Limits of Extractability Assumptions with Distributional Auxiliary Input

Elette Boyle*
Cornell University
ecb227@cornell.edu

Rafael Pass†
Cornell University
rafael@cs.cornell.edu

November 4, 2013

Abstract

Extractability, or "knowledge," assumptions (such as the "knowledge-of-exponent" assumption) have recently gained popularity in the cryptographic community—leading to the study of primitives such as extractable one-way functions, extractable hash functions, succinct non-interactive arguments of knowledge (SNARKs), and extractable obfuscation, and spurring the development of a wide spectrum of new applications relying on these primitives. For most of these applications, it is required that the extractability assumption holds even in the presence of attackers receiving some *auxiliary information* that is sampled from some *fixed* efficiently computable distribution $\mathcal{Z}$.

We show that, assuming the existence of collision-resistant hash functions, there exists a pair of efficient distributions $\mathcal{Z}, \mathcal{Z}'$ such that either

- extractable one-way functions w.r.t. $\mathcal{Z}$ do not exist, or
- extractability obfuscations for Turing machines w.r.t. $\mathcal{Z}$ do not exist.

A corollary of this result shows that assuming existence of fully homomorphic encryption with decryption in NC^1, there exist efficient distributions $\mathcal{Z}, \mathcal{Z}'$ such that either

- extractability obfuscations for NC^1 wr.t. $\mathcal{Z}$ do not exist, or
- SNARKs for NP w.r.t. $\mathcal{Z}'$ do not exist.

To achieve our results, we develop a "succinct punctured program" technique, mirroring the powerful "punctured program" technique of Sahai and Waters (ePrint'13), and present several other applications of this new technique.

*Supported in part by AFOSR YIP Award FA9550-10-1-0093.

†Pass is supported in part by a Alfred P. Sloan Fellowship, Microsoft New Faculty Fellowship, NSF Award CNS-1217821, NSF CAREER Award CCF-0746990, NSF Award CCF-1214844, AFOSR YIP Award FA9550-10-1-0093, and DARPA and AFRL under contract FA8750-11-2- 0211. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the US Government.

1 Introduction

Extractability Assumptions. Extractability, or "knowledge," assumptions (such as the "knowledge-of-exponent" assumption), have recently gained in popularity, leading to the study of primitives such as extractable one-way functions, extractable hash-functions, SNARKs (succinct non-interactive arguments of knowledge), and extractable obfuscation:

- **Extractable OWF:** An extractable family of one-way (resp. collision-resistant) functions [Dam91, HT98, CD09], is a family of one-way (resp. collision-resistant) functions $\{f_i\}$ such that any attacker who outputs an element y in the range of a randomly chosen function f_i given the index i must "know" a pre-image x of y (i.e., $f_i(x) = y$). This is formalized by requiring for every adversary $\mathcal{A}$, the existence of an "extractor" $\mathcal{E}$ that (with overwhelming probability) given the view of $\mathcal{A}$ outputs a pre-image x whenever $\mathcal{A}$ outputs an element y in the range of the function.

  For example, the "knowledge-of-exponent" assumption of Damgard [Dam91] stipulates the existence of a particular such extractable one-way function.

- **SNARKs:** Succinct non-interactive arguments of knowledge (SNARKs) [Mic94, Val08, BCCT12] are communication-efficient (i.e., "short" or "succinct") arguments for NP with the property that if a prover generates an accepting (short) proof, it must "know" a corresponding (potentially long) witness for the statement proved, and this witness can be efficiently "extracted" out from the prover.

- **Extractability obfuscation:** [BGI$^+$12, BCP13] An extractability obfuscator $\mathcal{O}$ for a class of programs $\mathcal{C}$ is an efficient procedure which ensures if any efficient attacker $\mathcal{A}$ can distinguish obfuscations $\mathcal{O}(C_1)$ and $\mathcal{O}(C_2)$ of any two programs C_1, C_2 in the class $\mathcal{C}$, then it must "know" an input x such that $C_1(x) \neq C_2(x)$, and this input can be efficiently "extracted" (as above) from $\mathcal{A}$.

The above primitives have proven extremely useful in constructing cryptographic tools for which instantiations under complexity-theoretic hardness assumptions are not known (e.g., [HT98, BCCT12, GLR11, DFH12, BCP13]).

Extraction with (Distribution-Specific) Auxiliary Input. In all of these applications, we require a notion of an *auxiliary-input* extractable one-way function [HT98, CD09], where both the attacker and the extractor may receive an auxiliary input. The strongest formulation requires extractability in the presence of an *arbitrary* auxiliary input. Yet, as informally discussed already in the original work by Hada and Tanaka [HT98], extractability w.r.t. an arbitrary auxiliary input is an "overly strong" (or in the language of [HT98], "unreasonable") assumption. Indeed, a recent beautiful result of Bitansky, Canetti, Rosen and Paneth [BCPR13] (formalizing earlier intuitions from [HT98, BCCT12]) demonstrates that assuming the existence of indistinguishability obfuscators for the class of polynomial-size circuits[1] there cannot exist auxiliary-input extractable one-way functions that remain secure for an arbitrary auxiliary input.

However, for most of the above applications, we actually do not require extractability to hold w.r.t. an arbitrary auxiliary input; rather, as proposed by Bitansky et al [BCCT12, BCCT13], it

[1]The notion of indistinguishability obfuscation [BGI$^+$12] requires that obfuscations $\mathcal{O}(C_1)$ and $\mathcal{O}(C_2)$ of any two *equivalent* circuits C_1 and C_2 (i.e., whose outputs agree on all inputs) from some class $\mathcal{C}$ are computationally indistinguishable. A candidate construction for general-purpose indistinguishability obfuscation was recently given by Garg et al [GGH$^+$13].

often suffices to consider extractability with respect to specific distributions $\mathcal{Z}$ of auxiliary input.[2] More precisely, it would suffice to show that for every desired output length $\ell(\cdot)$ and distribution $\mathcal{Z}$ there exists a function family $\mathcal{F}_{\mathcal{Z}}$ (which, in particular, may be tailored for $\mathcal{Z}$) such that $\mathcal{F}_{\mathcal{Z}}$ is a family of extractable one-way (or collision-resistant) functions $\{0,1\}^k \to \{0,1\}^{\ell(k)}$ with respect to $\mathcal{Z}$. In fact, for some of these results (e.g., [BCCT12, BCCT13]), it suffices to just assume that extraction works for just for the *uniform* distribution.

In contrast, the result of [BCPR13] can be interpreted as saying that (assuming indistinguishability obfuscation), there do not exist extractable one-way functions with respect to *every* distribution of auxiliary input: That is, for every candidate extractable one-way function family $\mathcal{F}$, there exists *some* distribution $\mathcal{Z}_{\mathcal{F}}$ of auxiliary input that breaks it.

Our Results. In this paper, we show limitations of extractability primitives with respect to "distribution-specific" auxiliary input (assuming the existence of collision-resistant hash functions). Our main result shows a conflict between extractability obfuscation for Turing machines [BCP13] and extractable one-way functions.

Theorem 1.1 (Main Theorem)**.** *Assume the existence of collision-resistant hash functions. Then for every polynomial ℓ, there exist efficiently computable distributions $\mathcal{Z}$ and $\mathcal{Z}'$ such that one of the following two primitives does not exist:*

- *extractable one-way functions $\{0,1\}^k \to \{0,1\}^{\ell(k)}$ w.r.t. auxiliary input from $\mathcal{Z}$.*
- *extractability obfuscation for Turing machines w.r.t. auxiliary input from $\mathcal{Z}'$.*

If additionally we assume the existence of public-coin collision-resistant hash functions, the distribution $\mathcal{Z}'$ is uniform.

By combining our main theorem with results from [BCCT12] and [BCP13], we obtain the following corollary:

Theorem 1.2. *Assume the existence of fully homomorphic encryption with decryption in NC^1.*[3] *Then there exist efficiently computable distributions $\mathcal{Z}, \mathcal{Z}'$ such that one of the following two primitives does not exist:*

- *SNARKs w.r.t. auxiliary input from $\mathcal{Z}$.*
- *extractability obfuscation for NC^1 w.r.t. auxiliary input from $\mathcal{Z}'$.*

If additionally we assume the existence of public-coin collision-resistant hash function, the distribution $\mathcal{Z}'$ is uniform.

To prove our results, we develop a new proof technique, which we refer to as the "succinct punctured program" technique, extending the "punctured program" paradigm of Sahai and Waters [SW13]; see Section 1.1 for more details. This technique has several other interesting applications, as we discuss in Section 1.2.

[2] As far as we know, the only exceptions are in the context of zero-knowledge simulation, where the extractor is used in the simulation (as opposed to being used as part of a reduction), and we require simulation w.r.t. arbitrary auxiliary inputs. Nevertheless, as pointed out in the works on zero-knowledge [HT98, GS12], to acheive "plain" zero-knowledge [GMR89, BLV06] (where the verifier does not receive any auxiliary input), weaker "bounded" auxiliary input assumptions suffice.

[3] As is the case for nearly all existing FHE constructions (e.g., [GSW13, BV13]).

Interpretation of Our Results. Our results suggest that one must take care when making extractability assumptions, even in the presence of specific distributions of auxiliary inputs. In particular, we must develop a way to distinguish "good" distributions of auxiliary inputs (for which extractability assumptions may make sense) and "bad" ones (for which extractability assumptions are unlikely to hold). As mentioned above, for some applications of extractability assumptions, it in fact suffices to consider a particularly simple distribution of auxiliary inputs—namely the *uniform* distribution.[4] We emphasize that our results do not present any limitations of extractable one-way functions in the presence of uniform auxiliary input, and as such, this still seems like a plausible assumption at this point.

1.1 Proof Techniques

To explain our techniques, let us first explain earlier arguments against the plausibility of extractable one-way functions with auxiliary input. For simplicity of notation, we focus on extractable one-way function over $\{0,1\}^k \to \{0,1\}^k$ (as opposed to over $\{0,1\}^k \to \{0,1\}^{\ell(k)}$ for some polynomial ℓ), but emphasize that the approach described directly extends to the more general setting.

Early Intuitions. As mentioned above, already the original work of Hada and Tanaka [HT98], which introduced auxiliary input extractable one-way functions (EOWFs) (for the specific case of exponentiation), argued the "unreasonableness" of such functions, reasoning informally that the auxiliary input could contain a program that evaluates the function, and thus a corresponding extractor must be able to "reverse-engineer" *any* such program. Bitansky et al [BCCT12] made this idea more explicit: Given some candidate EOWF family $\mathcal{F}$, consider the distribution $\mathcal{Z}_{\mathcal{F}}$ over auxiliary input formed by "obfuscating" a program $\Pi^s(\cdot)$ for uniformly chosen s, where $\Pi^s(\cdot)$ takes as input a function index e from the alleged EOWF family $\mathcal{F} = \{f_i\}$, applies a pseudorandom function (PRF) with hardcoded seed s to the index i, and then outputs the evaluation $f_i(\mathsf{PRF}_s(i))$. Now, consider an attacker $\mathcal{A}$ who, given an index i, simply runs the obfuscated program to obtain a "random" point in the range of f_i. If it were possible to obfuscate Π^s in a "virtual black-box (VBB)" way (as in [BGI$^+$12]), then it easily follows that any extractor $\mathcal{E}$ for this particular attacker $\mathcal{A}$ can invert f_i. Intuitively, the VBB-obfuscated program hides the PRF seed s (revealing, in essence, only black-box access to Π^s), and so if $\mathcal{E}$ can successfully invert f_i on $\mathcal{A}$'s output $f_i(\mathsf{PRF}_s(i))$ on a pseudorandom input $\mathsf{PRF}_s(i)$, he must also be able to invert for a *truly* random input. Formally, given an index i and a random point y in the image of f_i, we can "program" the output of $\Pi^s(i)$ to simply be y, and thus E will be forced to invert y.

The problem with this argument is that (as shown by Barak et al [BGI$^+$12]), for large classes of functions VBB program obfuscation simply does not exist.

The Work of [BCPR13] and the "Punctured Program" Paradigm of [SW13]. Intriguingly, Bitansky, Canetti, Rosen and Paneth [BCPR13] show that by using a particular PRF and instead relying on indistinguishability obfuscation, the above argument still applies! To do so, they rely on the powerful "punctured-program" paradigm of Sahai and Waters [SW13] (and the closely related work of Hohenberger, Sahai and Waters [HSW13] on "instantiating random oracles"). Roughly speaking, the punctured program paradigm shows that if we use indistinguishability obfus-

[4] Note that this is not the case for all applications; for instance [HT98, GKP$^+$13, BGI13, GS12] require considering more complicated distributions.

cation to obfuscate a (function of) a special kind of "puncturable" PRF[5] [BW13, BGI13, KPTZ13], we can still "program" the output of the program on *one* input (which was used in [SW13, HSW13] to show various applications of indistinguishability obfuscation). Bitansky et al. [BCPR13] show that by using this approach, then from any alleged extractor $\mathcal{E}$ we can construct a one-way function inverter Inv by "programming" the output of the program Π^s at the input i with the challenge value y. More explicitly, mirroring [SW13, HSW13], they consider a hybrid experiment where $\mathcal{E}$ is executed with fake (but indistinguishable) auxiliary input, formed by obfuscating a *"punctured"* variant $\Pi^s_{i,y}$ of the program Π^s that contains an i-punctured PRF seed s^* (enabling evaluation of $\mathsf{PRF}_s(j)$ for any $j \neq i$) and directly outputs the hardcoded value $y := f_i(\mathsf{PRF}_s(i))$ on input i: indistinguishability of this auxiliary input follows by the security of indistinguishability obfuscation since the programs $\Pi^s_{i,y}$ and Π^s are equivalent when $y = f_i(\mathsf{PRF}_s(i)) = \Pi^s(i)$. In a second hybrid experiment, the "correct" hardcoded value y is replaced by a *random* evaluation $f_i(u)$ for uniform u; here, indistinguishability of the auxiliary inputs follows directly by the security of the punctured PRF. Finally, by indistinguishability of the three distributions of auxiliary input in the three experiments, it must be that $\mathcal{E}$ can extract an inverse to y with non-negligible probability in each hybrid; but, in the final experiment this implies the ability to invert a random evaluation, breaking one-wayness of the EOWF.

The Problem: Dependence on $\mathcal{F}$. Note that in the above approach, the auxiliary input distribution is selected *as a function* of the family $\mathcal{F} = \{f_j\}$ of (alleged) extractable one-way functions. Indeed, the obfuscated program Π^s must be able to evaluate f_j given j. One may attempt to mitigate this situation by instead obfuscating a universal circuit that takes as input both $\mathcal{F}$ and the index j, and appropriately evaluates f_j. But here still the *size* of the universal circuit must be greater than the running time of f_j, and thus such an auxiliary input distribution would only rule out EOWFs with a-priori bounded running time. This does not suffice for what we aim to achieve: in particular, it still leaves open the possibility that for every distribution of auxiliary inputs, there may exist a family of extractable one-way functions that remains secure for that particular auxiliary input distribution (although the running time of the extractable one-way function needs to be greater than the length of the auxiliary input).

A First Idea: Using Turing Machine Obfuscators. At first sight, it would appear this problem could be solved if we could obfuscate *Turing machines*. Namely, by obfuscating a universal Turing machine in the place of a universal circuit in the construction above, the resulting program Π^s would depend only on the size of the PRF seed s, and not on the runtime of $f_j \in F$. Indeed, recently [BCP13] showed candidate constructions of obfuscators for Turing machines, based on the existence of extractability obfuscators for NC^1, fully homomorphic encryption with decryption in NC^1, and so-called P-certificates.[6]

But there is a catch. To rely on the punctured program paradigm, we must be able to obfuscate the program Π^s in such a way that the result is indistinguishable from an obfuscation of a related "punctured" program $\Pi^s_{i,y}$; in particular, the *size* of the obfuscation must be at least as large as $|\Pi^s_{i,y}|$. Whereas the size of Π^s is now bounded by a polynomial in the size of the PRF seed s, the description of this punctured program must specify a punctured input i (corresponding to an index of the candidate EOWF $\mathcal{F}$) and hardcoded output value y, and hence must grow with the size of $\mathcal{F}$.

[5] That is, a PRF where we can surgically remove one point in the domain of the PRF, keeping the rest of the PRF intact, and yet, even if we are given the seed of the punctured PRF, the value of the original PRF on the surgically removed point remains computationally indistinguishable from random.

[6] That is, succinct non-interactive arguments for P in the Common Reference String Model.

We thus run into a similar wall: even with obfuscation of Turing machines, the resulting auxiliary input distribution $\mathcal{Z}$ would only rule out EOWF with a-priori bounded index length.

Our "Succinct Punctured Program" Technique. To deal with this issue, we develop a "succinct punctured program" technique. That is, we show how to make the size of the obfuscation be independent of the length of the input, while still retaining its usability as an obfuscator. The idea is two-fold: First, we modify the program Π^s to *hash* the input to the PRF, using a collision-resistant hash function h. That is, we now consider a program $\Pi^{h,s}(j) = f_j(PRF_s(h(j)))$. Second, we make use of *extractability obfuscation*, as opposed to just indistinguishability obfuscation. Specifically, our constructed auxiliary input distribution $\mathcal{Z}$ will sample a uniform s and a random hash function h (from some appropriate collection of collision-resistant hash functions) and then output an extractability obfuscation of $\Pi^{h,s}$.

To prove that this "universal" distribution $\mathcal{Z}$ over auxiliary input breaks *all* alleged extractable one-way functions over $\{0,1\}^k \to \{0,1\}^k$, we define a one-way function inverter Inv just as before, except that we now feed the EOWF extractor $\mathcal{E}$ the obfuscation of the "punctured" variant $\Pi^{h,s}_{i,y}$ which contains a PRF seed punctured at point $h(i)$. The program $\Pi^{h,s}_{i,y}$ proceeds just as $\Pi^{h,s}$ except on all inputs j such that $h(j)$ is equal to this special value $h(i)$; for those inputs it simply outputs the hardcoded value y. (Note that the index i is no longer needed to specify the function $\Pi^{h,s}_{i,y}$—rather, just its hash $h(i)$—but is included for notational convenience). As before, consider a hybrid experiment where y is selected as $y := \Pi^{h,s}(i)$.

Whereas before the punctured program was equivalent to the original, and thus indistinguishability of auxiliary inputs in the different experiments followed by the definition of indistinguishability obfuscation, here it is *no longer* the case that if $y = \Pi^{h,s}(i)$, then $\Pi^{h,s}_{i,y}$ is equivalent to $\Pi^{h,s}$—in fact, they may differ on many points. More precisely, the programs may differ in all points j such that $h(j) = h(i)$, but $j \neq i$ (since f_j and f_i may differ on the input $PRF_s(h(i))$). Thus, we can no longer rely on indistinguishability obfuscation to provide indistinguishability of these two hybrids.

We resolve this issue by relying *extractability obfuscation* instead of just indistinguishability obfuscation. Intuitively, if obfuscations of $\Pi^{h,s}$ and $\Pi^{h,s}_{i,y}$ can be distinguished when y is set to $\Pi^{h,s}(i)$, then we can efficiently recover some input j where the two programs differ. But, by construction, this must be some point j for which $h(j) = h(i)$ (or else the two program are the same), *and* $j \neq i$ (since we chose the hardcoded value $y = \Pi^{h,s}(i)$ to be consistent with $\Pi^{h,s}$ on input i. Thus, if the obfuscations can be distinguished, we can find a collision in h, contradicting its collision resistance.

To formalize this argument, we require a notion of extractability obfuscation with auxiliary input. As it turns out, we only require extractability to hold with respect to the distribution $\mathcal{Z}'$ that samples a random hash function h from the CRHF family. We have thus achieved our goal of demonstrating two distributions $\mathcal{Z}, \mathcal{Z}'$ such that either extractable one-way functions $\{0,1\}^k \to \{0,1\}^k$ do not exist w.r.t. $\mathcal{Z}$, or extractability obfuscators do not exists w.r.t. $\mathcal{Z}'$. Finally, note that assuming the existence *public-coin* collision-resistant hash functions [HR04], then $\mathcal{Z}'$ can be uniform.

1.2 Other Applications of the "Succinct Punctured Program" Technique

As mentioned above, the "punctured program" paradigm of [SW13] has been used in multiple applications [SW13, HSW13, GGHR13, BZ13]. Many of them rely on punctured programs in an essentially identical way to the approach described above, and in particular follow the same hybrids. Furthermore, for some of these applications, there are significant gains in making the

obfuscation succinct (i.e., independent of the input size of the obfuscated program). Thus, for these applications, if we instead rely on extractability obfuscations (and the existence of collision-resistant hash functions), by using our succinct punctured program technique, we can obtain significant improvements. For instance, by relying on the same approach as above, we can show based on these assumptions:

- "Succinct" Perfect Zero-Knowledge Non-Interactive *Universal* Argument System (with communication complexity k^ϵ for every ϵ), by relying on the non-succinct Perfect NIZK construction of [SW13].
- A *universal* instantiation of Random Oracles, for which the Full Domain Hash (FDH) signature paradigm [BR93] is (selectively) secure for *every* trapdoor (one-to-one) function (if hashing not only the message but also the index of the trapdoor function), by relying on the results of [HSW13] showing how to provide a trapdoor-function specific instantiation of the random oracle in the FDH.[7]

We explore these applications further in Section 4.

2 Preliminaries

2.1 Extractability Obfuscation

For each (possibly non-uniform) distribution $\mathcal{Z} = \{\mathcal{Z}_k\}$, we define extractability obfuscation secure with respect to auxiliary input from $\mathcal{Z}$.

We present a definition as formalized in [BCP13], which is a variant of the notion of "differing-inputs" obfuscation of [BGI$^+$12].[8] We remark that the notion of differing-inputs obfuscation is stronger, and thus also ruled out by our negative result.

Definition 2.1 ($\mathcal{Z}$-Auxiliary-Input Extractability Obfuscator)**.** (Variant of [BGI$^+$12]) A uniform PPT machine $e\mathcal{O}$ is an *extractability obfuscator* for a class of Turing machines $\{\mathcal{M}_k\}_{k\in\mathbb{N}}$ if the following conditions are satisfied:

- **Correctness:** There exists a negligible function $\mathsf{negl}(k)$ such that for every security parameter $k \in \mathbb{N}$, for all $M \in \mathcal{M}_k$, for all inputs x, we have

$$\Pr[\tilde{M} \leftarrow e\mathcal{O}(1^k, M) : M'(x) = M(x)] = 1 - \mathsf{negl}(k).$$

- **Security:** For every non-uniform PPT adversary $\mathcal{A}$ and polynomial $p(k)$, there exists a non-uniform PPT extractor E and polynomial $q(k)$ such that the following holds. For every $k \in \mathbb{N}$, every pair of Turing machines $M_0, M_1 \in \mathcal{M}_k$,

$$\Pr\left[z \leftarrow \mathcal{Z}_k;\ b \leftarrow \{0,1\};\ \tilde{M} \leftarrow e\mathcal{O}(1^k, M_b) : \mathcal{A}(1^k, \tilde{M}, M_0, M_1, z) = b\right] \geq \frac{1}{2} + \frac{1}{p(k)} \quad (1)$$

$$\implies \quad \Pr\left[z \leftarrow \mathcal{Z}_k;\ w \leftarrow E(1^k, M_0, M_1, z) : M_0(w) \neq M_1(w)\right] \geq \frac{1}{q(k)}. \quad (2)$$

[7] That is, [HSW13] shows that for every trapdoor one-to-one function, there exists some way to instantiate the random oracle so that the resulting scheme is secure. In constrast, our results shows that there exists a single instantation that works no matter what the trapdoor function is.

[8] Formally, our notion of extractability obfuscation departs from differing-inputs obfuscation of [BGI$^+$12] in two ways: First, [BGI$^+$12] require the extractor E to extract a differing input for M_0, M_1 given *any* pair of programs M_0', M_1' evaluating equivalent functions. Second, [BGI$^+$12] consider also adversaries who distinguish with negligible advantage $\epsilon(k)$, and require that extraction still succeeds in this setting, but within time polynomial in $1/\epsilon$. In contrast, we restrict our attention only to adversaries who succeed with noticeable advantage.

Definition 2.2 (Extractability Obfuscator for TM)**.** A uniform PPT machine $e\mathcal{O}_{\mathsf{TM}}$ is called an *extractability obfuscator for the class* TM *of polynomial-size Turing machines* if it satisfies the following. For each k, let $\mathcal{M}_k$ be the class of Turing machines Π containing a description of a Turing machine M of size bounded by k, such that Π takes two inputs, (t, x), with $|t| = k$, and the output of $\Pi(t, x)$ is defined to be the the output of running the Turing machine $M(x)$ for t steps. Then $e\mathcal{O}_{\mathsf{TM}}$ is an extractability obfuscator for $\{\mathcal{M}_k\}$.

Note that applying the properties of extractability obfuscation to this class of Turing machines $\{\mathcal{M}_k\}$ implies that for programs $\Pi_0, \Pi_1 \in \mathcal{M}_k$ defined above (corresponding to underlying size-k Turing machines M_0, M_1), efficiently distinguishing between obfuscations of Π_0 and Π_1 implies that one can efficiently extract an input *pair* (t', x') for which $\Pi_0(t', x') \neq \Pi_1(t', x')$. In particular, either $M_0(x') \neq M_1(x')$ or $Runtime(M_0, x') \neq Runtime(M_1, x)$. Thus, if restricting attention to a subclass of $\mathcal{M}_k$ for which each pair of programs satisfies $Runtime(M_0, x) = Runtime(M_1, x)$ for each input x, then "standard" extraction is guaranteed (i.e., such that the extracted input contains x' satisfying $M_0(x') \neq M_1(x')$).

In the sequel, when referring to an extractability obfuscation of a Turing machine M, we will implicitly mean the related program Π_M as above, but will suppress notation of the additional input t. For our application, it will be the case for the relevant class of Turing machines that every pair of programs M_0, M_1 has same runtime per input (and thus we will achieve "standard" input-extraction guaranteed).

2.2 Extractable One-Way Functions

For this work, we consider a slightly weakened version of EOWFs with respect to distributional auxiliary input information, from some distribution $\mathcal{Z}$. Namely, we require one-wayness and extractability to hold with overwhelming probability over auxiliary input z sampled from $\mathcal{Z}$. In contrast, typical definitions require these properties to hold for *any* auxiliary input z in some auxiliary input set $\mathcal{Z}$ (e.g., *all* length-bounded values). For example, "standard" auxiliary-input-secure EOWFs [CD08] are required to be $\mathcal{Z}$-auxiliary-input EOWFs for *every* (possibly non-uniform) distribution $\mathcal{Z}$.

We present a non-uniform version of the definition, in which both one-wayness and extractability are with respect to *non-uniform* polynomial-time adversaries.

Definition 2.3 ($\mathcal{Z}$-Auxiliary-Input EOWF)**.** Let ℓ, m be polynomially bounded length functions. An efficiently computable family of functions

$$\mathcal{F} = \left\{ f_i : \{0,1\}^k \to \{0,1\}^{\ell(k)} \;\middle|\; i \in \{0,1\}^{m(k)}, k \in \mathbb{N} \right\},$$

associated with an efficient probabilistic key sampler $\mathcal{K}_{\mathcal{F}}$, is a *$\mathcal{Z}$-auxiliary-input extractable one-way function* if it satisfies:

- **One-wayness:** For non-uniform polynomial-time $\mathcal{A}$ and sufficiently large security parameter $k \in \mathbb{N}$,

$$\Pr\left[z \leftarrow \mathcal{Z}_k;\ i \leftarrow \mathcal{K}_{\mathcal{F}}(1^k);\ x \leftarrow \{0,1\}^k;\ x' \leftarrow \mathcal{A}(i, f_i(x); z) : f_i(x') = f_i(x) \right] \leq \mathsf{negl}(k).$$

- **Extractability:** For any non-uniform polynomial-time adversary $\mathcal{A}$, there exists a non-uniform polynomial-time extractor $\mathcal{E}$ such that, for sufficiently large security parameter $k \in \mathbb{N}$:

$$\Pr\left[z \leftarrow \mathcal{Z}_k;\ i \leftarrow \mathcal{K}_{\mathcal{F}}(1^k);\ y \leftarrow \mathcal{A}(i; z);\ x' \leftarrow \mathcal{E}(i; z) : \exists x \text{ s.t. } f_i(x) = y \wedge f_i(x') \neq y \right] \leq \mathsf{negl}(k).$$

2.3 Succinct Non-Interactive Arguments (SNARGs and SNARKs)

We focus attention to *publicly verifiable* succinct arguments.

We consider succinct non-interactive arguments *of knowledge* (SNARKs) with adaptive soundness in Section 3.2, and consider the case of specific distributional auxiliary input.

Definition 2.4 ($\mathcal{Z}$-Auxiliary Input Adaptive SNARK)**.** A triple of algorithms $(\mathsf{CRSGen}, \mathsf{Prove}, \mathsf{Verify})$ is a *publicly verifiable, adaptively sound succinct non-interactive argument of knowledge (SNARK)* for the relation $\mathcal{R}$ if the following conditions are satisfied for security parameter k:

- **Completeness:** For any $(x, w) \in \mathcal{R}$,

$$\Pr[\mathsf{crs} \leftarrow \mathsf{CRSGen}(1^k); \pi \leftarrow \mathsf{Prove}(x, w, \mathsf{crs}) : \mathsf{Verify}(x, \pi, \mathsf{crs}) = 1] = 1.$$

  In addition, $\mathsf{Prove}(x, w, \mathsf{crs})$ runs in time $\mathsf{poly}(k, |y|, t)$.

- **Succinctness:** The length of the proof π output by $\mathsf{Prove}(x, w, \mathsf{crs})$, as well as the running time of $\mathsf{Verify}(x, \pi, \mathsf{crs})$, is bounded by $p(k + |X|)$, where p is a universal polynomial that does not depend on $\mathcal{R}$. In addition, $\mathsf{CRSGen}(1^k)$ runs in time $\mathsf{poly}(k)$: in particular, crs is of length $\mathsf{poly}(k)$.

- **Adaptive proof of knowledge:** For any non-uniform polynomial-size prover P^* there exists a non-uniform polynomial-size extractor $\mathcal{E}_{P^*}$, such that for all sufficiently large $k \in \mathbb{N}$ and auxiliary input $z \leftarrow \mathcal{Z}$, it holds that

$$\Pr[z \leftarrow \mathcal{Z};\ \mathsf{crs} \leftarrow \mathsf{CRSGen}(1^k);\ (x, \pi) \leftarrow P^*(z, \mathsf{crs});\ (x, w) \leftarrow \mathcal{E}_{P^*}(z, \mathsf{crs}) : \mathsf{Verify}(\mathsf{crs}, x, \pi) = 1 \wedge w \notin R(x)] \leq \mathsf{negl}(k).$$

We also consider the following notion of zero-knowledge (ZK) succinct non-interactive arguments (SNARGs) without the extraction property, in Section 4.1.

Definition 2.5 (Perfect ZK-SNARG)**.** A triple of algorithms $(\mathsf{CRSGen}, \mathsf{Prove}, \mathsf{Verify})$ is a *publicly verifiable perfect zero-knowledge (ZK) succinct non-interactive argument (SNARG)* for the relation $\mathcal{R}$ (corresponding to language L) if it satisfies the Correctness and Succinctness properties as in Definition 2.4, in addition to the following properties:

- **(Non-Adaptive) Soundness:** For every PPT P^*, for every $x \notin L$, it holds that

$$\Pr[\mathsf{crs} \leftarrow \mathsf{CRSGen}(1^k);\ \pi \leftarrow P^*(1^k, x, \mathsf{crs}) : \mathsf{Verify}(x, \pi, \mathsf{crs}) = 1] \leq \mathsf{negl}(k).$$

- **Perfect Zero Knowledge:** There exist PPT algorithms $\mathcal{S} = (\mathcal{S}^{\mathsf{crs}}, \mathcal{S}^{\mathsf{Proof}})$ such that for any polynomial collection $(x_i, w_i) \in \mathcal{R}$, $i \in [\ell(k)]$, the following distributions are *identical*:

$$\{\mathsf{crs} \leftarrow \mathsf{CRSGen}(1^k);\ \pi_1 \leftarrow \mathsf{Prove}(x_1, w_1, \mathsf{crs}); \cdots;\ \pi_\ell \leftarrow \mathsf{Prove}(x_\ell, w_\ell, \mathsf{crs}) : (\mathsf{crs}, \pi_1, \ldots, \pi_\ell)\}.$$

$$\{(\mathsf{crs}^{\mathsf{sim}}, \mathsf{td}) \leftarrow \mathcal{S}^{\mathsf{crs}}(1^k);\ \pi_1^{\mathsf{sim}} \leftarrow \mathcal{S}^{\mathsf{Proof}}(x_1, \mathsf{crs}, \mathsf{td}); \cdots;\ \pi_\ell^{\mathsf{sim}} \leftarrow \mathcal{S}^{\mathsf{Proof}}(x_\ell, \mathsf{crs}, \mathsf{td}) : (\mathsf{crs}^{\mathsf{sim}}, \pi_1^{\mathsf{sim}}, \ldots, \pi_\ell^{\mathsf{sim}})\}.$$

Definition 2.6 ((Perfect ZK) Universal Arguments)**.** [BG08] We say that $(\mathsf{CRSGen}, \mathsf{Prove}, \mathsf{Verify})$ is a *(perfect zero-knowledge) universal argument* if it is a (Perfect ZK) SNARG for the universal relation R_U, defined to be the set of instance-witness pairs (y, w), where $y = (M, x, t), |w| \leq t$, and M is a Turing machine, such that M accepts (x, w) after at most t steps.

Note that while the witness w for each instance $y = (M, x, t)$ in the relation R_U is of size at most t, there is no a-priori polynomial bounding t in terms of $|x|$.

2.4 Puncturable PRFs

Our result makes use of puncturable PRFs, which are PRFs that can be defined on all bit strings of a certain length, except for any polynomial-size set of inputs. We focus on the simple case of puncturing PRFs at a single point. The definition is formulated as in [SW13], following the specific exposition of [BCPR13].

Definition 2.7 (Puncturable PRFs)**.** Let m', ℓ be polynomially bounded length functions. An efficiently computable family of functions

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

associated with an efficient (probabilistic) seed sampler $\mathcal{K}_{\mathcal{PRF}}$, is a puncturable PRF if there exists a puncturing algorithm Punct that takes as input a seed $s \in \{0,1\}^k$ and a point $x^* \in \{0,1\}^{m'(k)}$, and outputs a punctured seed s_{x^*}, so that the following conditions are satisfied:

- **Functionality is preserved under puncturing:** For every $x^* \in \{0,1\}^{m'(k)}$,

$$\Pr\left[s \leftarrow \mathcal{K}_{\mathcal{PRF}}(1^k); s_{x^*} \leftarrow \mathsf{Punct}(k, x^*) : \forall x \neq x^*,\ \mathsf{PRF}_s(x) = \mathsf{PRF}_{s_{x^*}}(x) \right] = 1.$$

- **Indistinguishability at punctured points:** The following ensembles are computationally indistinguishable:

$$\{ s \leftarrow \mathcal{K}_{\mathcal{PRF}}(1^k), s_{x^*} \leftarrow \mathsf{Punct}(s, x^*) : x^*, s_{x^*}, \mathsf{PRF}_s(x^*) \}_{x^* \in \{0,1\}^{m(k)}, k \in \mathbb{N}}$$

$$\{ s \leftarrow \mathcal{K}_{\mathcal{PRF}}(1^k), s_{x^*} \leftarrow \mathsf{Punct}(s, x^*), u \leftarrow \{0,1\}^{\ell(k)} : x^*, s_{x^*}, u \}_{x^* \in \{0,1\}^{m(k)}, k \in \mathbb{N}}.$$

Note that the definition is "selectively secure," where x^* is specified before sampling the public parameters. For notational simplicity, (and without loss of generality), we will assume the punctured key s_{x^*} explicitly includes x^* in the clear.

As observed in [BW13, BGI13, KPTZ13], the GGM tree-based PRF construction [GGM86] yields puncturable PRFs as defined above, based on any one-way function. The size of such a punctured key s_{x^*} in this construction is $O(m'(k) \cdot \ell(k))$ (specifically, a punctured key at input $x^* = x_1 x_2 \cdots x_{m'(k)}$ can be attained by providing $m'(k)$ size-$\ell(k)$ partial evaluations in the GGM tree, corresponding to prefixes $(\bar{x}_1), (x_1 \bar{x}_2), \ldots, (x_1 x_2 \cdots \bar{x}_{m'(k)})$.)

Theorem 2.8 ([BW13, BGI13, KPTZ13])**.** *If one-way functions exist, then for all efficiently computable functions $m'(k)$ and $\ell(k)$, there exists a puncturable PRF family that maps $m'(k)$ bits to $\ell(k)$ bits, such that the size of a punctured key is $O(m'(k) \cdot \ell(k))$.*

3 Extractability Obfuscation or Extractable One-Way Functions

3.1 From $\mathcal{Z}'$-Auxiliary-Input $e\mathcal{O}$ to Impossibility of $\mathcal{Z}$-Auxiliary-Input EOWF

We demonstrate bounded polynomial-time uniformly samplable distributions $\mathcal{Z}, \mathcal{Z}'$ (with bounded poly-size output length) such that if there exists extractability obfuscation for Turing machines with respect to auxiliary input from distribution $\mathcal{Z}'$, and there exist collision-resistant hash functions (CRHF), then there do *not* exist extractable one-way functions (EOWF) with respect to auxiliary information sampled from distribution $\mathcal{Z}$. In our construction, $\mathcal{Z}$ consists of an obfuscated Turing

machine, and $\mathcal{Z}'$ is precisely the distribution of CRHF function descriptions. In particular, if there exist CRHFs whose indices are random (e.g., public-coin CRHFs), then $\mathcal{Z}'$ can be *uniform*.

We emphasize that we provide a single distribution $\mathcal{Z}$ of auxiliary inputs for which *all* candidate EOWF families $\mathcal{F}$ with given output length will fail. This is in contrast to the result of [BCPR13], which show for each candidate family $\mathcal{F}$ that there exists a tailored distribution $\mathcal{Z}_{\mathcal{F}}$ (whose size grows with $|\mathcal{F}|$) for which $\mathcal{F}$ will fail.

Theorem 3.1. *For every polynomial ℓ, there exist efficient, uniformly samplable distributions $\mathcal{Z}, \mathcal{Z}'$ such that, assuming the existence of collision-resistant hash functions and $\mathcal{Z}'$-auxiliary input extractability obfuscation for Turing machines, then there* cannot *exist $\mathcal{Z}$-auxiliary-input extractable one-way functions $\{f_i : \{0,1\}^k \to \{0,1\}^{\ell(k)}\}$.*

Proof. We construct an adversary $\mathcal{A}$ and desired distribution $\mathcal{Z}$ on auxiliary inputs, such that for any alleged EOWF family $\mathcal{F}$, there cannot exist an efficient extractor corresponding to $\mathcal{A}$ given auxiliary input from $\mathcal{Z}$ (assuming existence of the listed tools).

The Universal Adversary $\mathcal{A}$. We consider a universal PPT adversary $\mathcal{A}$ that, given $(i, z) \in \{0,1\}^{\mathsf{poly}(k)} \times \{0,1\}^{n(k)}$, parses z as a Turing machine and returns $z(i)$. Note that in our setting, i corresponds to the index of the selected function $f_i \in \mathcal{F}$, and (looking ahead) the auxiliary input z will contain an obfuscated program.

The Auxiliary Input Distribution $\mathcal{Z}$. Let $\mathcal{PRF} = \{\mathsf{PRF}_s : \{0,1\}^{m(k)} \to \{0,1\}^k\}_{s \in \{0,1\}^k}$ be a puncturable pseudorandom function family, and $\mathcal{H} = \{\mathcal{H}_k\}$ a collision-resistant hash function family with $h : \{0,1\}^* \to \{0,1\}^{m(k)}$ for each $h \in \mathcal{H}_k$. (Note that by Theorem 2.8, punctured PRFs for these parameters exist based on OWFs, which are implied by CRHF). We begin by defining two classes of *Turing machines*:

$$\mathcal{M} = \left\{ \Pi^{h,s} \;\middle|\; s \in \{0,1\}^k,\ h \in \mathcal{H}_k,\ k \in \mathbb{N} \right\},$$

$$\mathcal{M}^* = \left\{ \Pi^{h,s}_{i,y} \;\middle|\; s \in \{0,1\}^k,\ y \in \{0,1\}^{\ell(k)},\ h \in \mathcal{H}_k,\ k \in \mathbb{N} \right\},$$

which we now describe. We assume without loss of generality for each k that the corresponding collection of Turing machines $\Pi^{h,s} \in \mathcal{M}_k$, $\Pi^{h,s}_{i,y} \in \mathcal{M}^*_k$ are of the *same size*; this can be achieved by padding. (We address the size bound of each class of machines below). In a similar fashion, we may further assume that for each k the runtime of each $\Pi^{h,s}$ and $\Pi^{h,s}_{i,y}$ on any given input f_i is equal (see discussion in Section 2.1).

At a high level, each machine $\Pi^{h,s}$ accepts as input a poly-size circuit description of a function f_i (with canonical description, including a function index i), computes the hash of the corresponding index i with respect to the hardcoded hash function h, applies a PRF with hardcoded seed s to the hash, and then evaluates the circuit f_i on the resulting PRF output value x: that is, $\Pi^{h,s}_{i,y}(f_i)$ outputs $U_k(f_i, \mathsf{PRF}_s(h(i)))$, where U_k is the universal Turing machine. See Figure 1. Note that each $\Pi^{h,s}$ can be described by a Turing machine of size $O(|s| + |h| + U_k|)$, which is bounded by $p(k)$ for some fixed polynomial p.

The machines $\Pi^{h,s}_{i,y}$ perform a similar task, except that instead of having the entire PRF seed s hardcoded, they instead only have a *punctured* seed s^* derived from s by puncturing it at the point $h(i)$ (i.e., enabling evaluation of the PRF on all points except $h(i)$). In addition, it has hardwired an output y to replace the punctured result. More specifically, on input a circuit description f_j (with explicitly specified index j), the program $\Pi^{h,s}_{i,y}$ will first compute the hash $h = h(j)$, continue

Turing Machine $\Pi^{h,s}$:

Hardwired: Hash function $h : \{0,1\}^* \to \{0,1\}^{m(k)}$, PRF seed $s \in \{0,1\}^k$.

Inputs: Circuit description f_i

1. Hash the index: $v = h(i)$.
2. Compute the PRF on this hash: $x = \mathsf{PRF}_s(v)$.
3. Evaluate the universal Turing machine on inputs f_i, x: i.e., $y = U_k(f_i, x)$.
4. Output y.

Figure 1: Turing machines $\Pi^{h,s} \in \mathcal{M}$.

Auxiliary Input Distribution $\mathcal{Z}_k$:

1. Sample a hash function $h \leftarrow \mathcal{H}_k$.
2. Sample a PRF seed $s \leftarrow \mathcal{K}_{\mathcal{PRF}}(1^k)$.
3. Output an obfuscation $\tilde{\Pi} \leftarrow e\mathcal{O}(\Pi^{h,s})$.

Figure 2: The auxiliary input distribution $\mathcal{Z}_k$.

the computation as usual for any $h \neq h(i)$ using the punctured PRF key, and for $h = h(i)$, it will skip the PRF and U_k evaluation steps and directly output y. Note that because the hash function h is not injective, this puncturing may change the value of the program on multiple inputs f_j (corresponding to functions $f_j \in \mathcal{F}$ with $h(j) = h(i)$). When the hardcoded value y is chosen to be $y = f_i(\mathsf{PRF}_s(h(i)))$, the Turing machine $\Pi^{h,s}_{i,y}$ agrees with $\Pi^{h,s}$ additionally on the input f_i, but not necessarily on the other inputs f_j for which $h(j) = h(i)$. (Indeed, whereas the hash of their indices collide, and thus their corresponding PRF outputs, $\mathsf{PRF}(h(j))$, will agree, the final step will apply *different* functions f_j to this value). We remark that indistinguishability obfuscation arguments will thus not apply to this scenario, since we are modifying the computed functionality. In contrast, extractability obfuscation will guarantee that the two obfuscated programs are indistinguishable, otherwise we can efficiently *find* one of the disagreeing inputs, which will correspond to a collision in the CRHF.

Note that each $\Pi^{h,s}_{i,y}$ can be described by a Turing machine of size $O(|s^*| + |h| + |y| + |U_k|)$. Recall by Theorem 2.8 the size of the punctured PRF key $|s^*| \in O(m'(k)\ell(k))$, where the PRF has input and output lengths $m'(k)$ and $\ell(k)$. In our application, note that the input to the PRF is not the function index i itself (in which case the machine $\Pi^{h,s}_{i,y}$ would need to grow with the size of the alleged EOWF family), but rather the *hashed* index $h(i)$, which is of fixed polynomial length. Thus, collectively, we have $|\Pi^{h,s}_{i,y}|$ is bounded by a fixed polynomial $p'(k)$, and finally that there exists a single fixed polynomial bound on the size of *all* programs $\Pi^{h,s} \in \mathcal{M}, \Pi^{h,s}_{i,y} \in \mathcal{M}^*$. This completely determines the defined auxiliary input distribution $\mathcal{Z} = \{\mathcal{Z}_k\}$, described in full in Figure 2. (Note that the size of the auxiliary output generated by $\mathcal{Z}$, which corresponds to an obfuscation of an appropriately padded program $\Pi^{h,s}$ is thus also bounded by a fixed polynomial in k).

$\mathcal{A}$ Has No Extractor. We show that, based on the assumed security of the underlying tools, the constructed adversary $\mathcal{A}$ given auxiliary input from the constructed distribution $\mathcal{Z} = \{Z_k\}$, cannot have an extractor $\mathcal{E}$ satisfying Definition 2.3:

Proposition 3.2. *For any non-uniform polynomial-time candidate extractor $\mathcal{E}$ for $\mathcal{A}$, it holds that*

Turing Machine $\Pi^{h,s}_{i,y}$:

Hardwired: Hash function $h : \{0,1\}^* \to \{0,1\}^{m(k)}$, punctured PRF seed $s^* \in \{0,1\}^k$, punctured point $h(i)$, bit string $y \in \{0,1\}^{\ell(k)}$.

Input: Circuit description f_j (containing index j)

1. Hash the index: $v = h(j)$.
2. If $v \neq h(i)$, compute $x = \mathsf{PRF}_{s^*}(v)$, and output $U_k(f_j, x)$.
3. If $v = h(i)$, output y.

Figure 3: "Punctured" Turing machines $\Pi^{h,s}_{i,y} \in \mathcal{M}^*$.

Auxiliary Input Distribution $\mathcal{Z}_k(i, y)$:

1. Sample a hash function $h \leftarrow \mathcal{H}_k$.
2. Sample a PRF seed $s \leftarrow \mathcal{K}_{\mathcal{PRF}}(1^k)$.
3. Sample a punctured PRF seed $s^* \leftarrow \mathsf{Punct}(s, h(i))$, punctured at point $h(i)$.
4. Compute the "correct" punctured evaluation: $y = f_i(\mathsf{PRF}_s(h(i)))$.
5. Output an obfuscation $\tilde{M} \leftarrow e\mathcal{O}(\Pi^{h,s}_{i,y})$, where $\Pi^{h,s}_{i,y}$ is defined from the tuple of values (h, s^*, y), as in Figure 10.

Figure 4: The "punctured" distribution $\mathcal{Z}_k(i, y)$.

$\mathcal{E}$ fails with overwhelming probability: i.e.,

$$\Pr\Big[z \leftarrow \mathcal{Z}_k;\ i \leftarrow \mathcal{K}_{\mathcal{F}}(1^k);\ y \leftarrow \mathcal{A}(i; z);\ x' \leftarrow \mathcal{E}(i; z) : \exists x \ s.t.\ f_i(x) = y \wedge f_i(x') \neq y\Big] \geq 1 - \mathsf{negl}(k).$$

Proof. First note that given auxiliary input $z \leftarrow \mathcal{Z}_k$, $\mathcal{A}$ produces an element in the image of the selected f_i with high probability. That is,

$$\Pr\Big[z \leftarrow \mathcal{Z}_k; i \leftarrow \mathcal{K}_{\mathcal{F}}(1^k); y \leftarrow \mathcal{A}(i; z) : \exists x \text{ s.t. } f_i(x) = y\Big] \geq 1 - \mathsf{negl}(k).$$

Indeed, by the definition of $\mathcal{A}$ and $\mathcal{Z}_k$, and the correctness of the obfuscator $e\mathcal{O}$, then we have with overwhelming probability

$$\mathcal{A}(i; z) = \tilde{M}(f_i) = \Pi^{h,s}(f_i) = f_i(\mathsf{PRF}_s(h(i))),$$

where $z = \tilde{M}$ is an obfuscation of $\Pi^{h,s} \in \mathcal{M}$; i.e., $z = \tilde{M} \leftarrow e\mathcal{O}(\Pi^{h,s})$.

Now, suppose for contradiction that there exists a non-negligible function $\epsilon(k)$ such that for all $k \in \mathbb{N}$ the extractor $\mathcal{E}$ successfully outputs a preimage corresponding to the output $\mathcal{A}(i; z) \in \mathit{Range}(f_i)$ with probability $\epsilon(k)$: i.e.,

$$\Pr\Big[z \leftarrow \mathcal{Z}_k;\ i \leftarrow \mathcal{K}_{\mathcal{F}}(1^k);\ x' \leftarrow \mathcal{E}(i; z) : f_i(x') = \mathcal{A}(i; z) = f_i(\mathsf{PRF}_s(h(i)))\Big] \geq \epsilon(k).$$

where as before, s, h are such that $z = e\mathcal{O}(\Pi^{h,s})$. We show that this cannot be the case, via three steps.

Step 1: Replace $\mathcal{Z}$ with "punctured" distribution $\mathcal{Z}(i,y)$. For every index i of the EOWF family $\mathcal{F}$ and $k \in \mathbb{N}$, consider an alternative distribution $\mathcal{Z}_k(i,y)$ that, instead of sampling and obfuscating a Turing machine $\Pi^{h,s}$ from the class $\mathcal{M}$, as is done for $\mathcal{Z}$, it does so with a Turing machine $\Pi^{h,s}_{i,y} \in \mathcal{M}^*$ as follows. First, it samples a hash function $h \leftarrow \mathcal{H}_k$ and PRF seed s as usual. It then generates a *punctured* PRF key $s^* \leftarrow \mathsf{Punct}(s, h(i))$ that enables evaluation of the PRF on all points except the value $h(i)$. For the specific index i, it computes the correct full evaluation $y := f_i(\mathsf{PRF}_s(h(i)))$. Finally, $\mathcal{Z}_k(i,y)$ outputs an obfuscation of the constructed program $\Pi^{h,s}_{i,y}$ as specified in Figure 10 from the values (h, s^*, y): i.e., $\tilde{M} \leftarrow e\mathcal{O}(\Pi^{h,s}_{i,y})$. See Figure 4 for a full description of $\mathcal{Z}(i,y)$.

We now argue that the extractor $\mathcal{E}$ must also succeed in extracting a preimage when given a value $z^* \leftarrow \mathcal{Z}_k(i,y)$ from this modified distribution instead of $\mathcal{Z}_k$. At a high level, the argument runs as follows. If $\mathcal{E}$'s extraction success drops by a non-negligible amount when instead given a sample from this new distribution, then he can be used to distinguish between obfuscations of corresponding pair of programs $\Pi^{h,s}, \Pi^{h,s}_{i,y}$. By the extractability obfuscation property, this implies that we can efficiently extract an input j on which the two programs differ. But, by construction, the only such inputs j are those that collide with i with respect to the collision-resistant hash function. Thus, such an extractor cannot exist, as it can be used to efficiently find collisions in the CRHF.

We now formalize this intuition.

Lemma 3.3. *It holds that*

$$\Pr\Big[i \leftarrow \mathcal{K}_{\mathcal{F}}(1^k);\ z^* \leftarrow \mathcal{Z}_k(i,y);\ x' \leftarrow \mathcal{E}(i; z^*) : f_i(x') = \mathcal{A}(i; z^*) = f_i(\mathsf{PRF}_s(h(i)))\Big] \geq \epsilon(k) - \mathsf{negl}(k). \quad (3)$$

Proof. Suppose, to the contrary, there exists a non-negligible function $\alpha(k)$ for which the probability in Equation (3) is less than $\epsilon(k) - \alpha(k)$. In particular, there exists a polynomial $p(k)$ such that for infinitely many values of k, there exists an index i_k in the range of $\mathcal{K}_{\mathcal{F}}(1^k)$ for which

$$\Pr\Big[z^* \leftarrow \mathcal{Z}_k(i_k, y^*);\ x' \leftarrow \mathcal{E}(i_k; z^*) : f_{i_k}(x') = \mathcal{A}(i_k; z^*) = f_{i_k}(\mathsf{PRF}_s(h(i_k)))\Big] < \epsilon(k) - \frac{1}{p(k)}. \quad (4)$$

Denote by $I = \{i_k\}$ this ensemble of function indices.

Consider the following non-uniform obfuscation adversary $\mathcal{A}^I_{\mathsf{obf}}$ making use of $\mathcal{E}$, hardcoded with the ensemble of "good" index values I, and given auxiliary input a hash function description h (note that we eventually wish to turn $\mathcal{E}$ into an adversary for the CRHF family).

Obfuscation adversary $\mathcal{A}^I_{\mathsf{obf}}(1^k, M_0, M_1, \tilde{M}, h)$:

1. Execute the EOWF extractor $\mathcal{E}$, giving i_k (from the hardcoded ensemble I) as the target index, and the obfuscated program $\tilde{M}$ as auxiliary input. That is, $x' \leftarrow \mathcal{E}(i_k; \tilde{M})$.
2. Output 0 if $\mathcal{E}$ succeeded in extracting a preimage: i.e., if $f_{i_k}(x') = \tilde{M}(i_k)$. Otherwise, output a random bit $b' \leftarrow \{0,1\}$.

Now, expanding out the sampling procedures of the distributions $\mathcal{Z}_k, \mathcal{Z}_k(i_k, y)$, Equation (4) implies that for the hardcoded ensemble $E = \{i_k\}$, it holds for infinitely many values of k that

$$\Pr\Big[h \leftarrow \mathcal{H}_k;\ (M_0, M_1) \leftarrow \mathsf{ProgSample}(1^k, i_k, h);\ b \leftarrow \{0,1\}; \tilde{M} \leftarrow e\mathcal{O}(M_b) : \mathcal{A}^I_{\mathsf{obf}}(1^k, M_0, M_1, \tilde{M}, h) = b\Big] \geq \frac{1}{2} + \frac{1}{2p(k)}, \quad (5)$$

where $\mathsf{ProgSample}(1^k, i_k, h)$ is given by:

1. $s \leftarrow \mathcal{K}_{\mathcal{PRF}}(1^k)$.
2. $s^* \leftarrow \mathsf{Punct}(s, h(i_k))$.
3. Output $M_0 := \Pi^{h,s},\ M_1 := \Pi^{h,s}_{i,y}$.

Indeed, note that if $b = 0$ then the obfuscated program $\tilde{M}$ given to $\mathcal{A}^I_{\mathsf{obf}}$ in Equation (5) is distributed as $\tilde{M} \leftarrow \mathcal{Z}_k$, in which case $\mathcal{E}$ extracts a preimage with probability at least $\epsilon(k)$. In contrast, if $b = 1$, then $\tilde{M}$ given to $\mathcal{A}^E_{\mathsf{obf}}$ is distributed as $\tilde{M} \leftarrow \mathcal{Z}_k(i_k, y)$, in which case (for infinitely many k) $\mathcal{E}$ extracts a preimage with probability $\epsilon(k) - 1/p(k)$.

Thus, with probability at least $1/4p(k)$ over $h \leftarrow \mathcal{H}_k; (M_0, M_1) \leftarrow \mathsf{ProgSample}(1^k, i_k, h)$, it holds that

$$\Pr\left[b \leftarrow \{0,1\}; \tilde{M} \leftarrow e\mathcal{O}(M_b) : \mathcal{A}^I_{\mathsf{obf}}(1^k, M_0, M_1, \tilde{M}, h) = b\right] \geq \frac{1}{2} + \frac{1}{4p(k)}. \tag{6}$$

Now, by the extractability obfuscation security of $e\mathcal{O}$ *with respect to auxiliary input distribution* $\mathcal{Z}' \equiv \mathcal{H}$ corresponding to the distribution of hash function descriptions, there exists a non-uniform PPT algorithm $\mathcal{E}^{I'}_{\mathsf{obf}}$ (with some non-uniform advice ensemble I') and a polynomial $q(k)$ corresponding to $\mathcal{A}^I_{\mathsf{obf}}$ and $p(k)$, such that for any pair of programs M_0, M_1 for which $\mathcal{A}^I_{\mathsf{obf}}$ successfully distinguishes between the obfuscations $e\mathcal{O}(M_0)$ and $e\mathcal{O}(M_1)$ with advantage $1/4p(k)$ (when given auxiliary input $h \leftarrow \mathcal{H}_k$), then $\mathcal{E}^{I'}_{\mathsf{obf}}$ (given the pair M_0, M_1) will extract a disagreeing input j such that $M_0(j) \neq M_1(j)$ with non-negligible probability $1/q(k)$ when given the same auxiliary input $h \leftarrow \mathcal{H}_k$.

By (6), this means that for infinitely many k, with probability $1/4p(k)$ over $h \leftarrow \mathcal{H}_k;\ (M_0, M_1) \leftarrow \mathsf{ProgSample}(1^k, i_k, h)$, we have

$$\Pr\left[j \leftarrow \mathcal{E}^{I'}_{\mathsf{obf}}(1^k, M_0, M_1, h) : M_0(j) \neq M_1(j)\right] \geq \frac{1}{q(k)}. \tag{7}$$

We now argue that this contradicts the collision resistance of the hash family $\mathcal{H}$. Namely, consider the following non-uniform PPT collision-finding adversary $\mathcal{A}^{I,I'}_{\mathsf{CR}}$, who is given both advice ensembles I, I'.

Collision-finding adversary $\mathcal{A}^{I,I'}_{\mathsf{CR}}(1^k)$:

1. Receive a random hash function $h \leftarrow \mathcal{H}_k$ from the CRHF challenger.
2. Sample a corresponding pair of programs $(M_0, M_1) \leftarrow \mathsf{ProgSample}(1^k, i_k, h)$ where i_k is specified by the advice ensemble I. Recall that M_0 is a program $\Pi^{h,s}$ and M_1 is a corresponding "i_k-punctured" program $\Pi^{h,s}_{i,y}$.
3. Execute the (non-uniform) extractor $\mathcal{E}^{I'}_{\mathsf{obf}}$ from above on this pair of programs, hash function descriptor h, and advice from I': that is, $j \leftarrow \mathcal{E}^{I'}_{\mathsf{obf}}(1^k, M_0, M_1, h)$.
4. Output the pair (i_k, j) as the alleged collision for h.

By Equation (7), we have that with probability at least $1/4p(k)$ over the sampling of the challenge function $h \leftarrow \mathcal{H}_k$ and $\mathcal{A}^I_{\mathsf{CR}}$'s sampling of $(M_0, M_1) \leftarrow \mathsf{ProgSample}(1^k, i_k, h)$ that the extractor $\mathcal{E}^{I'}_{\mathsf{obf}}$ succeeds in extracting a disagreeing input with probability $1/q(k)$. Thus, with probability $1/4p(k)q(k)$, the value j output by $\mathcal{A}^{I,I'}_{\mathsf{CR}}(1^k)$ satisfies $M_0(j) \neq M_1(j)$; that is, $\Pi^{h,s}(j) \neq \Pi^{h,s}_{i,y}(j)$. But, by construction of $\Pi^{h,s}_{i,y}$, the set of such inputs consists exactly of the values $\{j \neq i_k | h(j) = h(i_k)\}$. (Indeed, recall that for all inputs j with $h(j) = h(i_k)$, we hardcoded their final output to the value corresponding to $\Pi^{h,s}(i_k)$). Therefore, with non-negligible probability

$(1/4p(k)q(k)$ for infinitely many k), $\mathcal{A}_{\mathsf{CR}}^{I,I'}$ has found a collision in h, yielding a contradiction to the security of $\mathcal{H}$. The lemma follows.

□

Step 2: Replace "correct" hardcoded y in $\mathcal{Z}(i,y)$ with random f_i evaluation. Next, we consider another experiment where $\mathcal{Z}_k(i,y)$ is altered to a nearly identical distribution $\mathcal{Z}_k(i,u)$ where, instead of hardcoding the "correct" i-evaluation value $y = f_i(\mathsf{PRF}_s(h(i)))$ in the generated "punctured" program $\Pi_{i,y}^{h,s}$, the distribution $\mathcal{Z}_k(i,u)$ now simply samples a random f_i output $y = f_i(u)$ for an independent random $u \leftarrow \{0,1\}^k$. We claim that the original EOWF extractor $\mathcal{E}$ still succeeds in finding a preimage when given this new auxiliary input distribution:

Lemma 3.4. *It holds that*

$$\Pr\Big[i \leftarrow \mathcal{K}_{\mathcal{F}}(1^k);\ z^{**} \leftarrow \mathcal{Z}_k(i,u);\ x' \leftarrow \mathcal{E}(i;z^{**}) : f_i(x') = \mathcal{A}(i;z^{**}) = f_i(u)\Big] \geq \epsilon(k) - \mathsf{negl}(k). \quad (8)$$

Proof. This follows from the fact that $\mathsf{PRF}_s(h(i))$ is pseudorandom, even given the $h(i)$-punctured key s^*.

Formally, consider an algorithm $\mathcal{A}_{\mathsf{PRF}}^0$ which, on input the security parameter 1^k, a pair of values i, h, and a pair s^*, x (that will eventually correspond to a challenge punctured PRF key, and either $\mathsf{PRF}_s(h(i))$ or random u), performs the following steps.

Algorithm $\mathcal{A}_{\mathsf{PRF}}^0(1^k, i, h, s^*, x)$:

1. Take $y = f_i(x)$, and obfuscate the associated program $\Pi_{i,y}^{h,s}$: i.e., $z^{**} \leftarrow e\mathcal{O}(1^k, \Pi_{i,y}^{h,s})$.
2. Run the EOWF extractor given index i and auxiliary input z^{**}: $x' \leftarrow \mathcal{E}(i;z^{**})$.
3. Output 0 if $\mathcal{E}$ succeeds in extracting a valid preimage: i.e., if $f_i(x') = y^* = f_i(x)$. Otherwise, output a random bit $b \leftarrow \{0,1\}$.

Now, suppose Lemma 3.4 does not hold: i.e., the probability in Equation (8) differs by some non-negligible amount from $\epsilon(k)$. Then, expanding out the sampling procedure of $\mathcal{Z}_k(i,y)$ and $\mathcal{Z}_k(i,u)$, we have for some non-negligible function $\alpha(k)$ that

$$\Pr\Big[i \leftarrow \mathcal{K}_{\mathcal{F}}(1^k);\ h \leftarrow \mathcal{H}_k;\ s \leftarrow \mathcal{K}_{\mathcal{PRF}}(1^k);\ s^* \leftarrow \mathsf{Punct}(s, h(i));\ u \leftarrow \{0,1\}^k; b \leftarrow \{0,1\} : \mathcal{A}_{\mathsf{PRF}}^0(1^k, i, h, x_b) = b\Big] \geq \frac{1}{2} + \alpha(k), \quad (9)$$

where $x_0 := \mathsf{PRF}_s(h(i))$ and $x_1 := u$. Indeed, in the case $b = 0$, the auxiliary input z^{**} generated by $\mathcal{A}_{\mathsf{PRF}}$ and given to $\mathcal{E}$ has distribution exactly $\mathcal{Z}(i,y)$, whereas in the case $b = 1$, the generated z^{**} has distribution exactly $\mathcal{Z}(i,u)$.

In particular, there exists a polynomial $p(k)$ such that for infinitely many k, there exists an index i_k and hash function $h_k \in \mathcal{H}_k$ with

$$\Pr\Big[s \leftarrow \mathcal{K}_{\mathcal{PRF}}(1^k);\ s^* \leftarrow \mathsf{Punct}(s, h(i_k));\ u \leftarrow \{0,1\}^k; b \leftarrow \{0,1\} : \mathcal{A}_{\mathsf{PRF}}^0(1^k, i_k, h, x_b) = b\Big] \geq \frac{1}{2} + \frac{1}{p(k)}, \quad (10)$$

where x_0, x_1 are as before.

Consider a non-uniform punctured-PRF adversary $\mathcal{A}^I_{\mathsf{PRF}}$ (with the ensemble $I = \{i_k, h_k\}$ hardcoded) that first selects the challenge point $h_k(i_k)$; receives the PRF challenge information (s^*, x) for this point; executes $\mathcal{A}^0_{\mathsf{PRF}}$ on input $(1^k, i_k, h_k, s^*, x)$, and outputs the corresponding bit b output by $\mathcal{A}^0_{\mathsf{PRF}}$. Then by (10), it follows that $\mathcal{A}^I_{\mathsf{PRF}}$ breaks the security of the punctured PRF.

□

Step 3: Such an extractor breaks one-wayness of EOWF. Finally, we observe that this means that $\mathcal{E}$ can be used to break the one-wayness of the original function family $\mathcal{F}$. Indeed, given a random key i and a challenge output $y = f_i(u)$, an inverter can simply sample a hash function h and $h(i)$-punctured PRF seed s^* on its own, construct the program $\Pi^{h,s}_{i,y}$ with its challenge y hardcoded in, and sample an obfuscation $z^{**} \leftarrow e\mathcal{O}(\Pi^{h,s}_{i,y})$. Finally, it runs $\mathcal{E}(i, z^{**})$ to invert y^*, with the same probability $\epsilon(k) - \mathsf{negl}(k)$.

□

This concludes the proof of Theorem 3.1. □

3.2 Extractability Obfuscation Versus SNARKs

We link the existence of extractability obfuscation for NC^1 and the existence of succinct non-interactive arguments of knowledge (SNARKs), via an intermediate step of *proximity* extractable one-way functions (PEOWFs), a notion related to EOWFs, introduced in [BCCT12]. Namely, building upon the results of the previous subsection, and results of [BCCT12], we show:

1. Assuming the existence of fully homomorphic encryption (FHE) with decryption in NC^1 and SNARKs for NP, there exist efficient distributions $\mathcal{Z}, \mathcal{Z}'$ such that extractability obfuscation for NC^1 w.r.t. auxiliary input distribution $\mathcal{Z}'$ implies that there *cannot* exist PEOWFs $\{f : \{0,1\}^k \to \{0,1\}^k\}$ w.r.t. $\mathcal{Z}$.
2. PEOWFs $\{f : \{0,1\}^k \to \{0,1\}^k\}$ w.r.t. this auxiliary input distribution $\mathcal{Z}'$ are *implied by* the existence of SNARKs for NP secure w.r.t. a third efficient auxiliary input distribution $\mathcal{Z}''$ (and collision-resistant hash functions), as shown in [BCCT12].
3. Thus, one of these conflicting hypotheses must be false. That is, there exist efficient distributions $\mathcal{Z}', \mathcal{Z}''$ such that assuming existence of FHE with decryption in NC^1 and collision-resistant hash functions, then either: (1) extractability obfuscation for NC^1 w.r.t. $\mathcal{Z}'$ does not exist, or (2) SNARKS for NP w.r.t. $\mathcal{Z}''$ do not exist.

Note that we focus on the specific case of PEOWFs with k-bit inputs and k-bit outputs, as this suffices to derive the desired contradiction; however, the theorems following extend also to the more general case of PEOWF output length (demonstrating an efficient distribution $\mathcal{Z}$ to rule out each potential output length $\ell(k)$). Further, the result in Step 1 holds also with SNARKs for NP replaced by the weaker assumption of existence of P-certificates.[9]

3.2.1 Proximity EOWFs

We begin by defining Proximity EOWFs.

[9] P-certificates are succinct non-interactive arguments for P, satisfying soundness but not necessarily extractability.

Proximity Extractable One-Way Functions (PEOWFs). In a Proximity EOWF (PEOWF), the extractable function family $\{f_i\}$ is associated with a "proximity" equivalence relation $\sim$ on the range of f_i, and the one-wayness and extractability properties are modified with respect to this relation. The one-wayness is strengthened: not only must it be hard to find an exact preimage of v, but it is also hard to find a preimage of any equivalent $v \sim v'$. The extractability requirement is weakened accordingly: the extractor does not have to output an exact preimage of v, but only a preimage of of some equivalent value $v' \sim v$.

As an example, consider functions of the form $f : x \mapsto (f_1(x), f_2(x))$ and equivalence relation on range elements $(a, b) \sim (a, b')$ whose first components agree. Then the proximity extraction property requires for any adversary $\mathcal{A}$ who outputs an image element $(a, b) \in \mathit{Range}(f)$ that there exists an extractor $\mathcal{E}$ who finds an input x for which $f(x) = (a, b')$ for some b' not necessarily equal to b.

In this work, we allow the relation $\sim$ to depend on the function index i, but require that the relation $\sim$ is *publicly* (and efficiently) testable. We further consider non-uniform adversaries and extraction algorithms, and (in line with this work) auxiliary inputs coming from a specified distribution $\mathcal{Z}$.

Definition 3.5 ($\mathcal{Z}$-Auxiliary-Input Proximity EOWFs)**.** Let ℓ, m be polynomially bounded length functions. An efficiently computable family of functions

$$\mathcal{F} = \left\{ f_i : \{0,1\}^k \to \{0,1\}^{\ell(k)} \;\middle|\; i \in \{0,1\}^{m(k)}, k \in \mathbb{N} \right\},$$

associated with an efficient probabilistic key sampler $\mathcal{K}_{\mathcal{F}}$, is a *$\mathcal{Z}$-auxiliary-input proximity extractable one-way function* if it satisfies the following (strong) one-wayness, (weak) extraction, and public testability properties:

- **(Strengthened) One-wayness:** For non-uniform polynomial-time $\mathcal{A}$ and sufficiently large security parameter $k \in \mathbb{N}$,

$$\Pr\left[z \leftarrow \mathcal{Z}_k;\ i \leftarrow \mathcal{K}_{\mathcal{F}}(1^k);\ x \leftarrow \{0,1\}^k;\ x' \leftarrow \mathcal{A}(i, f_i(x); z) : f_i(x') \sim f_i(x) \right] \leq \mathsf{negl}(k).$$

- **(Weakened) Extractability:** For any non-uniform polynomial-time adversary $\mathcal{A}$, there exists a non-uniform polynomial-time extractor $\mathcal{E}$ such that, for sufficiently large security parameter $k \in \mathbb{N}$,

$$\Pr\left[z \leftarrow \mathcal{Z}_k;\ i \leftarrow \mathcal{K}_{\mathcal{F}}(1^k);\ y \leftarrow \mathcal{A}(i; z);\ x' \leftarrow \mathcal{E}(i; z) : \exists x \text{ s.t. } f_i(x) = y \wedge f_i(x') \not\sim y \right] \leq \mathsf{negl}(k).$$

- **Publicly Testable Relation:** There exists a deterministic polytime machine $\mathcal{T}$ such that, given the function index i, $\mathcal{T}$ accepts $y, y' \in \{0,1\}^{\ell(k)}$ if and only if $y \sim_k y'$.

3.2.2 (NC^1 Extractability Obf + FHE + SNARK) $\implies$ *No* $\mathcal{Z}$-PEOWF

We now show that, assuming the existence of fully homomorphic encryption (FHE) with decryption in NC^1,[10] then for some efficiently computable distributions $\mathcal{Z}_{\mathsf{obf}}, \mathcal{Z}_{\mathsf{SNARK}}, \mathcal{Z}_{\mathsf{PEOWF}}$, if there exist extractability obfuscators for NC^1 w.r.t. auxiliary input $\mathcal{Z}_{\mathsf{obf}}$ and SNARKs w.r.t. auxiliary input $\mathcal{Z}_{\mathsf{SNARK}}$, then there *cannot* exist PEOWFs w.r.t. auxiliary input $\mathcal{Z}_{\mathsf{PEOWF}}$. This takes place in two steps.

[10] As is the case for nearly all existing FHE constructions (e.g., [GSW13, BV13]).

First, we remark that an identical proof to that of Theorem 3.1 rules out the existence of $\mathcal{Z}$-auxiliary-input *proximity EOWFs* in addition to standard EOWFs, based on the same assumptions: namely, assuming $\mathcal{Z}'$-auxiliary-input extractability obfuscation for Turing machines and collision-resistant hash functions (which are implied by FHE [IKO05]). Indeed, assuming the existence of a PEOWF extractor $\mathcal{E}$ for the adversary $\mathcal{A}$ and auxiliary input distribution $\mathcal{Z}$ (who extracts a "related" preimage to the target value), the same procedure yields a PEOWF inverter who similarly extracts a "related" preimage to any challenge output. In the reduction, it is merely required that the success of $\mathcal{E}$ is efficiently and publicly testable (this is used to construct a distinguishing adversary for the extractability obfuscation scheme, in Step 1). However, this is directly implied by the public testability of the PEOWF relation $\sim$, as specified in Definition 3.5.

Theorem 3.6. *There exist efficient, uniformly samplable distributions $\mathcal{Z}, \mathcal{Z}'$ such that, assuming the existence of collision-resistant hash functions and $\mathcal{Z}'$-auxiliary-input extractability obfuscation for polynomial-size Turing machines, there cannot exist (publicly testable) $\mathcal{Z}$-auxiliary-input PEOWFs $\{f_i : \{0,1\}^k \to \{0,1\}^k\}$.*

Now, in [BCP13], it was shown that extractability obfuscation for all polynomial-size Turing machines can be achieved by bootstrapping up from extractability obfuscation for NC^1, assuming the existence of FHE with decryption in NC^1 and SNARKs.[11] The resulting Turing machine obfuscator will be secure w.r.t. auxiliary input $\mathcal{Z}'$ if the underlying SNARK scheme is secure w.r.t. auxiliary input $\mathcal{Z}'$, and the original NC^1 obfuscator is secure w.r.t. an augmented auxiliary input distribution $\mathcal{Z}_{\mathsf{obf}} := (\mathcal{Z}', \mathcal{Z}_{\mathsf{crs}})$, formed by concatenating a sample from $\mathcal{Z}'$ with a CRS generated for the SNARK scheme.

Putting this together with Theorem 3.6, we thus have the following corollary.

Corollary 3.7. *There exist efficient, uniformly samplable distributions $\mathcal{Z}, \mathcal{Z}_{\mathsf{obf}}$ such that, assuming the existence of SNARKs and FHE with decryption in NC^1, then assuming the existence of extractability obfuscation* for NC^1 *w.r.t. auxiliary input $\mathcal{Z}_{\mathsf{obf}}$, there cannot exist PEOWFs $\{f_i : \{0,1\}^k \to \{0,1\}^k\}$ w.r.t. auxiliary input $\mathcal{Z}$.*

3.2.3 (SNARK + CRHF) $\implies$ $\mathcal{Z}$-PEOWF

As shown in [BCCT12], Proximity EOWFs (PEOWFs) with respect to an auxiliary input distribution $\mathcal{Z}$ are *implied by* collision-resistant hash functions (CRHF) and SNARKs secure with respect to a related auxiliary input distribution.[12]

Loosely, the transformation converts any collision-resistant hash function family $\mathcal{F}$ into a PEOWF by appending to the output of each $f \in \mathcal{F}$ a succinct SNARK argument π_x that there exists a preimage x yielding output $f(x)$. (If the Prover algorithm of the SNARK system is randomized, then the function is also modified to take an additional input, which is used as the random coins for the SNARK generation). The equivalence relation on outputs is defined by $(y, \pi) \sim (y', \pi')$ if $y = y'$ (note that this relation is publicly testable). More explicitly, consider the new function family $\mathcal{F}'$ composed of functions

$$f'_{\mathsf{crs}}(x, r) = \Big(f(x), \mathsf{Prove}(1^k, \mathsf{crs}, f(x), x; r)\Big),$$

[11] The result of [BCP13] actually only requires P-certificates (succinct arguments for languages in P); such objects can, in particular, be instantiated by SNARKs.

[12] [BCCT12] consider the setting of arbitrary auxiliary input; however, their construction directly implies similar results for specific auxiliary input distributions.

where a function $f'_{\mathsf{crs}} \in \mathcal{F}'$ is sampled by first sampling a function $f \leftarrow \mathcal{F}$ from the original CRHF family, and then sampling a CRS for the SNARK scheme, $\mathsf{crs} \leftarrow \mathsf{CRSGen}(1^k)$.

Now (as proved in [BCCT12]), the resulting function family will be a PEOWF with respect to auxiliary input $\mathcal{Z}$ if the underlying SNARK system is secure with respect to an augmented auxiliary input distribution $\mathcal{Z}_{\mathsf{SNARK}} := (\mathcal{Z}, \mathcal{Z}')$, formed by concatenating a sample from $\mathcal{Z}$ with a function index sampled from the collision-resistant hash function family $\mathcal{F}$.

Theorem 3.8 ([BCCT12])**.** *There exist efficient, uniformly samplable distributions $\mathcal{Z}, \mathcal{Z}_{\mathsf{SNARK}}$ such that, assuming the existence of collision-resistant hash functions and SNARKs for NP secure w.r.t. auxiliary input distribution $\mathcal{Z}_{\mathsf{SNARK}}$, then there exist PEOWFs $\{f_i : \{0,1\}^k \to \{0,1\}^k\}$ w.r.t. $\mathcal{Z}$.*

3.2.4 Reaching a Standoff

Observe that the conclusions of Corollary 3.7 and Theorem 3.8 are in direct contradiction. Thus, it must be that one of the two sets of assumptions is false. Namely,

Corollary 3.9. *Assuming the existence of fully homomorphic encryption with decryption in NC^1, there exist efficiently samplable distributions $\mathcal{Z}_{\mathsf{SNARK}}, \mathcal{Z}_{\mathsf{obf}}$ such that one of the following two objects cannot exist:*

- *SNARKs w.r.t. auxiliary input distribution $\mathcal{Z}_{\mathsf{SNARK}}$.*
- *Extractability obfuscation for NC^1 w.r.t. auxiliary input distribution $\mathcal{Z}_{\mathsf{obf}}$.*

More explicitly, we have that $\mathcal{Z}_{\mathsf{SNARK}} = (\mathcal{Z}, \mathcal{Z}')$ and $\mathcal{Z}_{\mathsf{obf}} = (\mathcal{Z}_{\mathsf{crs}}, \mathcal{Z}')$, where $\mathcal{Z}$ is composed of an obfuscated program, $\mathcal{Z}'$ consists of a randomly sampled index from a CRHF family, and $\mathcal{Z}_{\mathsf{crs}}$ corresponds to a randomly sampled CRS from a SNARK system.

4 Applications of "Succinct Punctured Programs" Technique

We now demonstrate a variety of applications of our "succinct punctured programs" technique.

4.1 Perfect NIZK Universal Arguments

In [SW13], Sahai and Waters demonstrated a construction of a Non-Interactive Zero-Knowledge (NIZK) argument system with perfect zero knowledge from indistinguishability obfuscation, supporting fixed NP languages with statements (and witnesses) up to an a-priori bounded size. Using our succinct punctured programs technique, we achieve a non-interactive *universal argument system*, also perfectly zero knowledge, for languages and statements of unbounded polynomial size (see Definition 2.6).

Theorem 4.1. *Assume the existence of auxiliary-input-secure extractability obfuscation for TM and collision-resistant hash functions. Then for any constant $\epsilon > 0$ there exists a perfect zero-knowledge universal argument system, as in Definition 2.6.*

As with the construction of [SW13], the obfuscation scheme we use must be secure with respect to a particular distribution $\mathcal{Z}$ consisting of a second obfuscated program. For simplicity, however, we state our result with respect to obfuscation secure against arbitrary auxiliary input.

NIZK of [SW13]. We first recall the NIZK construction of [SW13]. The system consists of two obfuscated circuits (serving as the CRS):

- A Prove circuit, which has hardcoded a PRF seed s, takes as input a statement and witness pair (x, w) and outputs the PRF evaluation $\mathsf{PRF}_s(x)$ on x if w is a valid witness (i.e., $R(x, w) = 1$).
- A Verify circuit, which has hardcoded the same PRF seed s and a description of a one-way function f, takes as input a statement and alleged proof (x, π), and outputs 1 exactly if $f(\pi)$ is equal to $f(\mathsf{PRF}_s(x))$. (The introduction of the one-way function f is not needed for correctness, but rather to argue security by use of the "punctured programs" technique).

Sahai and Waters show this scheme satisfies perfect zero knowledge and (non-adaptive) soundness assuming the obfuscation scheme used is a secure indistinguishability obfuscator with respect to arbitrary auxiliary input [SW13].

Note that here the size of the Prove and Verify circuits must grow with the size of potential statements x, thus inherently fixing an upper bound on the handled statement size at the time of CRS generation.

Succinct ZK Universal Arguments. Following our technique of succinct punctured programs, we modify the construction of [SW13] in two ways.

First, we replace the obfuscated Prove and Verify circuits (which depend on a fixed NP relation R) with obfuscated *Turing machines*, which can accept arbitrary polynomial-size relations as part of their input. That is, we now consider instances of the universal relation R_U, as described in Definition 2.6. To obtain obfuscation of Turing machines, we use extractability obfuscation in the place of indistinguishability obfuscation. On the surface, this modification almost appears to suffice for our goal. However, one problem remains: to prove soundness of the resulting scheme for an instance $(M, x, t) \notin R_U$, we must argue that the obfuscated Turing machines are indistinguishable from obfuscations of corresponding (M, x, t)-*punctured* programs, whose size must grow with $|(M, x, t)|$. However, this requires the size of the obfuscated Turing machines to grow with $|(M, x, t)|$, thus annihilating our universality.

To solve this problem, we incorporate a second modification to the [SW13] construction, by first computing a *collision-resistant hash* of the input statement, and proceeding with this hashed value $h(M, x, t)$. Now we need only that the obfuscated Prove and Verify programs are indistinguishable from an obfuscation of a program that is punctured at $h(M, x, t)$, whose size can now be made independent of $|(M, x, t)|$. As in the previous section, indistinguishability of these programs will follow from the security of the *extractability* obfuscation together with the collision resistance of h.

Formally, consider the following tools:

1. $\mathcal{PRF} = \{\mathsf{PRF}_s : \{0,1\}^{m(k)} \to \{0,1\}^k\}_{s \in \{0,1\}^k}$ a puncturable PRF family.
   Note that by Theorem 2.8, punctured PRFs for these parameters exist based on OWFs, which are implied by CRHF.
2. $\mathcal{H} = \{\mathcal{H}_k\}$ a CRHF family with $h : \{0,1\}^* \to \{0,1\}^{m(k)}$ for each $h \in \mathcal{H}_k$.
3. $\mathcal{F} = \{f : \{0,1\}^k \to \{0,1\}^{k^\epsilon}\}$ be a OWF family.

We now present our construction.

Succinct ZK Universal Argument Construction $(\mathsf{CRSGen}, \mathsf{Prove}, \mathsf{Verify})$:

$\mathsf{CRSGen}(1^k)$: on input the security parameter, the CRS generation procedure samples a PRF seed $s \leftarrow \mathcal{K}_{\mathcal{PRF}}(1^k)$, a hash function $h \leftarrow \mathcal{H}_k$, and a OWF $f \leftarrow \mathcal{F}_k$. It then generates obfuscations of the corresponding Turing machines $P^{h,s}$ and $V^{h,s,f}$, as defined in Figures 5

Turing Machine $P^{h,s}$:

Hardwired: Hash function $h : \{0,1\}^* \to \{0,1\}^{m(k)}$, PRF seed $s \in \{0,1\}^k$.

Inputs: Universal statement $\vec{x} = (M, x, t)$, alleged witness w.

1. Check validity of w as witness for $\vec{x} = (M, x, t)$ by executing the Turing machine M on input (x, w) for t steps. If $M(x, w) \neq 1$, output $\perp$ and terminate. Otherwise, continue.
2. Hash the statement description: $v = h(\vec{x})$.
3. Compute the PRF on this hash: $y = \mathsf{PRF}_s(v)$.
4. Output y.

Figure 5: Turing machines $P^{h,s}$.

and 7. That is, $\tilde{P} \leftarrow e\mathcal{O}(1^k, P^{h,s})$ and $\tilde{V} \leftarrow e\mathcal{O}(1^k, V^{h,s,f})$. Output the pair of obfuscated programs $\mathsf{crs} = (\tilde{P}, \tilde{V})$.

$\mathsf{Prove}(\mathsf{crs}, \vec{x}, w)$: Evaluate the obfuscated program $\tilde{P} \in \mathsf{crs}$ on input $(\vec{x}, w)$: i.e., output $\tilde{P}(\vec{x}, w)$.

$\mathsf{Verify}(\mathsf{crs}, \vec{x}, \pi)$: Evaluate the obfuscated program $\tilde{V} \in \mathsf{crs}$ on input $(\vec{x}, \pi)$: i.e., output $\tilde{V}(\vec{x}, \pi)$.

Proof of Theorem 4.1. Perfect zero knowledge holds as the distribution of proofs can be perfectly simulated given the PRF seed s (without knowledge of any witness).

The size and time complexities follow from a straightforward analysis of the programs $P^{h,s}, V^{h,s,f}$. In particular, for any input (M, x, t), the proof size is k^ϵ bits (independent of $|M|, t$) corresponding to the output of the OWF, and verification of a proof on (M, x, t) requires only computing a hash and PRF, and not executing M.

(Non-adaptive) soundness follows an analogous sequence of hybrids as Theorem 3.1, mirroring the approach of [SW13]. Namely, for $\vec{x} = (M, x, t) \notin R_U$, the obfuscated programs $P^{h,s}, V^{h,s,f}$ are sequentially replaced with their "$\vec{x}$-punctured" counterparts $P^{h,s}_{\vec{x}}$ and $V^{h,s,f}_{\vec{x},y}$ (as defined in Figures 6 and 8), and then replacing the hardcoded output y in the $V^{h,s,f}_{\vec{x},y}$ program with a random output $f(u)$ of the OWF f for uniform $u \in \{0,1\}^k$. By the security of the extractability obfuscator, an adversary's success probability in generating a proof on $\vec{x}$ cannot decrease by too much, or such adversary can be used to find collisions in the hash function h. Producing a verifying proof on $\vec{x}$ within this final hybrid then corresponds to inverting a random evaluation of the OWF, contradicting its one-wayness.

□

4.2 *Universal* Instantiation of Full-Domain Hash

The full-domain hash (FDH) signature paradigm, first proposed by Bellare and Rogaway [BR93, BR96], provides a means of building a signature scheme from any trapdoor permutation, within the heuristic random oracle model. Specifically, a signature on a message m is generated by evaluating the random oracle at input m, and then computing the inverse of the trapdoor permutation on this value $\mathsf{RO}(m)$. This work has been very influential and formed the foundation for part of the PKCS#1 standard [KS98]. However, negative results in later years called into question the rigorous implications of security proofs in the random oracle model [CGH04, GK03, BBP04], showing e.g. that for some such applications *no* concrete instantiation of the random oracle can yield security.

In a recent work, Hohenberger, Sahai, and Waters [HSW13] presented a methodology for instantiating the random oracle that provides (selective) security for full-domain hash signatures in

Punctured Turing Machine $P^{h,s}_{\vec{x}}$:

Hardwired: Hash function $h : \{0,1\}^* \to \{0,1\}^{m(k)}$, *punctured* PRF seed $s^* \in \{0,1\}^k$, punctured PRF input $h(\vec{x})$.

Inputs: Universal statement $\vec{x}' = (M', x', t')$, alleged witness w.

1. Check validity of w as witness for $\vec{x}' = (M', x', t')$ by executing the Turing machine M' on input (x', w) for t' steps. If $M(x', w) \neq 1$, output $\perp$ and terminate. Otherwise, continue.
2. Hash the statement description: $v = h(\vec{x})$.
3. Compute the PRF on this hash: $y = \mathsf{PRF}_s(v)$.
4. Output y.

Figure 6: Punctured Turing machines $P^{h,s}_{\vec{x}}$. Note for $(M, x, t) \notin R_U$, the punctured output is $\perp$.

Turing Machine $V^{h,s,f}$:

Hardwired: Hash function $h : \{0,1\}^* \to \{0,1\}^{m(k)}$, PRF seed $s \in \{0,1\}^k$, function $f \in \mathcal{F}$ from the OWF family.

Inputs: Universal statement $\vec{x} = (M, x, t)$, alleged proof π.

1. Compute the "correct" proof for $\vec{x}$: i.e., $\pi' = \mathsf{PRF}_s(h(\vec{x}))$.
2. Verify whether $f(\pi) = f(\pi')$. If so, output 1; otherwise output 0.

Figure 7: Turing machines $V^{h,s,f}$.

Punctured Turing Machine $V^{h,s,f}_{\vec{x},y}$:

Hardwired: Hash function $h : \{0,1\}^* \to \{0,1\}^{m(k)}$, *punctured* PRF seed $s \in \{0,1\}^k$, punctured PRF input $h(\vec{x})$, punctured output y, OWF $f \in \mathcal{F}$.

Inputs: Universal statement $\vec{x}' = (M', x', t')$, alleged proof π.

1. If $h(\vec{x}') \neq h(\vec{x})$, then compute the "correct" proof for x' as $\pi' = \mathsf{PRF}_s(h(\vec{x}'))$, and output 1 if and only if $f(\pi) = f(\pi')$.
2. If $h(\vec{x}') = h(\vec{x})$, then output 1 if and only if $f(\pi) = y$, where y is hardcoded.

Figure 8: Punctured Turing machines $V^{h,s,f}_{\vec{x},y}$.

the standard model, building on recent advances in indistinguishability obfuscation (in particular, using the punctured programs paradigm of [SW13]). They demonstrate for every trapdoor permutation f that there exists a hash function $\mathcal{R}_f$ (tailored to f) such that the FDH signature scheme is (selectively) secure in the standard model when using f and instantiating the random oracle by $\mathcal{R}_f$.

We show that our succinct punctured programs technique yields a *universal* instantiation of the random oracle providing security for full-domain hash signatures. That is, we provide a single family of Turing machines $\mathcal{R} = \{\mathcal{R}_k\}$ such that, for *any* injective trapdoor function f, the Bellare-Rogaway Full-Domain Hash signature scheme [BR93, BR96] using f and instantiating the random oracle by $\mathcal{R}$ is selectively secure in the standard model. This construction involves a tweak to the FDH signature structure, in which the random oracle takes as input a description of the trapdoor permutation f in addition to the message to be signed: i.e., $\mathsf{Sign}(m) = f^{-1}(\mathsf{RO}(m, f))$.

Our construction. Intuitively, to instantiate the random oracle, we would like for each message m to be able to sample a random image of a given trapdoor function f, without revealing information about the corresponding preimage. In [HSW13], this is done by providing an (indistinguishability) obfuscation of a circuit that computes a PRF on the input m and then evaluates f on this outcome. Using the technique of punctured programs [SW13], Hohenberger *et al.* [HSW13] show that given this obfuscated circuit, no information is revealed about the evaluation of the PRF on the (pre-selected) forgery challenge message m, so that forging on message m implies inverting a random(-looking) output $f(\mathsf{PRF}_s(m))$ of the trapdoor function, contradicting its assumed security. In this construction, however, the obfuscated circuit (i.e., the instantiation of the random oracle) inherently depends on the trapdoor function f.

We avoid this dependency by obfuscating a *Turing machine* that takes as input a message m and description of the desired trapdoor function f (described as a poly-size circuit), and outputs a seemingly random evaluation of f. As in our constructions from the previous sections, in order to allow the size of the obfuscated program to be independent of the size of the input trapdoor function (while still maintaining security), we first *hash* the input (m, f), and then proceed with this hashed value. Indeed, security of the construction will be shown by replacing the obfuscated program with a corresponding obfuscated "punctured" program, with the punctured (challenge) input and output hardcoded; this hardcoded input corresponds to a challenge pair (m, f) in the naïve case, forcing the obfuscated program to grow in size with $|f|$, but is reduced to a short hashed value $h(m, f)$ in the latter case. Again, to provide security in this modified setting, we rely on extractability obfuscation. Signature queries m' with $h(m', f) \neq h(m, f)$ can be simulated in the security reduction given the punctured PRF key; any query of the form m' with $h(m', f) = h(m, f)$ yields a collision in h (and thus will not occur).

More explicitly, each program $R^{h,s} \in \mathcal{R}_k$ in our random oracle instantiation is an (extractability) obfuscation of a program $\Pi^{h,s}$, indexed by a PRF seed s and collision-resistant hash function h. $\Pi^{h,s}$ accepts as input a message m, and a description of the trapdoor function f (modeled as a polynomial-size circuit), and it functions by: (1) computing the hash of (m, f) with respect to h (using a Merkle-Damgard hash tree approach), (2) applying a PRF with hardcoded seed s to this hash value, and then (3) evaluating f on the resulting PRF output value x. That is,

$$\Pi^{h,s}(m, f) = U_k\left(f, \mathsf{PRF}_s(h(m, f))\right),$$

where U_k is the universal Turing machine, and

$$R^{h,s} \leftarrow e\mathcal{O}(1^k, \Pi^{h,s}).$$

Program $\Pi^{h,s}$:

Hardwired: Collision-resistant hash function $h : \{0,1\}^* \to \{0,1\}^{m(k)}$, PRF seed $s \in \{0,1\}^k$.

Inputs: Message m, circuit description f

1. Hash the input: $v = h(m, f)$.
2. Compute the PRF on this hash: $x = \mathsf{PRF}_s(v)$.
3. Evaluate the universal Turing machine on inputs f, x: i.e., $y = U_k(f, x)$.
4. Output y.

Figure 9: Program $\Pi^{h,s}$ that is obfuscated to form $R^{h,s} \in \mathcal{R}_k$.

Program $\Pi^{h,s}_{m,y}$:

Hardwired: Hash function $h : \{0,1\}^* \to \{0,1\}^{m(k)}$, punctured PRF seed $s^* \in \{0,1\}^k$ (punctured at point $h(m, f)$), bit string $y \in \{0,1\}^{\ell'(k)}$.

Inputs: Message m', circuit description f'

1. Hash the input: $v' = h(m', f')$.
2. If $v' \neq h(m, f)$, compute $x = \mathsf{PRF}_{s^*}(h)$, and output $U_k(f', x)$.
3. If $v' = h(m, f)$, output y.

Figure 10: "Punctured" program $\Pi^{h,s}_{m,y}$, used within the security proof.

(See Figure 9 for details). Denote by $\mathcal{M}$ the class of Turing machines

$$\mathcal{M} = \left\{ \Pi^{h,s} \;\middle|\; s \in \{0,1\}^k, h \in \mathcal{H}_k, k \in \mathbb{N} \right\}.$$

To prove security of the resulting FDH signature scheme, we consider a second, related class of ("punctured") Turing machines

$$\mathcal{M}^* = \left\{ \Pi^{h,s}_{m,y} \;\middle|\; s \in \{0,1\}^k, h \in \mathcal{H}_k, k \in \mathbb{N} \right\},$$

where each Turing machine $\Pi^{h,s}_{m,y}$ is defined as in Figure 10. The obfuscator we use is with respect to the class of Turing machines $\mathcal{M} \cup \mathcal{M}^*$.

Now, consider the instantiation of the full-domain hash signature scheme using a hash function $R^{h,s} \in \mathcal{R}_k$ in the place of a random oracle, and with an injective trapdoor function family $\mathcal{F}$.

$\mathsf{Setup}(1^k)$: On input the security parameter 1^k, the setup algorithm samples a program $R^{h,s} \leftarrow \mathcal{R}_k$ to be used as a random oracle. That is, it samples a random seed for a puncturable PRF $s \leftarrow \mathcal{F}_{\mathcal{PRF}}(1^k)$, and an underlying hash function $h \leftarrow \mathcal{H}_k$ from the collision-resistant hash function family. This uniquely defines a program $\Pi^{h,s}$. Then, the algorithm obfuscates this program as $R^{h,s} \leftarrow e\mathcal{O}(1^k, \Pi^{h,s})$.

The setup algorithm then samples $(f, f^{-1}) \leftarrow \mathsf{TDFSetup}(1^k)$ that produces a public index f and trapdoor f^{-1} (that allows inversion)

The verification key for the signature scheme is set to $\mathsf{vk} = (f, R^{h,s})$, consisting of the TDF description f and the "random oracle" $R^{h,s}$. The secret key sk is the trapdoor f^{-1} and $R^{h,s}$.

$\mathsf{Sign}(\mathsf{sk}, m)$: The signature algorithm outputs $\sigma = f^{-1}(R^{h,s}(m, f))$.

$\mathsf{Verify}(\mathsf{vk}, m, \sigma)$: The verification algorithm tests if $f(\sigma) \stackrel{?}{=} (R^{h,s}(m, f))$ and outputs accept if and only if this holds.

Theorem 4.2. *Assuming the existence of collision-resistant hash functions, and extractability obfuscation for the class* TM*, then for* any *injective trapdoor function family* $\mathcal{F}$*, the scheme described above is a selectively secure signature scheme.*

Proof. The proof follows an analogous sequence of hybrids as in Theorem 3.1, mirroring the approach of [HSW13]. Namely, for challenge forgery message m and injective trapdoor function f, the hybrids are (loosely) as follows:

Hybrid 0: The real (selective) security experiment.

Hybrid 1: The obfuscated program $R^{h,s} \leftarrow e\mathcal{O}(1^k, \Pi^{h,s})$ is replaced by an obfuscation of the corresponding "(m, f)-punctured" program $\Pi^{h,s}_{m,y}$ with the "correct" hardcoded output $y = \Pi^{h,s}(m, f)$. Indistinguishability follows by the security of the extractability obfuscator, together with the collision resistance of h. (Note that in the reduction, signature queries can be simulated using the inversion trapdoor to f).

Hybrid 2: The obfuscated program is replaced by an obfuscation of $\Pi^{h,s}_{m,y}$, with hardcoded output y set to a *random* evaluation of f: i.e., $f(u)$ for uniform u. Indistinguishability follows by the pseudo randomness of PRF. (Note that in the reduction, signature queries can again be simulated using the inversion trapdoor to f).

By the indistinguishability of hybrids, it follows that a forging adversary must continue to successfully forge in this final experiment, Hybrid 2. But, this implies the adversary can invert a random output of the trapdoor function, yielding the desired contradiction.

□

Acknowledgements

The authors would like to thank Kai-Min Chung for several insightful discussions.

References

[BBP04] Mihir Bellare, Alexandra Boldyreva, and Adriana Palacio. An uninstantiable random-oracle-model scheme for a hybrid-encryption problem. In *EUROCRYPT*, pages 171–188, 2004.

[BCCT12] Nir Bitansky, Ran Canetti, Alessandro Chiesa, and Eran Tromer. From extractable collision resistance to succinct non-interactive arguments of knowledge, and back again. In *ITCS*, pages 326–349, 2012.

[BCCT13] Nir Bitansky, Ran Canetti, Alessandro Chiesa, and Eran Tromer. Recursive composition and bootstrapping for snarks and proof-carrying data. In *STOC*, pages 111–120, 2013.

[BCP13] Elette Boyle, Kai-Min Chung, and Rafael Pass. On extractability obfuscation. Cryptology ePrint Archive, Report 2013/650, 2013.

[BCPR13] Nir Bitansky, Ran Canetti, Omer Paneth, and Alon Rosen. Indistinguishability obfuscation vs. auxiliary-input extractable functions: One must fall. Cryptology ePrint Archive, Report 2013/641, 2013.

[BG08] Boaz Barak and Oded Goldreich. Universal arguments and their applications. *SIAM J. Comput.*, 38(5):1661–1694, 2008.

[BGI+12] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. *J. ACM*, 59(2):6, 2012.

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

[BLV06] Boaz Barak, Yehuda Lindell, and Salil P. Vadhan. Lower bounds for non-black-box zero knowledge. *J. Comput. Syst. Sci.*, 72(2):321–391, 2006.

[BR93] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In Dorothy E. Denning, Raymond Pyle, Ravi Ganesan, Ravi S. Sandhu, and Victoria Ashby, editors, *ACM Conference on Computer and Communications Security*, pages 62–73. ACM, 1993.

[BR96] Mihir Bellare and Phillip Rogaway. The exact security of digital signatures - how to sign with rsa and rabin. In *EUROCRYPT*, pages 399–416, 1996.

[BV13] Zvika Brakerski and Vinod Vaikuntanathan. Lattice-based fhe as secure as pke. Cryptology ePrint Archive, Report 2013/541, 2013.

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

[BZ13] Dan Boneh and Mark Zhandry. Multiparty key exchange, efficient traitor tracing, and more from indistinguishability obfuscation. Cryptology ePrint Archive, Report 2013/642, 2013.

[CD08] Ran Canetti and Ronny Ramzi Dakdouk. Extractable perfectly one-way functions. In *ICALP (2)*, pages 449–460, 2008.

[CD09] Ran Canetti and Ronny Ramzi Dakdouk. Towards a theory of extractable functions. In Omer Reingold, editor, *TCC*, volume 5444 of *Lecture Notes in Computer Science*, pages 595–613. Springer, 2009.

[CGH04] Ran Canetti, Oded Goldreich, and Shai Halevi. The random oracle methodology, revisited. *J. ACM*, 51(4):557–594, 2004.

[Dam91] Ivan Damgård. Towards practical public key systems secure against chosen ciphertext attacks. In *CRYPTO*, pages 445–456, 1991.

[DFH12] Ivan Damgård, Sebastian Faust, and Carmit Hazay. Secure two-party computation with low communication. In *TCC*, pages 54–74, 2012.

[DOP05] Yevgeniy Dodis, Roberto Oliveira, and Krzysztof Pietrzak. On the generic insecurity of the full domain hash. In *CRYPTO*, pages 449–466, 2005.

[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.

[GGHR13] Sanjam Garg, Craig Gentry, Shai Halevi, and Mariana Raykova. Two-round secure mpc from indistinguishability obfuscation. *IACR Cryptology ePrint Archive*, 2013:601, 2013.

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

[GK03] Shafi Goldwasser and Yael Tauman Kalai. On the (in)security of the fiat-shamir paradigm. In *FOCS*, pages 102–, 2003.

[GKP+13] Shafi Goldwasser, Yael Tauman Kalai, Raluca A. Popa, Vinod Vaikuntanathan, and Nickolai Zeldovich. How to run turing machines on encrypted data. In *CRYPTO (2)*, pages 536–553, 2013.

[GLR11] Shafi Goldwasser, Huijia Lin, and Aviad Rubinstein. Delegation of computation without rejection problem from designated verifier cs-proofs. *IACR Cryptology ePrint Archive*, 2011:456, 2011.

[GMR89] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal on Computing*, 18(1):186–208, 1989.

[GS12] Divya Gupta and Amit Sahai. On constant-round concurrent zero-knowledge from a knowledge assumption. Cryptology ePrint Archive, Report 2012/572, 2012. http://eprint.iacr.org/.

[GSW13] Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In *CRYPTO (1)*, pages 75–92, 2013.

[HR04] Chun-Yuan Hsiao and Leonid Reyzin. Finding collisions on a public road, or do secure hash functions need secret coins? In *CRYPTO*, pages 92–105, 2004.

[HSW13] Susan Hohenberger, Amit Sahai, and Brent Waters. Replacing a random oracle: Full domain hash from indistinguishability obfuscation. Cryptology ePrint Archive, Report 2013/509, 2013. http://eprint.iacr.org/.

[HT98] Satoshi Hada and Toshiaki Tanaka. On the existence of 3-round zero-knowledge protocols. In Hugo Krawczyk, editor, *CRYPTO*, volume 1462 of *Lecture Notes in Computer Science*, pages 408–423. Springer, 1998.

[IKO05] Yuval Ishai, Eyal Kushilevitz, and Rafail Ostrovsky. Sufficient conditions for collision-resistant hashing. In *TCC*, pages 445–456, 2005.

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

[KS98] B. Kaliski and J. Staddon. PKCS #‘: RSA cryptography specifications version 2.0, 1998.

[Mic94] Silvio Micali. Cs proofs (extended abstracts). In *FOCS*, pages 436–453, 1994.

[SW13] Amit Sahai and Brent Waters. How to use indistinguishability obfuscation: Deniable encryption, and more. Cryptology ePrint Archive, Report 2013/454, 2013. http://eprint.iacr.org/.

[Val08] Paul Valiant. Incrementally verifiable computation or proofs of knowledge imply time/space efficiency. In Ran Canetti, editor, *TCC*, volume 4948 of *Lecture Notes in Computer Science*, pages 1–18. Springer, 2008.