More on the Impossibility of Virtual-Black-Box Obfuscation with Auxiliary Input

Nir Bitansky[*] Ran Canetti[†] Omer Paneth[‡] Alon Rosen[§]

October 28, 2013

Abstract

We show that if there exist indistinguishability obfuscators for a certain class C of circuits then there do not exist independent-auxiliary-input virtual-black-box (VBB) obfuscators for any family of circuits that compute a pseudo-entropic function. A function f_k is pseudo-entropic if it is hard, given oracle access to f_k but without asking explicitly on a value x, to distinguish $f_k(x)$ from a random variable with some real entropy.

This strengthens the bound of Goldwasser and Kalai [FOCS '05, ePrint '13] that rules out *dependent*-auxiliary-input VBB obfuscation for the same set of circuit families, assuming inditinguishability obfuscators for another class, C', of circuits. That is, while they only rule out the case where the adversary and the simulator obtain auxiliary information that depends on the actual (secret) obfuscated function, we rule out even the case where the auxiliary input depends only on the (public) family of programs.

1 Introduction

The rigorous treatment of *program obfuscation* was initiated by Barak et al. [BGI$^+$01], who formulated a number of security notions for the task. The strongest and most applicable of these notions is *virtual black-box obfuscation* (VBB), requiring that any adversary trying to learn information from the obfuscated program, cannot do better than a simulator that is given only black-box access to the program. Barak et al. demonstrated a (contrived) class of programs that cannot be VBB obfuscated, but left open the possibility that natural and possibly expressive classes of programs may still be obfuscated. Subsequently, VBB obfuscators were shown only for a number of restricted (and mostly simple) classes of programs [Can97, CD08, CRV10, BR13]. To date, the classification of which programs can or cannot be VBB obfuscated is still not well understood.

In contrast, for other, more relaxed notions of obfuscation, recent progress suggests a much clearer and positive picture: Garg et al. [GGH$^+$13] propose a candidate construction for *indistinguishability obfuscation* for *all* circuits. This notion (referred to as $i\mathcal{O}$, from hereon) only requires that it is hard to distinguish an

[*]Tel Aviv University. Email: `nirbitan@tau.ac.il`. Supported by an IBM Ph.D. Fellowship, and the Check Point Institute for Information Security.

[†]Boston University and Tel Aviv University. Email: `canetti@bu.edu`. Supported by the Check Point Institute for Information Security, an NSF EAGER grant, and an NSF Algorithmic foundations grant 1218461.

[‡]Boston University. Email: `omer@bu.edu`. Supported by the Simons award for graduate students in theoretical computer science and an NSF Algorithmic foundations grant 1218461.

[§]Efi Arazi School of Computer Science, IDC Herzliya, Israel. Email: `alon.rosen@idc.ac.il`. Supported by ISF grant no. 1255/12 and by the ERC under the EU's Seventh Framework Programme (FP/2007-2013) ERC Grant Agreement n. 307952.

obfuscation of C_0 from an obfuscation of C_1, for any two circuits C_0 and C_1 of the same size that compute the same function [BGI$^+$01, GR07]. Indeed, unlike the case of VBB obfuscation, for $i\mathcal{O}$ there are no known impossibilities. Furthermore, the Garg et al. construction, and variants thereof, were shown to satisfy the VBB guarantee in ideal algebraic oracle models [CV13, BR13, BGTK$^+$13]. So far, however, none of the above results proved useful in achieving VBB obfuscation in the plain model.

A main step towards understanding which programs can or cannot be VBB obfuscated was made by Goldwasser and Kalai [GK05], who gave evidence that circuits that compute *pseudo-entropic functions* cannot be obfuscated under a strong definition of VBB with respect to *auxiliary input*. Pseudo-entropic functions can be seen as a generalization of pseudo-random functions where there exists a set of inputs I such that, for a random function f in the class, the output of f on I appears to have high entropy, even given black-box access to f outside of I. As shown there, various basic cryptographic primitives, indeed, fall into the class of pseudo-entropic functions, and are thus susceptible to the impossibility results.

Dependent vs. independent auxiliary input. Goldwasser and Kalai consider two variants of auxiliary-input obfuscation: VBB with *dependent* auxiliary input and VBB with *independent* auxiliary input. In the case of dependent auxiliary input, the VBB property is required to hold even when the auxiliary input given to the adversary and simulator depends on the actual (secret) obfuscated circuit. In the case of independent auxiliary input, the requirement is weakened: The auxiliary input may depend only on the family of circuits (which is public). The actual circuit to be obfuscated is chosen randomly from the class, independently of the auxiliary input given to the adversary and simulator.

For the case of dependent auxiliary input, Goldwasser and Kalai show that pseudo-entropic functions cannot be VBB obfuscated, assuming that a different class of *point filter functions* can be VBB obfuscated In a recent note, they show that the assumption can be relaxed to $i\mathcal{O}$ of point filter functions [GK13]. For the weaker notion of VBB with independent auxiliary input, they only show a more restricted impossibility result for a subclass of pseudo-entropic functions called *filter functions*.

We stress that, in both cases of independent and dependent auxiliary input, the same *joint* auxiliary input is given to both the adversary and the simulator. We also allow the adversary and simulator to each have additional *individual* auxiliary input. The simulator's individual auxiliary input is fixed after the adversary's individual auxiliary input, but before the joint auxiliary input.

We next briefly motivate the need of auxiliary input when using VBB obfuscation in applications. As usual in cryptography, security with respect to auxiliary input is needed when obfuscation is used together with other components in a larger scheme or protocol. Consider, for example, a zero-knowledge protocol where one of the prover messages to the verifier contains an obfuscated program $\mathcal{O}(C)$. To prove that the protocol is zero-knowledge, we would like to show that any verifier $\mathcal{V}$ has a zero-knowledge simulator $\mathcal{S}_{\mathsf{zk}}$ that can simulate $\mathcal{V}$'s view in the protocol. Intuitively, $\mathcal{S}_{\mathsf{zk}}$ would rely on the security of $\mathcal{O}$ by thinking of $\mathcal{V}$ as an "obfuscation adversary", trying to learn information from $\mathcal{O}(C)$. Such an adversary has an "obfuscation simulator" $\mathcal{S}_{\mathcal{O}}$ that can learn the same information given only black-box access to C, and $\mathcal{S}_{\mathsf{zk}}$ can try and use $\mathcal{S}_{\mathcal{O}}$. The problem is that the view of $\mathcal{V}$ does not depend only on the code of $\mathcal{V}$, but also on auxiliary input z to $\mathcal{V}$, such as other prover messages and the statement being proven. Here, an obfuscation definition that only supports individual auxiliary input is insufficient. Indeed, such a definition guarantees that for every z, there exists an obfuscation simulator $\mathcal{S}_{\mathcal{O}}$ with an individual auxiliary input z'; however, the zero-knowledge simulator $\mathcal{S}_{\mathsf{zk}}$ may not be able to efficiently compute z' from z.

The problem is avoided by considering a definition that guarantees the existence of a single obfuscation simulator that can simulate the view of $\mathcal{V}$ given any auxiliary input. Here, if the obfuscated program C depends on other prover message or on the statement, the obfuscation should be be secure with dependent auxiliary input. Else, obfuscation with independent auxiliary input suffices.

1.1 Our Result

We show that, assuming $i\mathcal{O}$, VBB obfuscation is impossible for any class of pseudo-entropic functions *even in the case of independent auxiliary input:*

Theorem 1.1 (informal)**.** *Assuming indistinguishability obfuscation for a certain class of circuits, there exist no independent-auxiliary-input virtual-black-box obfuscators for any circuit family that computes a pseudo-entropic function.*

Comparison to [GK05]. Our result can be seen as an extension of the negative results of Goldwasswer and Kalai [GK05] who rule out *all* pseudo-entropic functions but only with respect to *dependent auxiliary input*, and rule out the restricted class of *filter functions* with respect to *independent auxiliary input*. In terms of assumptions, the independent auxiliary-input result of [GK05] is unconditional and the dependent auxiliary-input result can be based on $i\mathcal{O}$ *for point filter functions* [GK13]. The result in this work relies on $i\mathcal{O}$ for a different class of functions, related to *puncturable pseudo-random functions* [BGI13, BW13, KPTZ13, SW13].

1.2 Proof Idea

To understand the main ideas behind our result, we first recall in somewhat more detail the concept of pseudo-entropic functions. A class of circuits $\mathcal{C} = \{\mathcal{C}_n\}$ is pseudo entropic (or "has superpolynomial pseudo entropy") if for every polynomial $p(n)$ and for every $n \in \mathbb{N}$ there exists a set of inputs $I = I_n$ such that for a random circuit $C \in \mathcal{C}_n$ the set of outputs $C(I)$ has pseudo-entropy at least $p(n)$. That is, $C(I)$ is computationally indistinguishable from a random variable with statistical min-entropy $p(n)$), even given oracle access to the value of C on all inputs outside I. At high-level, we rely on one central feature of pseudo-entropic functions: given oracle access to a random circuit $C \in \mathcal{C}_n$ it is hard to compress the set $C(I)$. In other words, it is infeasible to find a circuit $\tilde{C}$ such that $|\tilde{C}| \ll p(n)$ and $\tilde{C}$ agrees with C on all inputs in I. In fact, there is an indistinguishable oracle whose output on I has true min-entropy $p(n)$, in which case such a circuit $\tilde{C}$ is unlikely to exist.

To establish that $\mathcal{C}$ cannot be VBB obfuscated with independent auxiliary input, we follow the same high-level approach as [GK05]. We show a distribution $\mathcal{Z}$ of auxiliary inputs, such that for a random circuit $C \in \mathcal{C}_n$, and independent auxiliary input Z drawn from $\mathcal{Z}$, it is possible to learn a predicate $\pi(C, Z)$ from Z and an obfuscation $\mathcal{O}(C)$. But a simulator $\mathcal{S}$, given Z and oracle access to C, cannot do so.

Specifically, let m be the length of the obfuscation $\mathcal{O}(C)$ for circuits $C \in \mathcal{C}_n$. The pseudo-entropy of $\mathcal{C}$ implies that there exists a set of inputs I, such that, given an oracle to a random $C \leftarrow \mathcal{C}_n$, it is hard to come up with a circuit of size at most m that agrees with C on the set I. In contrast, an obfuscation $\mathcal{O}(C)$ is exactly such a circuit! We take advantage of this asymmetry using the auxiliary input.

Intuitively, our goal is to devise as auxiliary input a circuit that allows learning a secret predicate of C only when provided a small circuit that agrees with C on I. Concretely, the auxiliary-input Z would describe a circuit that takes as input a circuit $\tilde{C}$ of size m; the circuit Z would first evaluate $\tilde{C}$ on all points in I, and derive the vector of outputs $\tilde{C}(I)$. Then, it would apply a (one-bit) pseudo-random function G_s to $\tilde{C}(I)$, and output the resulting bit $\mathsf{G}_s(\tilde{C}(I))$.

The adversary, given input $\mathcal{O}(C)$, will run $Z(\mathcal{O}(C))$ and will obtain $\mathsf{G}_s(\tilde{C}(I))$. It now remains to argue that the simulator cannot predict $\mathsf{G}_s(\tilde{C}(I))$ even given Z and an oracle access to C. For this purpose we apply an iO to the circuit in Z and use the *puncturing technique* of Sahai and Waters [SW13]. In more detail, we consider an alternative auxiliary input $Z^*_{C(I)}$ that is defined like Z, except that it has a punctured key s^*. This punctured key allows evaluating G_s on all inputs except for the special input $x^* = C(I)$, while the value of $\mathsf{G}_s(C(I))$ remains pseudo-random. We define the output of $Z^*_{C(I)}$ on $x^* = C(I)$ to be arbitrary (e.g., 0). Now, given $Z^*_{C(I)}$, the simulator clearly cannot learn the bit $\mathsf{G}_s(C(I))$. It thus remains to show that Z and $Z^*_{C(I)}$ are indistinguishable.[1]

As described the circuits Z and $Z^*_{C(I)}$ may disagree on the input $x^* = C(I)$, and therefore their indistinguishability does not follow directly from the $i\mathcal{O}$ property. Instead, we consider an experiment where $\mathcal{S}$ has access to an indistinguishable oracle whose output outside the set I is the same as C but on I, instead of answering according to $C(I)$ it answers according to a random variable Y that has min-entropy much higher than m. The simulator $\mathcal{S}$ would be given accordingly an auxiliary input Z^*_Y where G_s is punctured on the point $x^* = Y$, instead of on the point $C(I)$. By the pseudo-randomness guarantee at the punctured point, in this experiment, the simulator $\mathcal{S}$ cannot learn the bit $\mathsf{G}_s(Y)$. Furthermore, Since Y has high pseudo-entropy, a circuit $\tilde{C}$ of size m such that $\tilde{C}(I) = Y$ is unlikely to exist, and therefore the circuits Z and Z^*_Y compute the same function, and we can invoke the $i\mathcal{O}$ security guarantee to deduce that the simulator cannot learn the bit $\mathsf{G}_s(Y)$, even when given the original Z, rather than Z^*_Y. Switching back to the indistinguishable experiment where $\mathcal{S}$ gets oracle access to C, we conclude that $\mathcal{S}^C(Z)$ is not able to learn the bit $\mathsf{G}_s(C(I))$.

Comparison to the technique of [GK05, GK13]. In both works the impossibility for pseudo-entropic functions is obtained by having the auxiliary input describe an obfuscated circuit Z that outputs a secret bit b when given a small circuit that agrees with the obfuscated circuit $\mathcal{O}(C)$ on a set of inputs I. In [GK05, GK13], this idea is implemented by hardcoding the set of outputs $C(I)$ and the secret bit b directly in the auxiliary input circuit Z; specifically their circuit Z outputs b when given as input a circuit $\tilde{C}$ such that $\tilde{C}(I) = C(I)$. Consequently in their case the circuit Z depends on C.

To make Z independent of C, we avoid hardcoding the set $C(I)$ into Z. Instead, our circuit Z has a hardcoded pseudo-random function G_s, independent of C, and the output bit b is defined as $\mathsf{G}_s(C(I))$. When given an input circuit $\tilde{C}$ that doesn't agree with C on I, Z outputs the bit $\mathsf{G}_s(\tilde{C}(I))$ which is (pseudo) independent of b. This means that b is not part of the description of Z, which means that Z does not depend on C. However, this also makes the proof more delicate: In the proof of [GK05, GK13], the bit b (that is hardcoded to Z) can be hidden from the simulator using VBB obfuscation or even $i\mathcal{O}$. In our case, to prove that $i\mathcal{O}$ of Z hides the bit b, we use the fact that the pseudo-random function G_s is puncturable.

Finally we remark that the approach taken in this work can also be seen as a simplified version of the approach taken by Goldwasser and Kalai to prove impossibility for *independent* auxiliary input for *filter functions*.

2 Definitions

We define unpredictable functions, virtual-black-box obfuscation with independent auxiliary input, indistinguishability obfuscation, and puncturable pseudo-random functions.

2.1 Circuits with Super-Polynomial Pseudo-Entropy

We recall the definition of pseudo-entropy of circuits from [GK05]. Roughly, a class of circuits $\mathcal{C} = \{\mathcal{C}_n\}_{n\in\mathbb{N}}$ is said to have pseudo-entropy p if there exists polysize sets of inputs $\{I_n\}_{n\in\mathbb{N}}$, such that for a random $C \leftarrow \mathcal{C}_n$, the set of outputs $C(I_n)$ appears to have min-entropy p, even given an oracle that computes C on all inputs outside I_n.

[1] For this to go through, we assume that the pseudo-random function G_s is puncturable. Notice that this is not and additional assumption since the existence of pseudo-entropic functions implies one-way functions, which in turn are sufficient for constructing puncturable PRFs.

Definition 2.1 (Pseudo-entropy of circuits [GK05]). *A class of circuits $\mathcal{C} = \{\mathcal{C}_n\}_{n\in\mathbb{N}}$ is said to have pseudo-entorpy p if there exist sets $\{I_n\}_{n\in\mathbb{N}}$ of polynomial size $t(n)$, and for every $C \in \mathcal{C}_n$, there is a random variable $Y^C = (Y_1, \ldots, Y_{t(n)})$, such that*

1. *Y^C has min-entropy at least $p(n)$.*
2. *For any polysize distinguisher D, and all large enough n:*
$$\left|\Pr\left[D^{C\circ Y^C}(1^n) = 1\right] - \Pr\left[D^C(1^n) = 1\right]\right| \le \mathrm{negl}(n) \ \ ,$$
where the probability is over a random C from $\mathcal{C}_n$, and over Y^C, and $C \circ Y^C$ is an oracle that answers according to C, except on inputs from I_n, for which it answers according to Y^C.

We say that $\mathcal{C}$ has **super-polynomial pseudo-entropy** *if it has pseudo-entropy p for any polynomial p.*

2.2 Virtual-Black-Box Obfuscation With Independent Auxiliary Input

We recall the definition of VBB obfuscation with independent auxiliary input from [GK05].

Definition 2.2. *A PPT algorithm $\mathcal{O}$ is an auxiliary-input obfuscator for a circuit family $\mathcal{C}$ if it satisfies:*

1. **Functionality:** *For any $C \in \mathcal{C}$,*
$$\Pr_{\mathcal{O}}\left[\forall x : \mathcal{O}(C)(x) = C(x)\right] = 1 \ \ .$$
2. **Virtual black-box:** *For any PPT adversary $\mathcal{A}$, there is a simulator $\mathcal{S}$, such that for every $n \in \mathbb{N}$, auxiliary input $z \in \{0,1\}^{\mathrm{poly}(n)}$ and every predicate π:*
$$\left|\Pr[\mathcal{A}(\mathcal{O}(C), z) = \pi(C, z)] - \Pr[\mathcal{S}^C(1^{|C|}, z) = \pi(C, z)]\right| \le \mathrm{negl}(|C|) \ \ ,$$
where the probability is over $C \leftarrow \mathcal{C}_n$, and the randomness of the algorithms $\mathcal{O}$,$\mathcal{A}$ and $\mathcal{S}$.

Remark 2.1 (On the auxiliary input). For our results, it is critical that the simulator $\mathcal{S}$ receives the same *joint* auxiliary input z, as $\mathcal{A}$ does, and has to operate efficiently with respect to this auxiliary input. This flavor of definition is standard in defining auxiliary-input security, e.g., auxiliary-input zero-knowledge, and auxiliary-input extractable functions [BCPR13]. Our result still hold if the adversary and simulator are allowed additional *individual* auxiliary input, where the simulator's individual auxiliary input is fixed after the adversary's individual auxiliary input, but before the joint auxiliary input (equivalently, in the above definition both $\mathcal{A}$ and $\mathcal{S}$ are modeled as non-uniform families of circuits.)

2.3 Indistinguishability Obfuscation

Indistinguishability obfuscation was introduced in [BGI$^+$01] and given a candidate construction in [GGH$^+$13], and subsequently in [BR13, BGTK$^+$13, CV13].

Definition 2.3 (Indistinguishability obfuscation [BGI$^+$01]). *A PPT algorithm $i\mathcal{O}$ is said to be an* indistinguishability obfuscator *(INDO) for $\mathcal{C}$, if it satisfies:*

1. **Functionality:** *As in Definition 2.2.*
2. **Indistinguishability:** *For any class of circuit pairs $\{(C_n^{(1)}, C_n^{(2)}) \in \mathcal{C} \times \mathcal{C}\}_{n\in\mathbb{N}}$, where the two circuits in each pair are of the same size and functionality, it holds that:*
$$\left\{i\mathcal{O}(C_n^{(1)})\right\}_{n\in\mathbb{N}} \approx_c \left\{i\mathcal{O}(C_n^{(2)})\right\}_{n\in\mathbb{N}} \ \ .$$

2.4 Puncturable PRFs

We next define puncturable PRFs. We consider a simple case of the puncturable PRFs where any PRF might be punctured at a single point. The definition is formulated as in [SW13].

Definition 2.4 (Puncturable PRFs)**.** *Let ℓ, m be polynomially bounded length functions. An efficiently computable family of functions*

$$\mathcal{G} = \left\{ \mathsf{G}_s : \{0,1\}^{m(n)} \to \{0,1\}^{\ell(n)} \;\middle|\; s \in \{0,1\}^n, n \in \mathbb{N} \right\} \ ,$$

associated with an efficient (probabilistic) key sampler $\mathsf{Gen}_{\mathcal{G}}$, is a puncturable PRF if there exists a puncturing algorithm Punc that takes as input a key $s \in \{0,1\}^n$, and a point x^*, and outputs a punctured key s_{x^*}, so that the following conditions are satisfied:

1. **Functionality is preserved under puncturing:** *For every $x^* \in \{0,1\}^{\ell(n)}$,*

$$\Pr_{s \leftarrow \mathsf{Gen}_{\mathcal{G}}(1^n)} \left[\forall x \neq x^* : \mathsf{G}_s(x) = \mathsf{G}_{s_{x^*}}(x) \;\middle|\; s_{x^*} = \mathsf{Punc}(s, x^*) \right] = 1 \ .$$

2. **Indistinguishability at punctured points:** *The following ensembles are computationally indistinguishable:*
   - $\{x^*, s_{x^*}, \mathsf{G}_s(x^*) \mid s \leftarrow \mathsf{Gen}_{\mathcal{G}}(1^n), s_{x^*} = \mathsf{Punc}(s, x^*)\}_{x^* \in \{0,1\}^{m(n)}, n \in \mathbb{N}}$
   - $\left\{x^*, s_{x^*}, u \;\middle|\; s \leftarrow \mathsf{Gen}_{\mathcal{G}}(1^n), s_{x^*} = \mathsf{Punc}(s, x^*), u \leftarrow \{0,1\}^{\ell(n)}\right\}_{x^* \in \{0,1\}^{m(n)}, n \in \mathbb{N}}$.

To be explicit, we include x^* in the distribution; throughout, we shall assume for simplicity that a punctured key s_{x^*} includes x^* in the clear. As shown in [BGI13, BW13, KPTZ13], the GGM [GGM86] PRF yield puncturable PRFs as defined above.

3 The Impossibility Result

In this section, we show that no class $\mathcal{C}$ of circuits with super-polynomial pseudo-entropy can be VBB obfuscated with respect to independent auxiliary input, assuming $i\mathcal{O}$ for (a related) class of circuits. We first describe, for any class $\mathcal{C}$ as above, an auxiliary-input distribution ensemble $\mathcal{Z}$ and a PPT adversary $\mathcal{A}$, such that, given an obfuscation of $C \leftarrow \mathcal{C}$ and $z \leftarrow \mathcal{Z}$, $\mathcal{A}$ always learns some predicate $\pi(C, z)$. Then, we show that any PPT simulator that is only given oracle access to C fails to learn the predicate. See the Introduction for a high-level overview of the proof.

Let $\mathcal{C} = \{\mathcal{C}_n\}_{n \in \mathbb{N}}$, be a class of circuits with super-polynomial pseudo-entropy such that each $C \in \mathcal{C}_n$ maps $\{0,1\}^{\ell(n)}$ to $\{0,1\}^{\ell'(n)}$. Let $\mathcal{O}$ be any candidate obfuscator for $\mathcal{C}$, and let $m(n)$ be a polynomial such that for every $C \in \mathcal{C}_n$, $|\mathcal{O}(C)| \leq m(n)$.

The auxiliary input distribution $\mathcal{Z}$. By assumption, $\mathcal{C}$ has pseudo-entropy at least $m(n) + n$. Let $\{I_n\}_{n \in \mathbb{N}}$ be the sets guaranteed by Definition 2.1, where I_n is of polynomial size $t(n)$. Let $\mathcal{G}$ a puncturable one-bit PRF family:

$$\mathcal{G} = \left\{ \mathsf{G}_s : \{0,1\}^{\ell'(n) \cdot t(n)} \to \{0,1\} \;\middle|\; s \in \{0,1\}^n, n \in \mathbb{N} \right\} \ .$$

We define two circuit families

$$\mathcal{K} = \left\{ K_s : \{0,1\}^{m(n)} \to \{0,1\} \;\middle|\; s \in \{0,1\}^n, n \in \mathbb{N} \right\} \ ,$$
$$\mathcal{K}^* = \left\{ K_{s_{x^*}} : \{0,1\}^{m(n)} \to \{0,1\} \;\middle|\; s \in \{0,1\}^n, x^* \in \{0,1\}^{\ell'(n)\cdot t(n)}, n \in \mathbb{N} \right\} \ .$$

The circuit K_s, given a circuit $\tilde{C} : \{0,1\}^\ell \to \{0,1\}^{\ell'}$ of size m, computes $x := \tilde{C}(I_n) := (\tilde{C}(i))_{i \in I_n}$, and outputs $\mathsf{G}_s(x)$.

Hardwired: a PRF key $s \in \{0,1\}^n$, and the set I_n.

Input: a circuit $\tilde{C} : \{0,1\}^\ell \to \{0,1\}^{\ell'}$, where $|\tilde{C}| = m(n)$.

```
1. Compute $x = \tilde{C}(I_n)$.
2. Return $\mathsf{G}_s(x)$.
```

Figure 1: The circuit K_s.

The circuit $K_{s_{x^*}}$, has a hardwired PRF key s_{x^*} that was derived from s by puncturing it at the point x^*. It operates the same as K_s, only that when if $x = x^*$, it outputs an arbitrary bit, say, 0. In particular, if for all circuits $\tilde{C} \in \{0,1\}^{m(n)}$, it holds that $x^* \neq \tilde{C}(I_n)$, then $K_{s_{x^*}}$ and K_s compute the exact same function.

Hardwired: a punctured PRF key $s_{x^*} = \mathsf{Punc}(s, x^*)$, the set I_n.

Input: a circuit $\tilde{C} : \{0,1\}^\ell \to \{0,1\}^{\ell'}$, where $|\tilde{C}| = m(n)$.

```
1. Compute $x = \tilde{C}(I_n)$.
2. If $x \neq x^*$, return $\mathsf{G}_{s_{x^*}}(x)$.
3. If $x = x^*$, return 0.
```

Figure 2: The circuit $K_{s_{x^*}}$.

We are now ready to define our auxiliary-input distribution $\mathcal{Z} = \{Z_n\}_{n \in \mathbb{N}}$. Let $d = d(n)$ be the maximal size of circuits in either $\mathcal{K}$ or $\mathcal{K}^*$, corresponding to security parameter n. Denote by $[K]_d$ a circuit K padded with zeros to size d, and by $[\mathcal{K}]_d$ the class of circuits where every circuit $K \in \mathcal{K}$ is replaced with $[K]_d$. Let $i\mathcal{O}$ be an indistinguishability obfuscator for the class $[\mathcal{K} \cup \mathcal{K}^*]_d$.

The distribution Z_n simply consists of an obfuscated (padded) circuit K_s for a randomly generated s

The adversary $\mathcal{A}$ and predicate π. The adversary $\mathcal{A}$, given auxiliary input $z = [i\mathcal{O}(K_s)]_{d(n)}$ and an obfuscation $\mathcal{O}(C)$, where $C \in \mathcal{C}_n$, outputs

$$z(\mathcal{O}(C)) = K_s(\mathcal{O}(C)) = \mathsf{G}_s(\mathcal{O}(C)(I_n)) = \mathsf{G}_s(C(I_n)) \ ,$$

1. Sample $s \leftarrow \mathsf{Gen}_{\mathcal{G}}(1^n)$.
2. Sample an obfuscation $z \leftarrow i\mathcal{O}([K_s]_{d(n)})$.
3. Output z.

Figure 3: The auxiliary input distribution Z_n.

where the above follows by the definition of K_s and the functionality of $i\mathcal{O}$ and $\mathcal{O}$.

Thus, $\mathcal{A}$ always successfully outputs the predicate

$$\pi(C, K_s) = K_s(C) = \mathsf{G}_s(C(I_n)) \ \ .$$

Adversary $\mathcal{A}$ cannot be simulated. We prove the following proposition implying that the candidate obfuscator $\mathcal{O}$, for the class $\mathcal{C}$, fails to meet the VBB requirement (Definition 2.2)

Proposition 3.1. *For any PPT simulator $\mathcal{S}$, and all large enough $n \in \mathbb{N}$:*

$$\Pr_{\substack{C \leftarrow \mathcal{C}_n \\ z \leftarrow Z_n}} \left[\mathcal{S}^C(z) = \pi(C, z)\right] \leq \frac{1}{2} + \mathrm{negl}(n) \ \ .$$

Proof. Assume towards contradiction that there exists a PPT $\mathcal{S}$ that learns $\pi(C, z)$ with probability $\frac{1}{2} + \epsilon(n)$, for some noticeable ϵ (and infinitely many $n \in \mathbb{N}$). We show how to use $\mathcal{S}$ to break either the pseudo-entropy of $\mathcal{C}$, or the pseudo-randomness at punctured points of $\mathcal{G}$.

According to the definition of Z_n, it holds that

$$\Pr\left[\mathcal{S}^C(i\mathcal{O}([K_s]_d) = \mathsf{G}_s(C(I_n))\right] \geq \frac{1}{2} + \epsilon(n) \ \ ,$$

where the probability is over $C \leftarrow \mathcal{C}_n$, $s \leftarrow \mathsf{Gen}_{\mathcal{G}}(1^n)$, and the coins of $\mathcal{S}$.

Now, for every $C \in \mathcal{C}_n$, let $Y^C = (Y_1, \dots, Y_t)$ be the random variable guaranteed by the pseudo-entropy of values in I_n (Definition 2.1). We first consider an alternative experiment where the oracle C is replaced with an oracle $C \circ Y^C$ that behaves like C on all points outside I_n, and on points in I_n answers according to Y^C. We claim that

$$\Pr\left[\mathcal{S}^{C \circ Y^C}(i\mathcal{O}([K_s]_d) = \mathsf{G}_s(Y^C)\right] \geq \frac{1}{2} + \epsilon(n) - \mathrm{negl}(n) \ \ ,$$

where the probability is over $C \leftarrow \mathcal{C}_n$, Y^C, $s \leftarrow \mathsf{Gen}_{\mathcal{G}}(1^n)$, and the coins of $\mathcal{S}$. Indeed, this follows directly from the pseudo-entropy guarantee (Definition 2.1), and the fact that a distinguisher can sample s, and compute $i\mathcal{O}([K_s]_d)$ on its own.

Next, we change the above experiment so that, instead of an $i\mathcal{O}$ of K_s, the simulator gets an $i\mathcal{O}$ of the circuit $K_{s_x^*}$, where s is punctured at the point $x^* = Y^C$. We claim that

$$\Pr\left[\mathcal{S}^{C \circ Y^C}(i\mathcal{O}([K_{s_{x^*}}]_d) = \mathsf{G}_s(Y^C)\right] \geq \frac{1}{2} + \epsilon(n) - \mathrm{negl}(n) \ \ ,$$

where the probability is over $C \leftarrow \mathcal{C}_n, Y^C$, $s \leftarrow \mathsf{Gen}_{\mathcal{G}}(1^n)$, and the coins of $\mathcal{S}$, $x^* = Y^C$, and $s_{x^*} = \mathsf{Punc}(s, x^*)$. Indeed, recalling that, for any $C \in \mathcal{C}_n$, Y^C has min-entropy $m(n) + n$, there does not exist a circuit $\tilde{C}$ such that $x^* := Y^C = \tilde{C}(I_n)$, except with negligible probability 2^{-n}. However, recall that in this case K_s and $K_{s_{x^*}}$ have the exact same functionality, and thus the above follows by the $i\mathcal{O}$ guarantee.

It is now left to note that $\mathcal{S}$ predicts with noticeable advantage the value of G_s at the punctured point x^*, and thus violates the pseudo-randomness at punctured points requirement (Definition 2.4). $\square$

Acknowledgments.
We thank Yael Kalai for useful discussions on relaxing the assumption in [GK05] to $i\mathcal{O}$.

References

[BCPR13] Nir Bitansky, Ran Canetti, Omer Paneth, and Alon Rosen. Indistinguishability obfuscation vs. auxiliary-input extractable functions: One must fall. Cryptology ePrint Archive, Report 2013/599, 2013.

[BGI$^+$01] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. In *CRYPTO*, pages 1–18, 2001.

[BGI13] Elette Boyle, Shafi Goldwasser, and Ioana Ivan. Functional signatures and pseudorandom functions. *IACR Cryptology ePrint Archive*, 2013:401, 2013.

[BGTK$^+$13] Boaz Barak, Sanjam Garg, Yael Tauman-Kalai, Omer Paneth, and Amit Sahai. Protecting obfuscation against algebraic attacks. *IACR Cryptology ePrint Archive*, 2013:631, 2013.

[BR13] Zvika Brakerski and Guy Rothblum. Virtual black-box obfuscation for all circuits via generic graded encoding. *IACR Cryptology ePrint Archive*, 2013:563, 2013.

[BW13] Dan Boneh and Brent Waters. Constrained pseudorandom functions and their applications. *IACR Cryptology ePrint Archive*, 2013:352, 2013.

[Can97] Ran Canetti. Towards realizing random oracles: Hash functions that hide all partial information. In *CRYPTO*, pages 455–469, 1997.

[CD08] Ran Canetti and Ronny Ramzi Dakdouk. Extractable perfectly one-way functions. In *Proceedings of the 35th International Colloquium on Automata, Languages and Programming*, pages 449–460, 2008.

[CRV10] Ran Canetti, Guy N. Rothblum, and Mayank Varia. Obfuscation of hyperplane membership. In *TCC*, pages 72–89, 2010.

[CV13] Ran Canetti and Vinod Vaikuntanathan. Obfuscating branching programs using black-box pseudo-free groups. *IACR Cryptology ePrint Archive*, 2013:500, 2013.

[GGH$^+$13] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *FOCS*, 2013.

[GGM86] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *J. ACM*, 33(4):792–807, 1986.

[GK05] Shafi Goldwasser and Yael Tauman Kalai. On the impossibility of obfuscation with auxiliary input. In *FOCS*, pages 553–562, 2005.

[GK13] Shafi Goldwasser and Yael Tauman Kalai. A note on the impossibility of obfuscation with auxiliary input. Cryptology ePrint Archive, Report 2013/665, 2013. `http://eprint.iacr.org/`.

[GR07] Shafi Goldwasser and Guy N. Rothblum. On best-possible obfuscation. In *TCC*, pages 194–213, 2007.

[KPTZ13] Aggelos Kiayias, Stavros Papadopoulos, Nikos Triandopoulos, and Thomas Zacharias. Delegatable pseudorandom functions and applications. *IACR Cryptology ePrint Archive*, 2013:379, 2013.

[SW13] Amit Sahai and Brent Waters. How to use indistinguishability obfuscation: Deniable encryption, and more. *IACR Cryptology ePrint Archive*, 2013:454, 2013.