}^C$ is a random variable with high (statistical) min-entropy. Using the connection between (statistical) min entropy and compression, it can be shown that with high probability (over the random variable $\vec{Y}^C$) there does not exist a small circuit v such that $v(x_i) = Y_i$, where $I_n = (x_1, \ldots, x_{t(n)})$ and $\vec{Y}^C = (Y_1, \ldots, Y_{t(n)})$. Thus, it must be the case that for a random $C \in \mathcal{C}_n$ it is hard to come up with a small circuit that agrees with C on I_n. Otherwise, this can be used to distinguish between the pair $(I_n, C(I_n))$ and the pair $(I_n, \vec{Y}^C)$. We elaborate on this in Sections 5 and 6.

5 Impossibility of Obfuscation w.r.t. Independent Auxiliary Input

In this section, we define the classes of *filter functions*, and show that many natural classes of filter functions are not even weakly obfuscatable w.r.t. independent auxiliary input.

Definition 7 *Let $L \in \mathcal{NP}$, and let $\mathcal{C}$ be any class of circuits. For every $C \in \mathcal{C}$ we define $C^L(x, w) = C(x)$ if $(x, w) \in R_L$, and $C^L(x, w) = \perp$ otherwise. The class of* filter functions *$\mathcal{C}^L$ is defined by $\mathcal{C}^L \triangleq \{C^L : C \in \mathcal{C}\}$.*

We show that the class of filter functions $\mathcal{C}^L$ is not weakly obfuscatable w.r.t. *independent* auxiliary input, for L and $\mathcal{C} = \{\mathcal{C}_n\}_{n \in \mathbb{N}}$ that satisfy the following properties:

1. *L is an $\mathcal{NP}$-complete language.*

2. *$\mathcal{C}$ is strongly unpredictable:* For every $x \in \{0,1\}^n$, and for a random $C \in_R \mathcal{C}_n$, given oracle access to C everywhere except at the point x, it is hard to guess $C(x)$ (except with negligible probability).

3. *$\mathcal{C}$ has super-polynomial pseudo entropy over elements in L*: for every polynomial $p(\cdot)$ there exists a polynomial $t(\cdot)$ and sets $I_n \subseteq L \cap \{0,1\}^n$ of $t(n)$ elements, and for every $C \in \mathcal{C}_n$ there exists a random variable $\vec{Y}^C = (Y_1, \ldots, Y_{t(n)})$ with (statistical) min entropy at least $p(n)$, such that every PPT oracle machine $\mathcal{D}^{C|_{\bar{I}_n}}$ cannot distinguish (for $C \in_R \mathcal{C}_n$) between the random variable $C(I_n)$ and the random variable $\vec{Y}^C$.

We note that all our natural examples of classes of circuits given in Claim 4.0.1, satisfy both properties (2) and (3).

Theorem 8 *For every L and $\mathcal{C}$ that satisfy the above three properties, the class of circuits $\mathcal{C}^L$ cannot be weakly obfuscated w.r.t. independent auxiliary input.*

Proof Idea: The proof is by contradiction. We assume that there exist $\mathcal{C}$ and L as above, such that $\mathcal{C}^L$ can be weakly obfuscated w.r.t. independent auxiliary input by an obfuscator $\mathcal{O}$. The main idea is to exploit the fact that $\mathcal{C}^L$ has the property, that for every $n \in \mathbb{N}$ there is a set $I_n = (x_1, \ldots, x_{t(n)})$ of $t(n)$ elements in $L \cap \{0,1\}^n$, and a set $W_n = (w_1, \ldots, w_{t(n)})$ of $t(n)$ corresponding witnesses, such that:

[15] A signature scheme is said to be randomized if for every message M and for every signing key SK, the random variable $SIG_{SK}(M)$ has (statistical) min-entropy at least 1.

1. For every PPT $\mathcal{S}$, given black-box access to a random circuit $C^L \in_R \mathcal{C}_n^L$ it is hard to come up with a "small" circuit that computes C^L on $(x_1, w_1), \ldots, (x_{t(n)}, w_{t(n)})$. This follows from the fact that $\mathcal{C}$ has super-polynomial pseudo entropy over L.

2. For every $C \in \mathcal{C}_n$, $\mathcal{O}(C^L)$ is itself a "small" circuit that computes C^L on $(x_1, w_1), \ldots, (x_{t(n)}, w_{t(n)})$.

We exploit this difference to obtain a contradiction. However, for this we need to show that given $\mathcal{O}(C^L)$ it is easy to compute a *single bit* b, whereas every PPT oracle machine $\mathcal{S}^{(C^L)}$ fails to compute this bit (with noticeable probability). This is where our auxiliary input comes into play.

Our first idea was to take the auxiliary input z to be an obfuscation of a circuit that outputs a secret bit b only on inputs that encode a small circuit that agrees with C^L on $(x_1, w_1), \ldots, (x_{t(n)}, w_{t(n)})$. Such an approach is actually taken in Section 6. However, here we need to take a different approach: first, because we need the auxiliary input to be independent of our obfuscated circuit, and z as defined above does depend on our obfuscated circuit C^L; and second, because we do not want to add the additional assumption that the above z is obfuscatable.

Instead, we let z be an obfuscation of a random filter function $K^L \in_R \mathcal{C}_n^L$.[16,17] We show that there exists a string $x^* \in \{0,1\}^n$ (that depends only on I_n and $C(I_n)$) such that:

1. For every PPT $\mathcal{S}$, given black-box access to a random circuit $C^L \in_R \mathcal{C}_n^L$, and given $\mathcal{O}(K^L)$ it is hard to compute $K(x^*)$.

2. For every $C \in \mathcal{C}_n$, given $(\mathcal{O}(C^L), \mathcal{O}(K^L))$ it is easy to compute $K(x^*)$.

The full proof of Theorem 8 is omitted due to lack of space.

6 Impossibility of Obfuscation w.r.t. Dependent Auxiliary Input

In this section we define the classes of *point-filter functions*. We show that if *point-filter functions* can be weakly obfuscated w.r.t. dependent auxiliary input then *every* class of circuits with super-polynomial pseudo entropy cannot be weakly obfuscated w.r.t. dependent auxiliary input.

Definition 9 *For every $L \in \mathcal{NP}$, the class $\Delta^L = \{\Delta_n^L\}_{n \in \mathbb{N}}$ is a class of* point-filter functions*, where $\Delta_n^L = \{\delta_{x,b}\}_{x \in \{0,1\}^n, b \in \{0,1\}}$ and $\delta_{x,b}(w) = (x,b)$ if $(x,w) \in R_L$ and $\delta_{x,b}(w) = x$ otherwise.*

[16]Notice that K^L is independent of C^L.

[17]Actually, in the full proof we let z be an obfuscation of $K^{L'} \in_R \mathcal{C}_n^{L'}$, where L' is some specific $\mathcal{NP}$-language. We use the $\mathcal{NP}$-completeness of L to deduce that $\mathcal{C}^{L'}$ is obfuscatable (assuming $\mathcal{C}^L$ is obfuscatable).

Theorem 10 *For every $\mathcal{NP}$-complete language L the class Δ^L is not weakly obfuscatable w.r.t. dependent auxiliary input, or every class $\mathcal{C}$ with super-polynomial pseudo entropy is not weakly obfuscatable w.r.t. dependent auxiliary input.*

The proof of Theorem 10 is omitted due to lack of space, and the intuition is summarized below.

Proof Idea: Assume that there exists an $\mathcal{NP}$-complete language L such that the class Δ^L is weakly obfuscatable w.r.t. dependent auxiliary input. This implies that for every $\mathcal{NP}$ language L' the class $\Delta^{L'}$ is also weakly obfuscatable w.r.t. dependent auxiliary input. (This follows from the existence of an $\mathcal{NP}$-reduction.[18])

We need to prove that every class $\mathcal{C}$ with super-polynomial pseudo entropy is not weakly obfuscatable w.r.t. dependent auxiliary input. Assume for the sake of contradiction that there exists a class $\mathcal{C}$ with super-polynomial pseudo entropy that can be weakly obfuscated w.r.t. dependent auxiliary input by an obfuscator $\mathcal{O}$.

The main idea is to exploit the fact for every $n \in \mathbb{N}$ there is a polynomial size set $I_n \subseteq \{0,1\}^n$, such that:

1. For every PPT $\mathcal{S}$, given black-box access to a random circuit $C \in_R \mathcal{C}_n$, it is hard to come up with a "small" circuit that computes C on I_n.

2. For every $C \in \mathcal{C}_n$, $\mathcal{O}(C)$ is itself a "small" circuit that computes C on I_n.

We want to exploit this difference to obtain a contradiction. However, for this we need to show that given $\mathcal{O}(C)$ it is easy to compute a *single bit* b, whereas every PPT oracle machine $\mathcal{S}^C$ fails to compute this bit (with noticeable probability). This is where the auxiliary input comes into play.

We let the auxiliary input $z = z_C$ be a point-filter function that is associated with a secret bit b. z_C outputs its secret bit b only on inputs that encode a "small" circuit that agrees with C on the elements in I_n. More precisely, z_C is the point filter function $\delta_{(I_n, C(I_n)),b}$, where a valid witness for $(I_n, C(I_n))$ is a "small" circuit that agrees with C on elements in I_n.

Thus, it remains to show:

1. For every PPT $\mathcal{S}$, given black-box access to a random circuit $C \in_R \mathcal{C}_n$, and given z_C it is hard to compute the secret bit b.

2. For every $C \in \mathcal{C}_n$, given $(\mathcal{O}(C), z_C)$ it is easy to compute the secret bit b.

It is easy to see that (2) holds, since $\mathcal{O}(C)$ is a valid witness of $(I_n, C(I_n))$. Thus, given $(\mathcal{O}(C), z_C)$, the secret bit

[18]We actually need to assume here that the $\mathcal{NP}$-reduction is witness preserving. We refer to the final version of this work for more details.

b can be easily computed by evaluating z_C on input $\mathcal{O}(C)$. However, it is not clear that (1) holds, since it may actually be easy to extract the secret bit b from z_C, which can be any circuit computing $\delta_{(I_n,C(I_n)),b}$. To hide the secret bit b from $\mathcal{S}$, we obfuscate this point-filter function. Namely, the auxiliary input z_C that we consider is an obfuscation of $\delta_{(I_n,C(I_n)),b}$. Now, it is easy to see intuitively that (1) holds since $\mathcal{S}$ does not have any valid witness of $(I_n, C(I_n))$, and therefore does not have any advantage in guessing the secret bit b from its obfuscated point-filter function at hand.

Using similar ideas, we can prove the following theorem.

Theorem 11 *For every $\mathcal{NP}$-complete language L the class Δ^L is not weakly obfuscatable w.r.t. dependent auxiliary input, or for* every *secure (secret-key or public-key) encryption scheme (G, E, D), the class of decryption algorithms $D = \{D_{sk}\}$ is not weakly obfuscatable w.r.t. dependent auxiliary input.*

6.1 Are Point-filter Functions Obfuscatable?

Theorems 10 and 11 both give conditional results. We would much prefer to have the explicit result that every class with super-polynomial pseudo entropy is not weakly obfuscatable w.r.t. dependent auxiliary input, which would imply that many natural cryptographic tasks (such as pseudo-random functions, encryption algorithms, signature algorithms, etc.) cannot be weakly obfuscated w.r.t. dependent auxiliary input. Therefore, we think that it is worth investigating the question of whether point-filter functions are weakly obfuscatable w.r.t. dependent auxiliary input. We do not have a complete answer to this question. Rather, we relate it to another (seemingly unrelated) problem that is of independent interest. We show that the existence of a weak obfuscator w.r.t. dependent auxiliary input for the class Δ^L, is closely related to the existence of a *hard-core predicate for the language L*.

Intuitively, $B(\cdot)$ is a hard-core predicate for the language L if the following two conditions hold: (1) for every $x \in L$, given any witness of x it is easy to compute $B(x)$. (2) It is hard to compute $B(x)$ without knowing a witness for x. We formalize the hard-core predicate $B(\cdot)$ as a probabilistic predicate, as follows.

Definition 12 (Hard-core predicate for L): *A randomized predicate $B(\cdot)$ is said to be a* hard-core predicate *for $L \in \mathcal{NP}$, if the following two conditions hold:*

1. *There exists a* PPT *machine $\mathcal{A}_1$ and a polynomial $p(\cdot)$ such that for every $(x, w) \in R_L$ and every $r \in \{0,1\}^{|x|}$,*

$$Pr[\mathcal{A}_1(x, w, r) = B(x, r)] \geq \frac{1}{2} + \frac{1}{p(|x|)},$$

where the probability is over the random coin tosses of $\mathcal{A}_1$.

2. *There exists a* PPT *oracle machine $\mathcal{A}_2$ such that for every polynomial $q(\cdot)$ there exists a polynomial $p(\cdot)$ such that for every $x \in L$,*

$$Pr[\mathcal{A}_2^{f_x}(x) = w \text{ s.t. } (x, w) \in R_L] \geq \frac{1}{p(|x|)},$$

where f_x is a function that on input a random r outputs $B(x, r)$ with probability $\frac{1}{2} + \frac{1}{q(|x|)}$, and where the probability is over the random coin tosses of $\mathcal{A}_2$.

Theorem 13 *If $L \in \mathcal{NP}$ has a hard-core predicate then the class Δ^L is weakly obfuscatable w.r.t. dependent auxiliary input.*

The proof of Theorem 13 is omitted due to lack of space.

7 Obfuscation of Point Functions

As was mentioned in the Introduction, the only (non-trivial) positive results for obfuscation (in the plain model) are for point functions [C97, CMR98, W05]. In this section, we explore the question of whether point functions can be obfuscated w.r.t. auxiliary input.

When considering *independent* auxiliary input we can prove the following:

Theorem 14 *Every obfuscator* (*resp. weak obfuscator*) *without auxiliary input is also an obfuscator* (*resp. weak obfuscator*) *w.r.t. independent auxiliary input.*

An interesting application of Theorem 14, suggested by Canetti [C97], is to content concealing signature schemes. Canetti's idea is the following. Take any secure digital signature $SIG = (G, S, V)$, and an obfuscator $\mathcal{O}$ for point functions (recall, $I_x(y) = 1$ if and only if $x = y$) and create a secure content concealing digital signature scheme $SIG' = (G, S', V')$ as follows. The signing algorithm S' on message m outputs $(\mathcal{O}(I_m), S(\mathcal{O}(I_m)))$. The verification algorithm V' on input message m and a presumed signature (C, σ) accepts if and only if $C(m) = 1$ and (C, σ) is a legal message and signature pair according to V.

It is easy to see that if the obfuscator $\mathcal{O}$ satisfies the virtual black box property, then SIG' is a secure content-concealing signature when used to sign *one-message*. However, in order to claim that this remains the case when several messages are signed, one must show that whatever predicate can be learned from $\mathcal{O}(I_m)$ *and* $O(I_{m'})$, can also be learned from input/output access to I_m and *auxiliary input* $\mathcal{O}(I_{m'})$. Theorem 14 implies that as long as the messages are drawn independently at random from $\{0,1\}^n$, proving the existence of obfuscator $\mathcal{O}$ for point functions suffices for this application, without having to address the issue of independent auxiliary inputs.

Theorem 14 does not seem to hold in the case of *dependent* auxiliary input. In particular we believe that the obfuscator in [W05] is not robust against dependent auxiliary input, as the following illustrates.

The obfuscator presented in [W05] assumes the existence of a very strong one-way permutation π, and is defined as follows: For every $x \in \{0,1\}^n$, and randomness $\tau_1, \ldots, \tau_{3n} \in_R \{0,1\}^n$, the obfuscator $\mathcal{O}(I_x; \tau_1, \ldots, \tau_{3n})$ is defined to be:

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

For a random string R, consider the auxiliary input $z(x,R) \triangleq (x \oplus R, \pi(x) \oplus R, \ldots, \pi^{3n-1}(x) \oplus R)$.

Claim 7.0.2 *Assuming the function* $z : (x,R) \rightarrow z(x,R)$ *cannot be inverted* (*except with negligible probability over random* x, R), $\mathcal{O}$ *is not an obfuscator w.r.t. dependent auxiliary input.*

Proof Sketch of Claim 7.0.2: It suffice to prove that the following two statements are true.

1. For every $n \in \mathbb{N}$, every $x, R \in \{0,1\}^n$, given the pair $(\mathcal{O}(I_x; \tau_1, \ldots, \tau_{3n}), z(x,R))$, it is easy to compute x.
2. For every PPT oracle machine $\mathcal{S}$ and for infinitely many n's, $\mathcal{S}^{I_x}(z(x,R))$ does not have any advantage in guessing x (for $x, R \in_R \{0,1\}^n$).

We omit further details due to lack of space.

8 Directions for Future Work

An intriguing question posed by this work is whether there exists an $\mathcal{NP}$-complete language L such that the point-filter class Δ^L is obfuscatable w.r.t. dependent auxiliary input? A positive answer would imply very strong negative results: every class with super-polynomial pseudo entropy is *not* obfuscatable w.r.t. dependent auxiliary input.

On the positive front, an interesting direction to pursue is to try to extend the model in which obfuscation is done, with the hope that such extension would enable wide and meaningful possibility results. For example, one may consider the question of whether obfuscation is possible in the Common Reference String (CRS) model. This is a natural first choice, as non-interactive zero-knowledge proofs, which can be thought of as "proof obfuscation" is possible in the CRS model. Unfortunately, our negative results, as well as the negative results of [BGI+01], hold also in the CRS model. Still, finding the right model which would be realizable in applications and in which wide obfuscation could be done is a worthy direction to pursue.

References

[B01] B. Barak. How to Go Beyond the Black-Box Simulation Barrier. In 42*nd FOCS*, pages 106–115, 2001.

[BGI+01] B. Barak, O. Goldreich, R. Impagliazzo, S. Rudich, A. Sahai, S. Vadhan, K. Yang On the (Im)possibility of Obfuscating Programs. In *CRYPTO 2001*, pages 1-18.

[C97] R. Canetti. Towards Realizing Random Oracles: Hash Functions That Hide All Partial Information. *CRYPTO 1997*, pages 455-469.

[CMR98] R. Canetti, D. Micciancio, and O. Reingold. Perfectly One-Way Probabilistic Hash Functions. In *Proceeding of STOC 1998.*

[G01] O. Goldreich. *Foundations of Cryptography: Vol. 1: Basic Tools.* Cambridge University Press, 2001.

[G04] O. Goldreich. *Foundations of Cryptography: Vol. 2: Basic Applications*. Cambridge University Press, 2004.

[GL89] O. Goldreich and L.A. Levin. A Hard-Core Predicate for all One-Way functions. In *Proceedings of the Twenty-First Annual ACM Symposium on Theory of Computing*, pages 25-32, 1989.

[GM84] S. Goldwasser and S. Micali. Probabilistic encryption. in *Journal of Computer and System Sciences*, 28:270–299, 1984.

[GGM86] G. Goldreich, S. Goldwasser, and S. Micali. How to Construct Random Functions. In *Journal of the ACM,* 33(4), 1984, 792-804.

[GMR88] S. Goldwasser, S. Micali, R. L. Rivest A Digital Signature Scheme Secure Against Adaptive Chosen-Message Attacks. In *SIAM Journal of Computation,* 17(2): 281-308 (1988).

[GO94] O. Goldreich and Y. Oren. Definitions and Properties of Zero-Knowledge Proof Systems. In *Journal of Cryptology,* Vol. 7, No. 1, pages 1-32, 1994.

[H00] S. Hada. Zero-Knowledge and Code Obfuscation. In *Advances in Cryptology – ASIACRYPT '00*.

[HILL99] J. Hastad, R. Impagliazzo, L.A. Levin, M. Luby. A Pseudorandom Generator from any One-Way Function. In *SIAM Journal on Computing* 28 (1999), pages 1364-1396 (electronic).

[LPS04] B.Lynn, M. Prabhakaran, and A. Sahai. Positive Results and Techniques for Obfuscation. In *Proceedings of Eurocrypt*, 2004.

[W05] H. Wee. On Obfuscating Point Functions. In *eprint* 2005/001.