Limits of Extractability Assumptions with Distributional Auxiliary Input

Elette Boyle[1(✉)] and Rafael Pass[2]

[1] Technion Israel, Haifa, Israel
eboyle@alum.mit.edu
[2] Cornell University, Ithaca, USA
rafael@cs.cornell.edu

Abstract. Extractability, or "knowledge," assumptions 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 (public-coin) differing-inputs obfuscation ((PC-)$di\mathcal{O}$), 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 public-coin collision-resistant hash functions, there exists an efficient distributions $\mathcal{Z}$ such that either

- PC-$di\mathcal{O}$ for Turing machines does not exist, or
- extractable one-way functions w.r.t. auxiliary input $\mathcal{Z}$ do not exist.

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

- SNARKs for NP w.r.t. auxiliary input $\mathcal{Z}$ do not exist, or
- PC-$di\mathcal{O}$ for NC^1 circuits does not exist.

E. Boyle—This work was primarily completed while at Cornell University, supported in part by AFOSR YIP Award FA9550-10-1-0093. The research of the first author has received funding from the European Union's Tenth Framework Programme (FP10/2010-2016) under grant agreement no. 259426 ERC-CaC, and ISF grant 1709/14. Part of this work was done while visiting the Simons Institute for the Theory of Computing, supported by the Simons Foundation and by the DIMACS/Simons Collaboration in Cryptography through NSF grant #CNS-1523467. Supported in part by the ERC under the EU's Seventh Framework Programme (FP/2007-2013) ERC Grant Agreement no. 307952.

R. Pass—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.

To achieve our results, we develop a "succinct punctured program" technique, mirroring the powerful punctured program technique of Sahai and Waters (STOC'14), and present several other applications of this new technique. In particular, we construct succinct perfect zero knowledge SNARGs and give a universal instantiation of random oracles in full-domain hash applications, based on PC-$di\mathcal{O}$.

As a final contribution, we demonstrate that *even in the absence of auxiliary input*, care must be taken when making use of extractability assumptions. We show that (standard) $di\mathcal{O}$ w.r.t. any distribution $\mathcal{D}$ over programs and bounded-length auxiliary input is directly implied by any obfuscator that satisfies the weaker indistinguishability obfuscation (i$\mathcal{O}$) security notion and $di\mathcal{O}$ for a slightly modified distribution $\mathcal{D}'$ of programs (of slightly greater size) and no auxiliary input. As a consequence, we directly obtain negative results for (standard) $di\mathcal{O}$ in the absence of auxiliary input.

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 differing-inputs obfuscation:

- **Extractable OWF:** An extractable family of one-way (resp. collision-resistant) functions [14,15,27], 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 [15] stipulates the existence of a particular such extractable one-way function.
- **SNARKs:** Succinct non-interactive arguments of knowledge (SNARKs) [5,32,35] 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.
- **Differing-Inputs Obfuscation:** [1,2,10] A differing-inputs obfuscator $\mathcal{O}$ for program-pair distribution $\mathcal{D}$ 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 programs C_1, C_2 generated via $\mathcal{D}$ given the randomness r used in sampling, then it must "know" an input x such that $C_1(x) \neq C_2(x)$, and this input can be efficiently "extracted" from $\mathcal{A}$.

A recently proposed (weaker) variant known as *public-coin differing-inputs obfuscation* [30] additionally provides the randomness used to sample the programs $(C_0, C_1) \leftarrow \mathcal{D}$ to the extraction algorithm (and to the attacker $\mathcal{A}$).

The above primitives have proven extremely useful in constructing cryptographic tools for which instantiations under complexity-theoretic hardness assumptions are not known (e.g., [1,5,10,16,24,27,30]).

Extraction with (Distribution-Specific) Auxiliary Input. In all of these applications, we require a notion of an *auxiliary-input* extractable one-way function [14,27], 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 [27], extractability w.r.t. an arbitrary auxiliary input is an "overly strong" (or in the language of [27], "unreasonable") assumption. Indeed, a recent result of Bitansky, Canetti, Rosen and Paneth [7] (formalizing earlier intuitions from [5,27]) 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. [5,6], it 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 $\{0,1\}^k \rightarrow \{0,1\}^{\ell(k)}$ with respect to $\mathcal{Z}$. In fact, for some of these results (e.g., [5,6]), it suffices to just assume that extraction works for just for the *uniform* distribution.

In contrast, the result of [7] can be interpreted as saying that (assuming $i\mathcal{O}$), 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.

[1] The notion of indistinguishability obfuscation [2] 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. [18].

[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 [26,27], to acheive "plain" zero-knowledge [3,25] (where the verifier does not receive any auxiliary input), weaker "bounded" auxiliary input assumptions suffice.

Our Results. In this paper, we show limitations of extractability primitives with respect to *distribution-specific* auxiliary input (assuming the existence of public-coin collision-resistant hash functions (CRHF) [29]). Our main result shows a conflict between public-coin differing-inputs obfuscation for Turing machines [30] and extractable one-way functions.

Theorem 1 (Main Theorem – Informal). *Assume the existence of public-coin collision-resistant hash functions. Then for every polynomial ℓ, there exists an efficiently computable distribution $\mathcal{Z}$ such that one of the following two primitives does not exist:*

- *extractable one-way functions $\{0,1\}^k \rightarrow \{0,1\}^{\ell(k)}$ w.r.t. auxiliary input from $\mathcal{Z}$.*
- *public-coin differing-inputs obfuscation for Turing machines.*

By combining our main theorem with results from [5,30], we obtain the following corollary:

Theorem 2 (Informal). *Assume the existence of public-coin CRHF and fully homomorphic encryption with decryption in NC^1.*[3] *Then there exists an efficiently computable distribution $\mathcal{Z}$ such that one of the following two primitives does not exist:*

- *SNARKs w.r.t. auxiliary input from $\mathcal{Z}$.*
- *public-coin differing-inputs obfuscation for NC^1 circuits.*

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 [34]; see Sect. 1.1 for more details. This technique has several other interesting applications, as we discuss in Sect. 1.3.

As a final contribution, we demonstrate that *even in the absence of auxiliary input*, care must be taken when making use of extractability assumptions. Specifically, we show that differing-inputs obfuscation (*di$\mathcal{O}$*) for any distribution $\mathcal{D}$ of programs and bounded-length auxiliary inputs, is directly implied by any obfuscator that satisfies a weaker indistinguishability obfuscation (*i$\mathcal{O}$*) security notion (which is not an extractability assumption) and *di$\mathcal{O}$* security for a related distribution $\mathcal{D}'$ of programs (of slightly greater size) which does *not contain auxiliary input*. Thus, negative results ruling out existence of *di$\mathcal{O}$* with bounded-length auxiliary input directly imply negative results for *di$\mathcal{O}$* in a setting without auxiliary input.

Theorem 3 (Informal). *Let $\mathcal{D}$ be a distribution over pairs of programs and ℓ-bounded auxiliary input information $\mathcal{P} \times \mathcal{P} \times \{0,1\}^\ell$. There exists di$\mathcal{O}$ with respect to $\mathcal{D}$ if there exists an obfuscator satisfying i$\mathcal{O}$ in addition to di$\mathcal{O}$ with respect to a modified distribution $\mathcal{D}'$ over $\mathcal{P}' \times \mathcal{P}'$ for slightly enriched program class $\mathcal{P}'$, and* no auxiliary input.

[3] As is the case for nearly all existing FHE constructions (e.g., [13,21]).

Our transformation applies to a recent result of Garg *et al.* [20], which shows that based on a new assumption (pertaining to special-purpose obfuscation of Turing machines) general-purpose $di\mathcal{O}$ w.r.t. auxiliary input cannot exist, by constructing a distribution over circuits and bounded-length auxiliary inputs for which no obfuscator can be $di\mathcal{O}$-secure. Our resulting conclusion is that, assuming such special-purpose obfuscation exists, then general-purpose $di\mathcal{O}$ cannot exist, *even in the absence of auxiliary input.*

We view this as evidence that public-coin differing inputs may be the "right" approach definitionally, as restrictions on auxiliary input without regard to the programs themselves will not suffice.

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, and in certain cases even in the absence of auxiliary input. In particular, we must develop a way to distinguish "good" distributions of instances and 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.

Comparison to [20]. An interesting subsequent[5] work of Garg *et al.* [19,20] contains a related study of differing-inputs obfuscation. In [20], the authors propose a new "special-purpose" circuit obfuscation assumption, and demonstrate based on this assumption an auxiliary input distribution (whose size grows with the desired circuit size of circuits to be obfuscated) for which general-purpose $di\mathcal{O}$ cannot exist. Using similar techniques of hashing and obfuscating Turing machines as in the current work, they further conclude that if the new obfuscation assumption holds also for *Turing machines*, then the "bad" auxiliary input distribution can have bounded length (irrespective of the circuit size).

Garg *et al.* [20] show the "special-purpose" obfuscation assumption is a falsifiable assumption (in the sense of [33]) and is implied by virtual black-box obfuscation for the relevant restricted class of programs, but plausibility of the notion in relation to other primitives is otherwise unknown. In contrast, our results provide a direct relation between existing, studied topics (namely, $di\mathcal{O}$, EOWFs, and SNARKs). Even in the case that the special-purpose obfuscation assumption *does* hold, our primary results provide conclusions for *public-coin* $di\mathcal{O}$, whereas Garg *et al.* [20] consider (stronger) standard $di\mathcal{O}$, with respect to auxiliary input.

[4] Note that this is not the case for all applications; e.g. [11,23,26,27] require considering more complicated distributions.

[5] A version of our paper with Theorems 1 and 2 for (standard) differing-inputs obfuscation in the place of public-coin $di\mathcal{O}$ has been on ePrint since October 2013 [12].

And, utilizing our final observation (which occurred subsequent to [20]), we show that based on their same special-purpose obfuscation assumption for Turing machines, we can in fact rule out general-purpose *di$\mathcal{O}$* for circuits even *in the absence of auxiliary input.*

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 [27], 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. [5] 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 [2]), 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. [2]), for large classes of functions VBB program obfuscation simply does not exist.

The Work of [7] *and the "Punctured Program" Paradigm of* [34]. Intriguingly, Bitansky, Canetti, Rosen and Paneth [7] 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 [34] (and the closely related work of Hohenberger, Sahai and Waters [28] on "instantiating random oracles"). Roughly speaking, the punctured program paradigm shows that if we use indistinguishability obfuscation to obfuscate a (function of) a special kind of "puncturable" PRF[6] [8,11,31], we can still "program" the output of the program on *one* input (which was used in [28,34] to show various applications of indistinguishability obfuscation). Bitansky et al. [7] 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 [28,34], 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 $f_j \in F$.

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

[6] 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.

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 $\mathcal{F}$. 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 *differing-inputs 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 a differing-inputs 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 *differing-inputs 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 using just public-coin $di\mathcal{O}$, we require that h is a *public-coin* collision-resistant hash function [29].

1.2 Removing Auxiliary Input in $di\mathcal{O}$

The notion of public-coin $di\mathcal{O}$ is weaker than "general" (not necessarily public-coin) $di\mathcal{O}$ in two aspects: (1) the programs M_0, M_1 are sampled using only public randomness, and (2) we consider only a very specific auxiliary input that is given to the attacker—namely the randomness of the sampling procedure.

In this section, we explore another natural restriction of $di\mathcal{O}$ where we simply disallow auxiliary input, but allow for "private" sampling of M_0, M_1. We show that "bad side information" cannot be circumvented simply by simply disallowing auxiliary input, but rather such information can appear in the *input-output behavior* of the programs to be obfuscated.

More precisely, we show that for any distribution $\mathcal{D}$ over $\mathcal{P} \times \mathcal{P} \times \{0,1\}^\ell$ of programs $\mathcal{P}$ and bounded-length auxiliary input, the existence of $di\mathcal{O}$ w.r.t. $\mathcal{D}$ is directly implied by the existence of any *indistinguishability* obfuscator ($i\mathcal{O}$) that is $di\mathcal{O}$-secure for a slightly enriched distribution of programs $\mathcal{D}'$ over $\mathcal{P}' \times \mathcal{P}'$, *without* auxiliary input.

Intuitively, this transformation works by embedding the "bad auxiliary input" into the *input-output behavior* of the circuits to be obfuscated themselves. That is, the new distribution $\mathcal{D}'$ is formed by sampling first a triple (P_0, P_1, z) of programs and auxiliary input from the original distribution $\mathcal{D}$, and then instead considering the tweaked programs P_0^z, P_1^z that have a special additional input x^* (denoted later as "$\mathsf{mode} = *$") for which $P_0^z(x^*) = P_1^z(x^*)$ is defined to be z. This introduces no new differing inputs to the original program pair P_0, P_1, but now there is no hope of preventing the adversary from learning z without sacrificing correctness of the obfuscation scheme.

A technical challenge arises in the security reduction, however, in which we must modify the obfuscation of the z-embedded program P_b^z to "look like" an obfuscation of the original program P_b. Interestingly, this issue is solved by making use of a *second layer* of obfuscation, and is where the $i\mathcal{O}$ security of the obfuscator is required. We refer the reader to the full version of this work for details.

1.3 Other Applications of the "Succinct Punctured Program" Technique

As mentioned above, the "punctured program" paradigm of [34] has been used in multiple applications (e.g., [9,17,28,34]). Many of them rely on punctured programs in an essentially identical way to the approach described above, and in particular follow the same hybrids within the security proof. 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 public-coin differing-inputs obfuscation (and the existence of public-coin collision-resistant hash functions), by using our succinct punctured program technique, we can obtain significant improvements. For instance, 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 [34].
– A *universal* instantiation of Random Oracles, for which the Full Domain Hash (FDH) signature paradigm [4] 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 [28] showing how to provide a trapdoor-function specific instantiation of the random oracle in the FDH.[7]

1.4 Overview of Paper

We focus in this extended abstract on the primary result: the conflict between public-coin differing inputs obfuscation and extractable OWFs (and SNARKs). Further preliminaries, applications of our succinct punctured programs technique, and our transformation removing auxiliary input in differing-inputs obfuscation are deferred to the full version [12].

2 Preliminaries

2.1 Public-Coin Differing-Inputs Obfuscation

The notion of public-coin differing-inputs obfuscation (PC-$di\mathcal{O}$) was introduced by Ishai *et al.* [30] as a refinement of (standard) differing-inputs obfuscation [2] to exclude certain cases whose feasibility has been called into question. (Note that we also consider "standard" differing-inputs obfuscation as described in Sect. 1.2. For a full treatment of the notion and our result, we refer the reader to the full version of this work [12]).

We now present the PC-$di\mathcal{O}$ definition of [30], focusing only on Turing machine obfuscation; the definition easily extends also to circuits.

Definition 1 (Public-Coin Differing-Inputs Sampler for TMs). *An efficient non-uniform sampling algorithm* $\mathsf{Samp} = \{\mathsf{Samp}_k\}$ *is called a* public-coin differing inputs sampler *for the parameterized collection of TMs* $\mathcal{M} = \{\mathcal{M}_k\}$ *if the output of* Samp_k *is always a pair of Turing machines* $(M_0, M_1) \in \mathcal{M}_k \times \mathcal{M}_k$

[7] That is, [28] 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 contrast, our results shows that there exists a single instantiation that works no matter what the trapdoor function is.

such that $|M_0| = |M_1|$ *and for every efficient non-uniform algorithm* $\mathcal{A} = \{\mathcal{A}_k\}$ *there exists a negligible function* ϵ *such that for all* $k \in \mathbb{N}$,

$$\Pr\Big[r \leftarrow \{0,1\}^*; (M_0, M_1) \leftarrow \mathsf{Samp}_k(r); (x, , 1^t) \leftarrow \mathcal{A}_k(r) \\ : \big(M_0(x) \neq M_(x)\big) \wedge \big(\mathsf{steps}(M_0, x) = \mathsf{steps}(M_1, x)\big)\Big] \leq \epsilon(k).$$

Definition 2 (Public-Coin Differing-Inputs Obfuscator for TMs). *A uniform PPT algorithm* $\mathcal{O}$ *is a* public-coin differing-inputs obfuscator for the collection $\mathcal{M} = \{\mathcal{M}_k\}$ *if the following requirements hold:*

- **Correctness:** *For every* $k \in \mathbb{N}$, *every* $M \in \mathcal{M}_k$, *and every* x, *we have that* $\Pr[\tilde{M} \leftarrow \mathcal{O}(1^k, M) : \tilde{M}(x) = M(x)] = 1$.
- **Security:** *For every public-coin differing-inputs sampler* $\mathsf{Samp} = \{\mathsf{Samp}_k\}$ *for the ensemble* $\mathcal{M}$, *every efficient non-uniform distinguishing algorithm* $\mathcal{D} = \{\mathcal{D}_k\}$, *there exists a negligible function* ϵ *such that for all* k,

$$\big|\Pr[r \leftarrow \{0,1\}^*; (M_0, M_1) \leftarrow \mathsf{Samp}_k(r); \tilde{M} \leftarrow \mathcal{O}(1^k, M_0) : \mathcal{D}_k(r, \tilde{M}) = 1] - \\ \Pr[r \leftarrow \{0,1\}^*; (M_0, M_1) \leftarrow \mathsf{Samp}_k(r); \tilde{M} \leftarrow \mathcal{O}(1^k, M_1) : \mathcal{D}_k(r, \tilde{M}) = 1]\big| \leq \epsilon(k).$$

2.2 Extractable One-Way Functions

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

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

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 poly-time* $\mathcal{A}$ *and sufficiently large* $k \in \mathbb{N}$,

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

- **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* $k \in \mathbb{N}$:

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

2.3 Succinct Non-Interactive Arguments of Knowledge (SNARKs)

We focus attention to *publicly verifiable* succinct arguments. We consider succinct non-interactive arguments of knowledge (SNARKs) with adaptive soundness in Sect. 3.2, and consider the case of specific distributional auxiliary input.

Definition 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*

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

In the full version of this work, we obtain as an application of our succinct programs technique zero-knowledge (ZK) succinct non-interactive arguments (SNARGs), without the extraction property. We refer the reader to [12] for a full treatment.

2.4 Puncturable PRFs

Our result makes use of puncturable PRFs, which are PRFs with an extra capability to generate keys that allow one to evaluate the function 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: that is, given a punctured key k^* with respect to input x, one can efficiently evaluate the PRF at all points *except* x, whose evaluation remains pseudorandom. We refer the reader to [34] for a formal definition.

As observed in [8,11,31], the GGM tree-based PRF construction [22] yields puncturable PRFs, based on any one-way function.

Theorem 4 ([8,11,31]). *If one-way functions exist, then for all efficiently computable* $m'(k)$ *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 Public-Coin Differing-Inputs Obfuscation or Extractable One-Way Functions

In this section, we present our main result: a conflict between extractable one-way functions (EOWF) w.r.t. a particular distribution of auxiliary information and public-coin differing-inputs obfuscation ("$\mathsf{PC}-di\mathcal{O}$") (for Turing Machines).

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

We demonstrate a bounded polynomial-time uniformly samplable distribution $\mathcal{Z}$ (with bounded poly-size output length) and a public-coin differing-inputs sampler for Turing Machines $\mathcal{D}$ (over $\mathsf{TM} \times \mathsf{TM}$) such that if there exists public-coin differing-inputs obfuscation for Turing machines (and, in particular, for the program sampler $\mathcal{D}$), and there exist public-coin collision-resistant hash functions (CRHF), then there do *not* exist extractable one-way functions (EOWF) w.r.t. auxiliary information sampled from distribution $\mathcal{Z}$. In our construction, $\mathcal{Z}$ consists of an obfuscated Turing machine.

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 [7], 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 5. *For every polynomial ℓ, there exists an efficient, uniformly samplable distribution $\mathcal{Z}$ such that, assuming the existence of public-coin collision-resistant hash functions and public-coin differing-inputs 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 public-coin CRHFs and $\mathsf{PC} - di\mathcal{O}$).

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 public-coin 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 4, 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.

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 w.r.t. 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 Fig. 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.

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

Hardwired: Hash function $h : \{0,1\}^* \rightarrow \{0,1\}^{m(k)}$, 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. Output the evaluation of the universal Turing machine on inputs f_i, x: i.e., $y = U_k(f_i, x)$.

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

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

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

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

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 $\Pi^{h,s}_{i,y}$ first computes the hash $h = h(j)$, continues computation as usual for any $h \neq h(i)$ using the punctured PRF key, and for $h = h(i)$, it skips the PRF and U_k evaluation steps and directly outputs y. Note that because 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 set to $y = f_i(\mathsf{PRF}_s(h(i)))$, then $\Pi^{h,s}_{i,y}$ 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 first remark that indistinguishability obfuscation arguments will thus not apply to this scenario, since we are modifying the computed functionality. In contrast, differing-inputs obfuscation would guarantee that the two obfuscated programs are indistinguishable, since otherwise we could efficiently *find* one of the disagreeing inputs, which would correspond to a collision in the CRHF. But, most importantly, this argument holds even if *the randomness used to sample the program pair* $(\Pi^{h,s}, \Pi^{h,s}_{i,y})$ *is revealed.* Namely, we consider a program sampler that generates pairs $(\Pi^{h,s}, \Pi^{h,s}_{i,y})$ of the corresponding distribution; this amounts to sampling a hash function h, an EOWF challenge index i, and a PRF seed s, and a $h(i)$-puncturing of the seed, s^*. All remaining values specifying the programs, such as $y = f_i(\mathsf{PRF}_s(h(i)))$, are deterministically computed given (h, i, s, s^*). Now, since $\mathcal{H}$ is a public-coin CRHF family, revealing the randomness used to sample $h \leftarrow \mathcal{H}$ is not detrimental to its collision resistance. And, the values i, s, and s^* are completely *independent* of the CRHF security (i.e., a CRHF adversary reduction could simply generate them on its own in order to break h). Therefore, we ultimately need only rely on *public-coin di$\mathcal{O}$*.

We finally consider the size of the program(s) to be obfuscated. 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 4 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 auxiliary input distribution $\mathcal{Z} = \{\mathcal{Z}_k\}$, described in full in Fig. 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 3:

```
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 $y$.
```

Fig. 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 $h \leftarrow \mathcal{H}_k$ and PRF seed $s \leftarrow \mathcal{K}_{\mathcal{PRF}}(1^k)$.
    2. Sample a punctured PRF seed $s^* \leftarrow \mathsf{Punct}(s, h(i))$, punctured at point $h(i)$.
    3. Compute the "correct" punctured evaluation: $y = f_i(\mathsf{PRF}_s(h(i)))$.
    4. Output an obfuscation $\tilde{M} \leftarrow \mathsf{PC}\text{-}di\mathcal{O}(\Pi^{h,s}_{i,y})$, where $\Pi^{h,s}_{i,y}$ is defined from $(h, s^*, y)$,
       as in Figure 3.
```

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

Proposition 1. *For any non-uniform polynomial-time candidate extractor $\mathcal{E}$ for $\mathcal{A}$, it holds that $\mathcal{E}$ fails with overwhelming probability: i.e.,*

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

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,

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

Indeed, by the definition of $\mathcal{A}$ and $\mathcal{Z}_k$, and the correctness of the obfuscator $\mathsf{PC} - di\mathcal{O}$, 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., $z = \tilde{M} \leftarrow \mathsf{PC} - di\mathcal{O}(\Pi^{h,s})$.

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 $\mathcal{A}(i;z) \in Range(f_i)$ 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 $z = \mathsf{PC} - di\mathcal{O}(\Pi^{h,s})$. 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 Fig. 3 from the values (h, s^*, y): i.e., $\tilde{M} \leftarrow \mathsf{PC} - di\mathcal{O}(\Pi^{h,s}_{i,y})$. See Fig. 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 $\mathcal{Z}_k$.

Consider the Turing Machine program sampler algorithm Samp as in Fig. 5.

Program Pair Sampler $\mathsf{Samp}(1^k, r)$:

1. Sample a hash function $h = \mathcal{H}_k(r_h)$.
2. Sample an EOWF index $i = K_{\mathcal{F}}(1^k; r_i)$.
3. Sample a PRF seed $s = K_{\mathsf{PRF}}(1^k; r_s)$.
4. Sample a punctured PRF seed $s^* = \mathsf{Punct}(s, h(i); r_*)$.
5. Let $y = f_i(\mathsf{PRF}_s(h(i)))$.
6. Denote $r := (r_h, r_i, r_s, r_*)$.
7. Output program pair $(\Pi^{h,s}, \Pi^{h,s}_{i,y})$, defined by h, i, s, s^*, y as above (and padded to equal length).

Fig. 5. Program pair sampler algorithm, to be used in public-coin differing inputs security step.

We first argue that, by the (public-coin) collision resistance of the hash family $\mathcal{H}$, the sampler algorithm Samp is a *public-coin differing-inputs sampler*, as per Definition 1.

Claim. Samp is a public-coin differing-inputs sampler. That is, for all efficient non-uniform $\mathcal{A}_{\mathsf{PC}}$, there exists a negligible function ϵ such that for all $k \in \mathbb{N}$,

$$\Pr\big[r \leftarrow \{0,1\}^*; (M_0, M_1) \leftarrow \mathsf{Samp}(1^k, r); (x, 1^t) \leftarrow \mathcal{A}_{\mathsf{PC}}(1^k, r) : \\ M_0(x) \neq M_1(x) \wedge \mathsf{steps}(M_0, x) = \mathsf{steps}(M_1, x) = t\big] \leq \epsilon(k). \quad (1)$$

Proof. Suppose, to the contrary, there exists an efficient (non-uniform) adversary $\mathcal{A}_{\mathsf{PC}}$ and non-negligible function $\alpha(k)$ for which the probability in Eq. 1 is greater than $\alpha(k)$. We show such an adversary contradicts the security of the (public-coin) CRHF. Consider an adversary $\mathcal{A}_{\mathsf{CR}}$ in the CRHF security challenge. Namely, for a challenge hash function $h \leftarrow \mathcal{H}_k(r_h)$, the adversary $\mathcal{A}_{\mathsf{CR}}$ receives h, r_h, and performs the following steps:

CRHF adversary $\mathcal{A}_{\mathsf{CR}}(1^k, h, r_h)$:

1. Imitate the remaining steps of Samp. That is, sample an EOWF index $i = K_{\mathcal{F}}(1^k; r_i)$; a PRF seed $s = K_{\mathsf{PRF}}(1^k; r_s)$; and a punctured PRF seed $s^* = \mathsf{Punct}(s, h(i); r_*)$. Define $y = f_i(\mathsf{PRF}_s(h(i)))$ and $r = (r_h, r_i, r_s, r_*)$, and let $M_0 = \Pi^{h,s}$ and $M_1 = \Pi^{h,s}_{i,y}$.
2. Run $\mathcal{A}_{\mathsf{PC}}(1^k, r)$ on the collection of randomness r used above. In response, $\mathcal{A}_{\mathsf{PC}}$ returns a pair $(x, 1^t)$.
3. $\mathcal{A}_{\mathsf{CR}}$ outputs the pair (i, x) as an alleged collision in the challenge hash function h.

Now, by assumption, the value x generated by $\mathcal{A}_{\mathsf{PC}}$ satisfies (in particular) that $M_0(x) \neq M_1(x)$. From the definition of M_0, M_1 (i.e., $\Pi^{h,s}, \Pi^{h,s}_{i,y}$), this must mean that $h(i) = h(x)$ (since all values with $h(x) \neq h(i)$ were not changed from $\Pi^{h,s}$ to $\Pi^{h,s}_{i,y}$), *and* that $i \neq x$ (since $\Pi^{h,s}_{i,y}(i)$ was specifically "patched" to the correct output value $\Pi^{h,s}(i)$). That is, $\mathcal{A}_{\mathsf{CR}}$ successfully identifies a collision with the same probability $\alpha(k)$, which must thus be negligible.

We now show that this implies, by the security of the public-coin $di\mathcal{O}$, that our original EOWF extractor $\mathcal{E}$ must succeed with nearly equivalent probability in the EOWF challenge when instead of receiving (real) auxiliary input from $\mathcal{Z}_k$, both $\mathcal{E}$ and $\mathcal{A}$ are given auxiliary input from the fake distribution $\mathcal{Z}_k(i, y)$. (Recall that ϵ is assumed to be $\mathcal{E}$'s success in the same experiment as below but with $z \leftarrow \mathcal{Z}_k$ instead of $z^* \leftarrow \mathcal{Z}_k(i, y)$).

Lemma 1. *It holds that*

$$\Pr\Big[i \leftarrow \mathcal{K}_{\mathcal{F}}(1^k);\; z^* \leftarrow \mathcal{Z}_k(i, y);\; x' \leftarrow \mathcal{E}(i; z^*) : \\ f_i(x') = \mathcal{A}(i; z^*) = f_i(\mathsf{PRF}_s\,(h(i)))\Big] \geq \epsilon(k) - \mathsf{negl}(k). \quad (2)$$

Proof. Note that given $z^* \leftarrow \mathcal{Z}_k(i, y)$ (which corresponds to an obfuscated program of the form $\Pi^{h,s}_{i,y}$) our EOWF adversary $\mathcal{A}$ indeed will still output $\Pi^{h,s}_{i,y}(i) = y := f_i(\mathsf{PRF}_s(h(i)))$ (see Figs. 3,4).

Now, suppose there exists a non-negligible function $\alpha(k)$ for which the probability in Eq. (2) is less than $\epsilon(k) - \alpha(k)$. We directly use such $\mathcal{E}$ to design another adversary $\mathcal{A}_{di\mathcal{O}}$ to contradict the security of the public-coin $di\mathcal{O}$ with respect to the program pair sampler Samp (which we showed in Claim 3.1 to be a void public-coin differing inputs sampler). Recall the $di\mathcal{O}$ challenge samples a program pair $(\Pi^{h,s}, \Pi^{h,s}_{i,y}) \leftarrow \mathsf{Samp}(1^k, r)$, selects a random $M \leftarrow \{\Pi^{h,s}, \Pi^{h,s}_{i,y}\}$ to obfuscate as $\tilde{M} \leftarrow \mathsf{PC} - di\mathcal{O}(1^k, M)$, and gives as a challenge the pair $(r, \tilde{M})$ of the randomness used by Samp and obfuscated program. Define $\mathcal{A}_{di\mathcal{O}}$ (who wishes to distinguish which program was selected) as follows.

PC-$di\mathcal{O}$ adversary $\mathcal{A}_{di\mathcal{O}}(1^k, r, \tilde{M})$:

1. Parse the given randomness r used in Samp as $r = (r_h, r_i, r_s, r_*)$ (see Fig. 5).
2. Recompute the "challenge index" $i = K_{\mathcal{F}}(1^k; r_i)$. Let $z^* = \tilde{M}$.
3. Run the extractor algorithm $\mathcal{E}(i; z^*)$, and receive an alleged preimage x'.
4. Recompute $h = \mathcal{H}_k(r_h)$, $s = K_{\mathsf{PRF}}(1^; r_s)$, again using the randomness from r.
5. If $f_i(x') = f_i(\mathsf{PRF}_s(h(i)))$ — i.e., if $\mathcal{E}$ succeeded in extracting a preimage — then $\mathcal{A}_{di\mathcal{O}}$ outputs 1. Otherwise, $\mathcal{A}_{di\mathcal{O}}$ outputs 0.

Now, if $\tilde{M}$ is an obfuscation of $\Pi^{h,s}$, then this experiment corresponds directly to the EOWF challenge where $\mathcal{E}$ (and $\mathcal{A}$) is given auxiliary input $z \leftarrow \mathcal{Z}_k$. On the other hand, if $\tilde{M}$ is an obfuscation of $\Pi^{h,s}_{i,y}$, then the experiment corresponds directly to the same challenge where $\mathcal{E}$ (and $\mathcal{A}$) is given auxiliary input $z^* \leftarrow \mathcal{Z}_k(i, y)$. Thus, $\mathcal{A}_{di\mathcal{O}}$ will succeed in distinguishing these two cases with probability at least $[\epsilon(k)] - [\epsilon(k) - \alpha(k)] = \alpha(k)$. By the security of $\mathsf{PC} - di\mathcal{O}$, it hence follows that $\alpha(k)$ must be negligible.

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 $\Pi^{h,s}_{i,y}$, 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 2. *It holds that*

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

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

1. Take $y = f_i(x)$, and obfuscate the associated program $\Pi^{h,s}_{i,y}$: i.e., $z^{**} \leftarrow \mathsf{PC} - di\mathcal{O}(1^k, \Pi^{h,s}_{i,y})$.
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 2 does not hold: i.e., the probability in Eq. (3) 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

$$\Pr\Big[i \leftarrow \mathcal{K}_{\mathcal{F}}(1^k);\ h \leftarrow \mathcal{H}_k;\ s \leftarrow \mathcal{K}_{\mathcal{PRF}}(1^k);\ s^* \leftarrow \mathsf{Punct}(s, h(i));$$
$$u \leftarrow \{0,1\}^k; b \leftarrow \{0,1\} : \mathcal{A}^0_{\mathsf{PRF}}(1^k, i, h, x_b) = b\Big] \geq \frac{1}{2} + \alpha(k), \quad (4)$$

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

$$\Pr\Big[s \leftarrow \mathcal{K}_{\mathcal{PRF}}(1^k);\ s^* \leftarrow \mathsf{Punct}(s, h(i_k));\ u \leftarrow \{0,1\}^k;$$
$$b \leftarrow \{0,1\} : \mathcal{A}^0_{\mathsf{PRF}}(1^k, i_k, h, x_b) = b\Big] \geq \frac{1}{2} + \frac{1}{p(k)}, \quad (5)$$

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 (5), 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 $z^{**} \leftarrow \mathsf{PC} - di\mathcal{O}(\Pi^{h,s}_{i,y})$. 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 5.

3.2 PC-$di\mathcal{O}$ or SNARKs

We link the existence of public-coin differing-inputs 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 [5]. Namely, *assume the existence of* *fully homomorphic encryption (FHE) with decryption in NC^1 and public-coin collision-resistant hash functions.* Then, building upon the results of the previous subsection, and the results of [5,30], we show:

1. Assuming SNARKs for NP, there exists an efficient distribution $\mathcal{Z}$ such that public-coin differing-inputs obfuscation for NC^1 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 second efficient auxiliary input distribution $\mathcal{Z}'$, as shown in [5].
3. Thus, one of these conflicting hypotheses must be false. That is, there exists an efficient distribution $\mathcal{Z}'$ such that assuming existence of FHE with decryption in NC^1 and collision-resistant hash functions, then either: (1) public-coin differing-inputs obfuscation for NC^1 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 $\ell(k)$).

Proximity EOWFs. We begin by defining Proximity EOWFs.

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 $\mathcal{E}$ finding an input x s.t. $f(x) = (a,b')$ 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 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 $k \in \mathbb{N}$,*

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

- **(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 $k \in \mathbb{N}$,*

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

- **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 $y \sim_k y'$.*

(PC – $di\mathcal{O}$ for NC^1 + PC-CRHF + FHE + SNARK) $\Rightarrow$ *No* $\mathcal{Z}$-PEOWF. We now show that, assuming the existence of public-coin collision-resistant hash functions (CRHF) and fully homomorphic encryption (FHE) with decryption in NC^1,[8] then for some efficiently computable distributions $\mathcal{Z}_{\mathsf{SNARK}}$, $\mathcal{Z}_{\mathsf{PEOWF}}$, if there exist public-coin differing-inputs obfuscators for NC^1 circuits, 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.

First, we remark that an identical proof to that of Theorem 5 rules out the existence of $\mathcal{Z}$-auxiliary-input *proximity EOWFs* in addition to standard EOWFs, based on the same assumptions: namely, assuming public-coin differing-inputs obfuscation for Turing machines, and public-coin collision-resistant hash functions. 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 differing-inputs obfuscation scheme, in Step 1). However, this is directly implied by the public testability of the PEOWF relation $\sim$, as specified in Definition 5.

[8] As is the case for nearly all existing FHE constructions (e.g., [13,21]).

Theorem 6. *There exist an efficient, uniformly samplable distribution $\mathcal{Z}$ such that, assuming the existence of public-coin collision-resistant hash functions and public-coin differing-inputs 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 [30], it was shown that public-coin differing-inputs obfuscation for the class of all polynomial-time Turing machines can be achieved by bootstrapping up from public-coin differing-inputs obfuscation for circuits in the class NC^1, assuming the existence of FHE with decryption in NC^1, public-coin CRHF, and public-coin SNARKs for NP.

Putting this together with Theorem 6, we thus have the following corollary.

Corollary 1. *There exists an efficient, uniformly samplable distribution $\mathcal{Z}$ s.t., assuming existence of public-coin SNARKs and FHE with decryption in NC^1, then assuming the existence of public-coin differing-inputs obfuscation for NC^1, there cannot exist PEOWFs $\{f_i : \{0,1\}^k \to \{0,1\}^k\}$ w.r.t. auxiliary input $\mathcal{Z}$.*

(SNARK + CRHF) $\implies$ $\mathcal{Z}$-PEOWF. As shown in [5], 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 $\mathcal{Z}'$.[9]

Loosely, the transformation converts any CRHF 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

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

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 [5]), 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 $\mathcal{Z}_{\mathsf{SNARK}} := (\mathcal{Z}, h)$, formed by concatenating a sample from $\mathcal{Z}$ with a function index h sampled from the collision-resistant hash function family $\mathcal{F}$. (Note that we will be considering public-coin CRHF, in which case h is uniform).

Theorem 7 ([5]). *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. $\mathcal{Z}$.*

[9] [5] consider the setting of arbitrary auxiliary input; however, their construction directly implies similar results for specific auxiliary input distributions.

Reaching a Standoff. Observe that the conclusions of Corollary 1 and Theorem 7 are in direct contradiction. Thus, it must be that one of the two sets of assumptions is false. Namely,

Corollary 2. *Assuming the existence of public-coin collision-resistant hash functions and fully homomorphic encryption with decryption in* NC^1*, there exists an efficiently samplable distribution* $\mathcal{Z}_{\mathsf{SNARK}}$ *such that one of the following two objects cannot exist:*

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

More explicitly, we have that $\mathcal{Z}_{\mathsf{SNARK}} = (\mathcal{Z}, U)$, where $\mathcal{Z}$ is composed of an obfuscated program, and U is a uniform string (corresponding to a randomly sampled index from a public-coin CRHF family).

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

References

1. Ananth, P., Boneh, D., Garg, S., Sahai, A., Zhandry, M.: Differing-inputs obfuscation and applications. Cryptology ePrint Archive, Report 2013/689 (2013)
2. Barak, B., Goldreich, O., Impagliazzo, R., Rudich, S., Sahai, A., Vadhan, S.P., Yang, K.: On the (im)possibility of obfuscating programs. J. ACM **59**(2), Article No. 6 (2012)
3. Barak, B., Lindell, Y., Vadhan, S.P.: Lower bounds for non-black-box zero knowledge. J. Comput. Syst. Sci. **72**(2), 321–391 (2006)
4. Bellare, M., Rogaway, P.: Random oracles are practical: A paradigm for designing efficient protocols. In: ACM Conference on Computer and Communications Security, pp. 62–73 (1993)
5. Bitansky, N., Canetti, R., Chiesa, A., Tromer, E.: From extractable collision resistance to succinct non-interactive arguments of knowledge, and back again. In: ITCS, pp. 326–349 (2012)
6. Bitansky, N., Canetti, R., Chiesa, A., Tromer, E.: Recursive composition and bootstrapping for snarks and proof-carrying data. In: STOC, pp. 111–120 (2013)
7. Bitansky, Nir, Canetti, Ran, Paneth, Omer, Rosen, Alon: On the existence of extractable one-way functions. In: STOC 2014, pp. 505–514 (2014)
8. Boneh, D., Waters, B.: Constrained pseudorandom functions and their applications. In: Sako, K., Sarkar, P. (eds.) ASIACRYPT 2013, Part II. LNCS, vol. 8270, pp. 280–300. Springer, Heidelberg (2013)
9. Boneh, D., Zhandry, M.: Multiparty key exchange, efficient traitor tracing, and more from indistinguishability obfuscation. In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014, Part I. LNCS, vol. 8616, pp. 480–499. Springer, Heidelberg (2014)
10. Boyle, E., Chung, K.-M., Pass, R.: On extractability obfuscation. In: Lindell, Y. (ed.) TCC 2014. LNCS, vol. 8349, pp. 52–73. Springer, Heidelberg (2014)
11. Boyle, E., Goldwasser, S., Ivan, I.: Functional signatures and pseudorandom functions. In: Krawczyk, H. (ed.) PKC 2014. LNCS, vol. 8383, pp. 501–519. Springer, Heidelberg (2014)

12. Boyle, E., Pass, R.: Limits of extractability assumptions with distributional auxiliary input. Cryptology ePrint Archive, Report 2013/703 (2013)
13. Brakerski, Z., Vaikuntanathan, V.: Lattice-based FHE as secure as PKE. In: Innovations in Theoretical Computer Science, ITCS 2014, pp. 1–12 (2014)
14. Canetti, R., Dakdouk, R.R.: Towards a theory of extractable functions. In: Reingold, O. (ed.) TCC 2009. LNCS, vol. 5444, pp. 595–613. Springer, Heidelberg (2009)
15. Damgård, I.B.: Towards practical public key systems secure against chosen ciphertext attacks. In: Feigenbaum, J. (ed.) CRYPTO 1991. LNCS, vol. 576, pp. 445–456. Springer, Heidelberg (1992)
16. Damgård, I., Faust, S., Hazay, C.: Secure two-party computation with low communication. In: Cramer, R. (ed.) TCC 2012. LNCS, vol. 7194, pp. 54–74. Springer, Heidelberg (2012)
17. Garg, S., Gentry, C., Halevi, S., Raykova, M.: Two-round secure MPC from indistinguishability obfuscation. In: Lindell, Y. (ed.) TCC 2014. LNCS, vol. 8349, pp. 74–94. Springer, Heidelberg (2014)
18. Garg, S., Gentry, C., Halevi, S., Raykova, M., Sahai, A., Waters, B.: Candidate indistinguishability obfuscation and functional encryption for all circuits. In: FOCS (2013)
19. Garg, S., Gentry, C., Halevi, S., Raykova, M., Sahai, A., Wichs, D.: On the implausibility of differing-inputs obfuscation and extractable witness encryption with auxiliary input. Cryptology ePrint Archive, Report 2013/860 (2013)
20. Garg, S., Gentry, C., Halevi, S., Wichs, D.: On the implausibility of differing-inputs obfuscation and extractable witness encryption with auxiliary input. In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014, Part I. LNCS, vol. 8616, pp. 518–535. Springer, Heidelberg (2014)
21. Gentry, C., Sahai, A., Waters, B.: Homomorphic encryption from learning with errors: conceptually-simpler, asymptotically-faster, attribute-based. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part I. LNCS, vol. 8042, pp. 75–92. Springer, Heidelberg (2013)
22. Goldreich, O., Goldwasser, S., Micali, S.: How to construct random functions. J. ACM **33**(4), 792–807 (1986)
23. Goldwasser, S., Kalai, Y.T., Popa, R.A., Vaikuntanathan, V., Zeldovich, N.: How to run turing machines on encrypted data. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part II. LNCS, vol. 8043, pp. 536–553. Springer, Heidelberg (2013)
24. Goldwasser, S., Lin, H., Rubinstein, A.: Delegation of computation without rejection problem from designated verifier cs-proofs. IACR Cryptology ePrint Archive 2011, 456 (2011)
25. Goldwasser, S., Micali, S., Rackoff, C.: The knowledge complexity of interactive proof systems. SIAM J. Comput. **18**(1), 186–208 (1989)
26. Gupta, D., Sahai, A.: On constant-round concurrent zero-knowledge from a knowledge assumption. Progress in Cryptology - INDOCRYPT 2014, pp. 71–88 (2014)
27. Hada, S., Tanaka, T.: On the existence of 3-round zero-knowledge protocols. In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, pp. 408–423. Springer, Heidelberg (1998)
28. Hohenberger, S., Sahai, A., Waters, B.: Replacing a random oracle: full domain hash from indistinguishability obfuscation. In: Nguyen, P.Q., Oswald, E. (eds.) EUROCRYPT 2014. LNCS, vol. 8441, pp. 201–220. Springer, Heidelberg (2014)
29. Hsiao, C.-Y., Reyzin, L.: Finding collisions on a public road, or do secure hash functions need secret coins? In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 92–105. Springer, Heidelberg (2004)

30. Ishai, Y., Pandey, O., Sahai, A.: Public-coin differing-inputs obfuscation and its applications. In: Dodis, Y., Nielsen, J.B. (eds.) TCC 2015, Part II. LNCS, vol. 9015, pp. 668–697. Springer, Heidelberg (2015)
31. Kiayias, A., Papadopoulos, S., Triandopoulos, N., Zacharias, T.: Delegatable pseudorandom functions and applications. In: CCS2013, pp. 669–684 (2013)
32. Micali, S.: CS proofs (extended abstracts). In: FOCS, pp. 436–453 (1994)
33. Naor, M.: On cryptographic assumptions and challenges. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 96–109. Springer, Heidelberg (2003)
34. Sahai, Amit, Waters, Brent: How to use indistinguishability obfuscation: deniable encryption, and more. In: STOC 2014, pp. 475–484 (2014)
35. Valiant, P.: Incrementally verifiable computation or proofs of knowledge imply time/space efficiency. In: Canetti, R. (ed.) TCC 2008. LNCS, vol. 4948, pp. 1–18. Springer, Heidelberg (2008)