Indistinguishability Obfuscation vs. Auxiliary-Input Extractable Functions: One Must Fall

Nir Bitansky∗ Ran Canetti† Omer Paneth‡ Alon Rosen§

June 3, 2014

This is an out of date draft. The paper was merged with How to Construct Extractable Functions against Uniform Adversaries [BCP13] to form On the Existence of Extractable One-Way Functions [BCPR14].

Abstract

We show that if there exist indistinguishability obfuscators for all circuits then there do not exist auxiliary-input extractable one-way functions. In particular, the knowledge of exponent assumption with respect to adversaries with auxiliary input is false in any group where computing discrete logarithms is intractable. The proof uses the “punctured programs” technique of [Sahai-Waters 2013].

1 Introduction

1.1 Program obfuscation

Program obfuscation, namely the task of making code unintelligible while preserving its functionality, has been long considered to be a holy grail of cryptography, with diverse and far reaching applications. The rigorous treatment of obfuscation was initiated by Barak et al. [BGI+01], who formulated a number of definitions of security for this task. However, until very recently, we only knew how to obfuscate a number of specific and restricted classes of programs under any of these definitions. Furthermore, Barak et al. demonstrated a class of programs that are unobfuscatable according to a natural definition, namely virtual black box (VBB) obfuscation, which guarantees that access to the obfuscated program gives no more power than access to an impenetrable black box with the same input-output functionality. Impossibility results for more natural classes of programs, with respect to stronger variants of VBB obfuscation, subsequently followed [GK05].

All of this changed with the work by Garg et al. [GGH+13b] who proposed a candidate construction of general-purpose obfuscators. They show that, under algebraic assumptions closely related to multilinear

∗Tel Aviv University. Email: nirbitan@tau.ac.il. Supported by an IBM Ph.D. Fellowship, the Check Point Institute for Information Security, and an ISF grant 20006317.
†Boston University and Tel Aviv University. Email: canetti@bu.edu. Supported by the Check Point Institute for Information Security, an ISF grant 20006317, 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.
maps [GGH13a, CLT13], their construction satisfies the relaxed notion of indistinguishability obfuscation (iO) [BGI+01], for which no impossibility results are known. The notion of iO only requires that it is hard to distinguish an 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 exact same function.

The security of the Garg et al. construction is based on a specific family of intractability assumptions (different for any obfuscated function). Being introduced only recently, these assumptions are still not well-understood, though several recent works have verified the validity of the constructions (or variants thereof) in idealized algebraic models [CV13, BR13, BGTK+13]. In fact, in these models the construction is even shown to obtain the stronger VBB notion.

A priori, it is not clear how strong or meaningful is the iO notion of security. However, as observed in [BGI+01, GR07], iO is “best possible,” in the sense that if a class of programs is obfuscatable according to some notion of security, then a general iO obfuscator, applied to this class of programs, will provide essentially the same security guarantee. Furthermore, several recent works [GGH+13b, SW13, HSW13, GGHR13] showed that general iO obfuscation can be combined with more standard cryptographic primitives to construct many powerful primitives such as functional encryption, public-key encryption from one way functions, attribute-based encryption, as well as NIZKs, CCA encryption, 2-message multi-party computation [GGHR13], deniable encryption, and more. Despite these dramatic advances, many questions are still open and the full range of implications of iO for all circuits still seems far from being well understood.

1.2 Extractable Functions

The concept of extractable functions originates with the work of Damgård over 20 years ago [Dam92], which first formulated the “knowledge of exponent assumption” (KEA). The KEA notion, with respect to a group $G$, says that for any adversary $A$, there exists an extractor $E$, such that whenever $A$, given two random generators $(g, h)$ of $G$ outputs two group elements of the form $(g^x, h^x)$ for some $x$, then $E$, given the same $(g, h)$, outputs $x$. Variants of this assumption have been used in the context of CCA and plaintext aware encryption, zero knowledge, non-interactive succinct arguments and other primitives, e.g., [Dam92, HT98, BP04a, BP04b, Gro10, BSW12, GST12].

Abstracting from this assumption, Canetti and Dakdouk [CD08, CD09] defined the notion of extractable function families. Similarly to KEA, a family of functions $F$ is extractable if for any adversary $A$ there exists an extractor $E$, such that whenever $A$, given a random key $e$ for a function $f_e \in F$, outputs an element $y$ in the image of $f_e$, then $E$, given the same $e$, outputs a preimage of $y$. Intuitively, this means that the “only way” to generate a value in the image of $f_e$ is to “honestly” apply $f_e$ on some chosen input. When $F$ has additional hardness properties (such as one-wayness or collision resistance), this abstraction has proven to be quite powerful [CD09, BCCT12, DFH12, GLR11].

Different formulations of assumptions of this kind exhibit widely different properties. While variants differ in several aspects, let us concentrate on a particular aspect: the “advice”, or “auxiliary information” available to the adversary and extractor. One straightforward formulation requires that, for any possible adversary (modeled as a uniform algorithm) there exists an extractor (again, modeled as a uniform algorithm) that successfully extracts as described above, given the adversary’s coin tosses. An alternative is to model both the adversary and the extractor as non-uniform families of polysize circuits.

However, neither formulation suffices when using extractable functions with other components in a larger cryptographic scheme or protocol. Indeed, during the execution of such a protocol or scheme, an adversary $A$ may gather information $z$ from other components and use it as additional auxiliary input when evaluating the extractable function. While, in the non-uniform definition, for every $z$, there exists an extractor for $A(z, \cdot)$, a reduction/simulator might not be able to efficiently find this extractor. Similar issues are
encountered in various cryptographic contexts that involve composition: a classic example, from the context of zero-knowledge, is in proving that zero-knowledge is closed under sequential composition. There, the solution is to require a stronger notion of auxiliary-input zero-knowledge.

In the context of extractable functions, the solution is to require a single extractor that can handle any auxiliary information $z$ gathered by the adversary. Specifically, we require that for any polytime adversary $\mathcal{A}$ there should exist a polytime extractor $\mathcal{E}$ such that extraction succeeds when $\mathcal{A}$ and $\mathcal{E}$ are given the same advice string $z$. That is, for any polysize $z$, and for a randomly chosen key $e$, the probability that $\mathcal{A}$, given $(z, e)$ outputs a value $y$ in the image of $f_e$ and $\mathcal{E}$, given $(z, e)$, does not output a preimage of $y$, is negligible. We call this property auxiliary-input extractability.

Indeed, this notion is needed in order to use extractable functions to obtain the standard notion of auxiliary-input zero-knowledge. In certain cases, auxiliary-input extractability can be relaxed to consider only the case where the common auxiliary input is taken from some specific distribution that captures the ‘possible’ auxiliary information in a given system (see e.g. [BCCT12]).

**Do auxiliary-input extractable functions exist?** With one recent exception, in which the adversary is assumed to have only bounded-length advice [BCP13], we do not have any candidate extractable one-way functions with an explicit, constructive extraction algorithm under any of the above formulations. Instead, existence of such an extractor is merely assumed (e.g., [Dam92, CD09, BCCT12, Gro10, GGPR13]. Such assumptions are arguably not satisfying. In particular, they do not qualify as “efficiently falsifiable” [Nao03]; namely, unlike standard assumptions where it possible to algorithmically study the best possible “breakers”, here we do not even have an algorithmic way to test whether a given adversary $\mathcal{A}$ breaks the assumption.

### 1.3 Our Result

Auxiliary-input extractability is a strong requirement: the auxiliary-input $z$ may potentially encode arbitrary circuits, which may be executed by the adversary, meaning that the extractor needs to extract from arbitrary circuits. Given our current lack of understanding of non-black-box extraction techniques, the latter further decreases our confidence in such assumptions. Furthermore, the need to extract from arbitrary code reveals a clear tension between extractable functions and obfuscation: if $z$ contains obfuscated code, how can we expect the extractor to algorithmically extract useful information out of it?

We show that general iO suffices to make this intuition rigorous:

**Theorem 1.1.** If there exist indistinguishability obfuscators for all circuits, then there do not exist one-way functions that are auxiliary-input extractable.

**So, is the knowledge of exponent assumption wrong?** Originally [Dam92], the knowledge of exponent assumption (KEA) was not stated with auxiliary-input extractability, but rather according to the notion where every adversary $\mathcal{A}$ has an extractor $\mathcal{E}$, and the only joint extra information is the adversary’s coin tosses and key for the function. In particular, given a non-uniform adversary $\mathcal{A}$ with an obfuscated code as advice $z$, the extractor is allowed to have a different advice $z'$, representing the “deobfuscated” code. Indeed, our result does not rule out such a notion of extraction (even assuming iO for all circuits). Our result does not disvalidate the intuition that “the only way” to compute $(g^x, h^x)$, given $(g, h)$ is by “knowing” $x$. As we shall see, our adversary and auxiliary-input will be devised so that $x$ is actually known, but only by an underlying obfuscated computation, and thus cannot be figured out efficiently from it by an external extractor.

We also note that our result does not rule out extractable functions with respect to adversaries with bounded polynomial advice, such as those constructed in [BCP13]. Neither do they rule out extractable
functions with respect to auxiliary input that is taken from specific distributions, e.g. the uniform distribution, required in [BCCT12].

1.4 Proof Idea

To show that the existence of $iO$ rules out auxiliary-input extractable functions, we follow the basic intuition given above. We focus on the ‘hardest scenario’, where the auxiliary input $z$ may represent an arbitrary malicious, and potentially obfuscated code. Specifically, we consider the following folklore case (sketched in [BCCT12, BC12, BCI+13, BCCT13]) where $z$ is an obfuscation of a circuit $C_k$ that, given key $e$ for an extractable $f_e$, chooses its preimage in an unpredictable way: it applies a pseudo-random function $\text{PRF}_k$ to the key, and outputs the result $f_e(\text{PRF}_k(e))$.

Note that an adversary, given such an obfuscated circuit as auxiliary input $z$, can run it on the key $e$ for the extractable function and always obtain a proper image. The question is whether the extractor, given the same $(e, z)$, can output a preimage. Intuitively, had we given the extractor black-box access to the circuit $C_k$, instead of an obfuscation of $C_k$, it would have to invert the one-way function to obtain such a preimage.

If $z$ is a VBB obfuscation of $C_k$, the above could be translated to an actual proof; but is that also the case if we use indistinguishability obfuscation? When $z = iO(C_k)$, it is not as clear what kind of information leaks on the PRF key $k$. Nevertheless, we show that the above argument can still be salvaged. The idea is to consider an alternative to the the circuit $C_k$ that computes the same function, but without actually “knowing” $\text{PRF}_k(e)$. This is achieved using the puncturing technique of Sahai and Waters [SW13].

Specifically, instead of using any PRF family, we use a puncturable PRF. In such PRFs it is possible to puncture a given key $k$ at an arbitrary point $x^*$ in the domain of the function. The punctured function $\text{PRF}_{k,x^*}$, with punctured key $k_{x^*}$, preserves functionality at any other point, but hides any information on the point $\text{PRF}_k(x^*)$; namely, this value is pseudo-random, even given $(x^*, k_{x^*})$. As shown in several recent works [BW13, BGI13, KPTZ13], such puncturable PRFs follow from the [GGM86] construction.

Using a puncturable PRF in the implementation of $C_k$, we can now show that if the extractor succeeds in finding a preimage of $y = f_e(\text{PRF}_k(e))$, it would also succeed had we provided it with an obfuscation of the alternative circuit $C_{k,e,y}$. The circuit $C_{k,e,y}$ computes the same function as $C_k$, but in a different way: it only has the punctured key $k_e$, and has the value $y = f_e(\text{PRF}_k(e))$ directly hardwired into it, so that it does not have to evaluate the PRF in order to compute it. Thus, the fact that the extractor still succeeds follows by the guarantee of indistinguishability obfuscation. However, now by the pseudo-randomness guarantee at the punctured point $e$, we know that $\text{PRF}_k(e)$ is pseudo random, and thus the extractor can be used to invert the one-way function $f_e$ from scratch.

Finally, we note that since puncturable PRFs can be constructed from one-way functions, and any EOWF is in particular a OWF, it follows that the impossibility of EOWFs is implied by indistinguishability obfuscation without any further assumptions.

2 Definitions

We define extractable one-way functions, indistinguishability obfuscation, and puncturable pseudo-random functions.
2.1 Auxiliary-Input Extractable One-Way Functions

In this paper, we focus attention to extractable one-way functions. Our results directly extend to stronger extractable function primitives, such as extractable collision-resistant hashing, and extractable commitments.

**Definition 2.1 (Auxiliary-input EOWFs [CD08]).** Let $\ell, \ell', m$ be polynomially bounded length functions. An efficiently computable family of functions

$$\mathcal{F} = \left\{ f_e : \{0,1\}^{\ell(n)} \rightarrow \{0,1\}^{\ell'(n)} \mid e \in \{0,1\}^m(n), n \in \mathbb{N} \right\},$$

associated with an efficient (probabilistic) key sampler $K_F$, is an auxiliary-input extractable one-way function if it satisfies:

1. **One-wayness:** For PPT $A$, large enough security parameter $n \in \mathbb{N}$, and $z \in \{0,1\}^{\text{poly}(n)}$:

$$\Pr_{e \leftarrow K_F(1^n), x \leftarrow \{0,1\}^{\ell(n)}} \left[ x' \leftarrow A(e, f_e(x); z) \mid f_e(x') = f_e(x) \right] \leq \text{negl}(n).$$

2. **Extractability:** For any PPT adversary $A$, there exists a PPT extractor $E$ such that, for any large enough security parameter $n \in \mathbb{N}$, and advice $z \in \{0,1\}^{\text{poly}(n)}$:

$$\Pr_{e \leftarrow K_F(1^n)} \left[ y \leftarrow A(e; z) \wedge \exists x : f_e(x) = y \wedge f_e(x') \neq y \right] \leq \text{negl}(n).$$

**Remark 2.1 (On the auxiliary input).** For our results it is critical that the extractor $E$ receives the same auxiliary input $z$, which could be of arbitrary polynomial length, as $A$ 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 obfuscation. Additional motivation for this formulation appears in the introduction.

Still, one could consider weaker notions of extractability which may still suffice for some applications, and are not ruled out by our results, even assuming indistinguishability obfuscation.

- **Separate auxiliary inputs:** Here we only require that for any $A$ with non-uniform advice $z_a$, there exists an extractor with non-uniform advice $z'_e$, which may arbitrarily and inefficiently depend on $z_a$, and could be of an arbitrary polynomial size. This weaker notion may be useful in cases where the adversary’s auxiliary inputs do not depend on computations that may have taken place in the system before the extractable function is used. Examples include CCA and plaintext-aware encryption with non-uniform security reductions [Dam92, BP04b], and weak versions of 3-message zero-knowledge where the where the verifier doe not get auxiliary information and simulator is allowed to be more non-uniform than the verifier [HT98, BP04a].

- **Common but benign auxiliary input:** Here $A$ and $S$, in addition to arbitrary separate auxiliary-inputs $z_a$ and $z'_e$, respectively, get common auxiliary input $z$ as that id drawn from a specific distribution that is conjectured to be ‘benign’, in the sense that it is unlikely to encode a malicious obfuscation. For instance, the distribution can be uniform or an encryption of a random string. Examples where this is sufficient includes essentially all the works on succinct non-interactive arguments (SNARGs), succinct NIZKs, and targeted malleability that rely on extractable primitives [DCL08, Mie08, Gro10, GLR11, BSW12, BCCT12, BC12, DFH12, Lip12, BCCT13, BCI+13, GGPR13, Lip13].
• **Bounded auxiliary input**: Here there is a bound on the size of the auxiliary-input that the adversary may get. EOWFs according to this notion are constructed in [BCP13] from standard assumptions, and shown to suffice for 3-message arguments of knowledge and 2-message arguments that are bounded auxiliary-input zero-knowledge.

Finally, we remark that one may consider adversaries with both separate and common dynamic auxiliary input. That is, for any \( A \) and auxiliary input \( z_s \) there should exist \( S \) and auxiliary input \( z_s' \) such that (1) holds for any common auxiliary input \( z \). This notion is also ruled out by our techniques.

2.2 Indistinguishability Obfuscation

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

**Definition 2.2 (Indistinguishability obfuscation [BGI+01]).** A PPT algorithm \( iO \) is said to be an indistinguishability obfuscator (INDO) for \( C \), if it satisfies:

1. **Functionality:** For any \( C \in C \),
\[
\Pr_{iO} \left[ \forall x : iO(1^n, C)(x) = C(x) \right] = 1 .
\]

2. **Indistinguishability:** For any ensemble of circuit pairs \( \{(C_n^{(1)}, C_n^{(2)}) \in C \times C \}_{n \in \mathbb{N}} \), where the two circuits in each pair are of the same size and functionality, it holds that:
\[
\left\{ iO(1^n, C_n^{(1)}) \right\}_{n \in \mathbb{N}} \approx_c \left\{ iO(1^n, C_n^{(2)}) \right\}_{n \in \mathbb{N}} .
\]

For ease of notation, we shall often omit \( 1^n \) from the input to the obfuscator.

2.3 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 2.3 (Puncturable PRFs).** Let \( \ell, m \) be polynomially bounded length functions. An efficiently computable family of functions
\[
PRF = \left\{ PRF_k : \{0, 1\}^{m(n)} \rightarrow \{0, 1\}^{\ell(n)} \mid k \in \{0, 1\}^n, n \in \mathbb{N} \right\} ,
\]
associated with an efficient (probabilistic) key sampler \( K_{PRF} \), is a puncturable PRF if there exists a puncturing algorithm \( Punc \) that takes as input a key \( k \in \{0, 1\}^n \), and a point \( x^* \), and outputs a punctured key \( k_{x^*} \), so that the following conditions are satisfied:

1. **Functionality is preserved under puncturing:** For every \( x^* \in \{0, 1\}^{\ell(n)} \),
\[
\Pr_{k \leftarrow K_{PRF}(1^n)} \left[ \forall x \neq x^* : PRF_k(x) = PRF_{k_{x^*}}(x) \mid k_{x^*} = Punc(k, x^*) \right] = 1 .
\]

2. **Indistinguishability at punctured points:** The following ensembles are computationally indistinguishable:
• \{x^*, k_{x^*}, \text{PRF}_k(x^*) \mid k \leftarrow K_{\text{PRF}}(1^n), k_{x^*} = \text{Punc}(k, x^*) \}_{x^* \in \{0,1\}^m, n \in \mathbb{N}}

• \{x^*, k_{x^*}, u \mid k \leftarrow K_{\text{PRF}}(1^n), k_{x^*} = \text{Punc}(k, x^*), u \leftarrow \{0,1\}^{\ell(n)} \}_{x^* \in \{0,1\}^m, n \in \mathbb{N}} \}

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

3 From \(iO\) to Impossibility of Extractable Functions

We now show that if indistinguishability obfuscators exist, there do not exist EOWFs according to Definition 2.1. For this purpose, assuming the existence of an EOWF family \(F\), we shall describe an adversary \(A\) and a distribution \(Z\) on auxiliary inputs, such that any extractor fails, for auxiliary inputs sampled from \(Z\).

3.1 The Universal Adversary

We consider a universal PPT adversary \(A\) that given \((e, z) \in \{0,1\}^m \times \{0,1\}^{\text{poly}(n)}\), parses \(z\) as a circuit and returns \(z(e)\).

3.2 The Auxiliary Input Distribution.

Let \(F\) be a family of extractable one-way functions and let \(\text{PRF}\) be a puncturable pseudo-random function family. We start by defining two families of circuits

\[
\mathcal{C} = \left\{ C_k : \{0,1\}^m \rightarrow \{0,1\}^\ell(n) \mid k \in \{0,1\}^n, n \in \mathbb{N} \right\},
\]

\[
\mathcal{C}^* = \left\{ C_{k^*, y^*} : \{0,1\}^m \rightarrow \{0,1\}^\ell(n) \mid k \in \{0,1\}^n, n \in \mathbb{N} \right\}.
\]

The circuit \(C_k\), given a key \(e\) for an EOWF, applies \(\text{PRF}_k\) to \(e\), obtains an input \(x\), and returns the result of applying the EOWF \(f_e\) to \(x\).

**Figure 1:** The circuit \(C_k\).

The circuit \(C_{k^*, y^*}\), has a hardwired PRF key \(k_{e^*}\) that was derived from \(k\) by puncturing it at the point \(e^*\). In addition, it has hardwired an output \(y^*\) to replace the punctured result. In particular, when \(y^* = f_{e^*}(\text{PRF}_k(e^*))\) the circuit \(C_{k^*, y^*}\) computes the same function as \(C_k\).

We are now ready to define our auxiliary input distribution \(Z = \{Z_n\}_{n \in \mathbb{N}}\). Let \(s = s(n)\) be the maximal size of circuits in either \(\mathcal{C}\) or \(\mathcal{C}^*\), corresponding to security parameter \(n\), and denote by \([\mathcal{C}]_s\) a
**C\textsubscript{ke\textasciistars, y\textasciistars}**

**Hardwired:** a punctured PRF key \( k\textsubscript{e\textasciistars} = \text{Punc}(k, e\textsuperscript{*}) \) and \( y\textsuperscript{*} \in \{0, 1\}^{e'(n)} \).

**Input:** an EOWF key \( e \in \{0, 1\}^{m(n)} \).

1. If \( e \neq e\textsuperscript{*} \), compute \( x = \text{PRF}_{k\textsubscript{e\textasciistars}}(e) \), and return \( y = f_e(x) \).
2. If \( e = e\textsuperscript{*} \), return \( y\textsuperscript{*} \).

**Figure 2:** The circuit \( C\textsubscript{ke\textasciistars, y\textasciistars} \).

---

Circuit \( C \) padded with zeros to size \( s \). Let \( i\mathcal{O} \) be an indistinguishability obfuscator. The distribution \( Z\textsubscript{n} \) simply consists of an obfuscated (padded) circuit \( C\textsubscript{k} \).

**Z\textsubscript{n}**

1. Sample \( k \leftarrow \mathcal{K}_{\text{PREF}(1^n)} \).
2. Sample an obfuscation \( z \leftarrow i\mathcal{O}([C\textsubscript{k}]_s) \).
3. Output \( z \).

**Figure 3:** The auxiliary input distribution \( Z\textsubscript{n} \).

### 3.3 \( \mathcal{A} \) Does Not Have an Extractor

We next show that \( \mathcal{A} \) cannot have any extractor \( \mathcal{E} \) satisfying Definition 2.1. In fact, we show a stronger claim; namely, that for the auxiliary input distribution \( \mathcal{Z} \), any extractor fails with overwhelming probability.

**Proposition 3.1.** Let \( \mathcal{E} \) be any PPT candidate extractor for \( \mathcal{A} \) then

\[
\Pr_{e \leftarrow \mathcal{K}_{\text{PREF}(1^n)}} \left[ y \leftarrow \mathcal{A}(e; z) \implies \exists x : f_e(x) = y \land f_e(x') \neq y \right] \geq 1 - \text{negl}(n) .
\]

We note that, since the key \( e \) is sampled above independently of the auxiliary input \( z \), the above indeed disproves extractability.

**Proof of Proposition 3.1** First, we note that

\[
\Pr_{e \leftarrow \mathcal{K}_{\text{PREF}(1^n)}} \left[ y \leftarrow \mathcal{A}(e; z) \implies \exists x : f_e(x) = y \right] = 1 ;
\]

indeed, by the definition of \( \mathcal{A} \) and \( Z\textsubscript{n} \), and the correctness of \( i\mathcal{O} \),

\[
\mathcal{A}(e, z) = z(e) = C\textsubscript{k}(e) = f_e(\text{PRF}_{k}(e)) ,
\]

8
where \( C_k \in \mathcal{C} \) is the circuit obfuscated in \( z \), i.e. \( z = i\mathcal{O}([C_k]_s) \).

Now, assume towards contradiction that, for infinitely many \( n \in \mathbb{N} \), the extractor \( \mathcal{E} \) successfully outputs an image with noticeable probability \( \epsilon(n) \), i.e.

\[
\Pr_{e \leftarrow \mathcal{K}_{\mathcal{F}}(1^n)} \left[ x' \leftarrow \mathcal{E}(e; z) \right. \\
\left. \quad f_e(x') = z(e) = f_e(\text{PRF}_k(e)) \right] \geq \epsilon(n) ,
\]

where as before, \( z = i\mathcal{O}([C_k]_s) \).

Next, for every \( e^* \) we consider an alternative distribution \( Z_n(e^*, y^*) \) that, instead of sampling a circuit \( C_k \), samples a circuit \( C_{k^*}, y^* \), by first sampling \( k \) as usual, and then computing \( y^* = f_{e^*}(\text{PRF}_k(e^*)) \), and the punctured key \( k_{e^*} \). (Note that \( Z_n(e^*, y^*) \) is actually only parameterized by \( e^* \), we add \( y^* \) to the notation, to be more explicit.) We claim that the extractor still succeeds in finding a preimage, i.e.,

\[
\Pr_{e^* \leftarrow \mathcal{K}_{\mathcal{F}}(1^n)} \left[ x' \leftarrow \mathcal{E}(e^*; z^*) \right. \\
\left. \quad f_{e^*}(x') = z^*(e^*) = y^* = f_{e^*}(\text{PRF}_k(e^*)) \right] \geq \epsilon(n) - \text{negl}(n) .
\]

This follows from the fact that, for any \( e^* \) and \( k \), \( C_k \) and \( C_{k^*} \) compute the same function, and the \( i\mathcal{O} \) indistinguishability guarantee.

Next, we consider another experiment where \( Z_n(e^*, y^*) \) is altered to a new distribution \( Z_n(e^*, u) \) that, instead of sampling \( y^* = f_{e^*}(\text{PRF}_k(e^*)) \) in \( C_{k^*}, y^* \), samples \( y^* = f_{e^*}(u) \), for an independent random \( u \leftarrow \{0,1\}^\ell \). We claim that

\[
\Pr_{e^* \leftarrow \mathcal{K}_{\mathcal{F}}(1^n)} \left[ x' \leftarrow \mathcal{E}(e^*; z^*) \right. \\
\left. \quad f_{e^*}(x') = z^*(e^*) = y^* = f_{e^*}(u) \right] \geq \epsilon(n) - \text{negl}(n) ;
\]

indeed, this follows from the fact that \( \text{PRF}_k(e^*) \) is pseudo-random, even given the punctured key \( k_{e^*} \).

This means that \( \mathcal{E} \) can be used to break the one-wayness of \( \mathcal{F} \). Indeed, given a random key \( e^* \), and a challenge \( y^* = f_{e^*}(u) \), an inverter can simply sample a punctured \( k_{e^*} \) on its own, construct the circuit \( C_{k^*}, y^* \), with its challenge \( y^* \) hardwired in, and sample an obfuscation \( z^* \leftarrow i\mathcal{O}(C_{k^*}, y^*) \). Finally, it runs \( \mathcal{E}(e^*, z^*) \) to invert \( y^* \), with the same probability \( \epsilon(n) - \text{negl}(n) \).

\( \square \)

**Remark 3.1** (Separate vs. common auxiliary input). As mentioned in Remark 2.1, our proof also holds in the case that the extractor \( \mathcal{E} \) is allowed extra (separate) auxiliary input \( s \), which does not depend on the (common) auxiliary input \( z \) (provided that the EOWF is one-way against non-uniform adversaries).

Finally, we note that since puncturable PRFs can be constructed from one-way functions, and any EOWF is, in particular, a OWF, the impossibility of auxiliary-input EOWFs is implied by indistinguishability obfuscation without any further assumptions. Thus, Theorem 1.1 follows.

### References


