Obfuscation of Probabilistic Circuits and Applications

Ran Canetti[1], Huijia Lin[2], Stefano Tessaro[2], and Vinod Vaikuntanathan[3]

[1] Boston University and Tel Aviv University
[2] UC Santa Barbara
[3] MIT CSAIL

Abstract. This paper studies the question of how to define, construct, and use obfuscators for *probabilistic programs*. Such obfuscators compile a possibly randomized program into a *deterministic* one, which achieves computationally indistinguishable behavior from the original program as long as it is run on each input at most once. For obfuscation, we propose a notion that extends *indistinguishability obfuscation* to probabilistic circuits: It should be hard to distinguish between the obfuscations of any two circuits whose output distributions at each input are computationally indistinguishable, possibly in presence of some auxiliary input. We call the resulting notion *probabilistic indistinguishability obfuscation (*pIO*).*

We define several variants of pIO, and study relations among them. Moreover, we give a construction of one of our variants, called X-pIO, from sub-exponentially hard indistinguishability obfuscation (for deterministic circuits) and one-way functions.

We then move on to show a number of applications of pIO. In particular, we first give a general and natural methodology to achieve fully homomorphic encryption (FHE) from variants of pIO and of semantically secure encryption schemes. In particular, one instantiation leads to FHE from any X-pIO obfuscator and any re-randomizable encryption scheme that's slightly super-polynomially secure.

We note that this constitutes the first construction of full-fledged FHE that does not rely on encryption with circular security.

Moreover, assuming sub-exponentially secure puncturable PRFs computable in $\mathbf{NC}^1$, sub-exponentially-secure indistinguishability obfuscation for (deterministic) $\mathbf{NC}^1$ circuits can be bootstrapped to obtain indistinguishability obfuscation for arbitrary (deterministic) poly-size circuits (previously such bootstrapping was known only assuming FHE with $\mathbf{NC}^1$ decryption algorithm).

1 Introduction

Program obfuscation, namely the algorithmic task of turning input programs into "unintelligible" ones while preserving their functionality, has been a focal point for cryptography for over a decade. However, while the concept is intuitively attractive and useful, the actual applicability of obfuscation has been limited. Indeed, the main notion to be considered has been virtual black box (VBB) [7] which, while natural and intuitively appealing, is very strong, hard to satisfy, and also not easy to use. In fact, for many program classes of interest, VBB obfuscation is unattainable [7,26,10].

All this changed with the recent breakthrough results of [21,37]. Their contribution is twofold: First they demonstrate a candidate general obfuscation algorithm for all circuits, thus reviving the hope in the possibility of making good of the initial intuitive appeal of program obfuscation as an important and useful cryptographic primitive. Second, they demonstrate how to make use of a considerably weaker notion of secure obfuscation than VBB, namely indistinguishability obfuscation (IO), initially defined in [7]. Indeed, following [21,37] there has been a gush of works demonstrating how to apply IO to a plethora of situations and applications, and even resolving long standing open problems.

Obfuscating probabilistic programs. Still, exiting notions of obfuscation, VBB and IO included, predominantly address the task of obfuscating *deterministic* programs. That is, the program to be obfuscated is a sequence of deterministic operations. This leaves open the question of obfuscating *probabilistic programs,* namely programs that make random choices as part of their specification, and whose output, on each input, is a random variable that depends on the internal random choices.

A priori it may not be clear what one wants to obtain when obfuscating such programs, or why is the problem different than that of obfuscating deterministic programs. Indeed, why not just obfuscate the deterministic program that takes both "main" and "random" input, and leave it to the evaluator to choose some of the input at random, if she so desires?

The main drawback of this approach is that it does not allow the obfuscation mechanism to hide the random choices of the program from the evaluator. Consider for instance the task of creating a program that allows generating elements of the form g^r, h^r for a random r, where g, h are two generators of a large group, and where r should remain hidden even from the evaluator of the program. Alternatively, consider the task of obfuscation-based re-encryption: Here we wish to "obfuscate" the program that decrypts a ciphertext using an internal decryption key, and then re-encrypts the obtained plaintext under a different key, using fresh randomness — all this while keeping the plaintext hidden from the evaluator.

Indeed, in both these examples, the goal is to create an obfuscation mechanism with two additional properties, stated very informally as follows: (a) the internal random choices of the obfuscated program should "remain hidden" from the evaluator, up to what is learnable from the output, and (b) the random choices of the program should remain "random", or "unskewed", as much as possible.

How can we define these properties in a sensible way? Barak et al. [7] take a first stab by defining the concept of *obfuscators for sampling algorithms,* namely algorithms that take only random input and at each execution output a sample from a distribution. Essentially, their definition requires that the (one bit) output of any adversary that has access to an obfuscated version of such a sampling algorithm be simulatable given only poly-many random samples from the distribution. However, while this definition does capture much of the essence of the problem, it is subject to essentially the same unattainability results that apply to VBB obfuscation.

Probabilistic IO. We propose an alternative definition for what it means to obfuscate probabilistic circuits. Our starting point is IO, rather than VBB, and hence we refer to the resulting general notion as *indistinguishability obfuscation for probabilistic circuits*, or pIO for short. This both reduces the susceptibility to unattainability results and allows making stronger distributional requirements on the outputs.

Consider a randomized circuit, namely a circuit C that takes an input x and a uniformly chosen random input r, and returns the random variable $C(x, r)$. The basic idea is to compile such a circuit into a *deterministic* obfuscated circuit $\Lambda = \mathcal{O}(C)$ that has essentially the same output distribution as the original circuit — with the one caveat that if Λ is run multiple times on the same input then it will give the same output.

The security requirements from a pIO obfuscator $\mathcal{O}$ for a family of circuits $\mathcal{C}$ are thus three: First, polynomial slowdown should hold as usual. Second, functionality should be preserved in the sense that for any $C \in \mathcal{C}$ and for any input x it should hold that $C(x) \approx_c \Lambda(x)$. Note that in $C(x)$ the probability is taken over the random choices of C (i.e., the sampling of r), whereas Λ is a deterministic circuit and the probability is taken *only over the random choices of* $\mathcal{O}$. In fact, we make the stronger requirement that no efficient adversary can distinguish whether it is given oracle access to the randomized oracle $C(\cdot)$ or the deterministic oracle $\Lambda(\cdot)$, as long as it does not submit repetitive queries to the oracles.)

Third, obfuscation should hold in the sense that $\mathcal{O}(C_1) \approx_c \mathcal{O}(C_2)$ for any two circuits C_1 and C_2 where the output distributions of $C_1(x)$ and $C_2(x)$ are "similar" for all inputs x, where similar means in general computationally indistinguishable. This property is trickiest to define, and to stress this even further, we note that the indistinguishability of $\mathcal{O}(C_1)$ and $\mathcal{O}(C_2)$ does *not* follow from IO even if the distributions of $C_1(x)$ and $C_2(x)$ are *identical.* Another important aspect is that we often need to consider programs that are parameterized by some additional system parameters, such as a public key of a cryptosystem. We thus extend the definition to consider also *families* of circuits with auxiliary input.

Concretely, we consider four variants of the above intuitive notion, depending on the specific notion of indistinguishability of probabilistic circuits assumed on the distribution. The four variants we consider differ in the level of adaptivity in choosing the inputs on which the programs are run in the experiment that determines whether programs are indistinguishable. Our formalization follows the approach of [9,2], capturing the strength of an IO algorithm $\mathcal{O}$ in terms of the distributions on triples (C_1, C_2, z) on which it succeeds in making $\mathcal{O}(C_1)$ and $\mathcal{O}(C_2)$ indistinguishable (given z).

A construction for X-pIO. As our first main result, we show how to construct a general X-pIO scheme, where X-pIO is one of our four variants (see definition within), from any subexponentially secure IO scheme and one way function. The scheme is natural: X-$pi\mathcal{O}(C)$ is the result of applying an indistinguishability obfuscator to the following circuit. First apply the puncturable PRF to the input x to obtain a pseudorandom value r, using a hard-coded PRF secret key. Next, we run the circuit C on input x and random input r.

We show by reduction that if the underlying IO and puncturable PRF are sub-exponentially secure then the scheme is X-pIO. Furthermore, one can consider the same natural construction as a candidate implementation of any of the other variants of pIO.

Applications: FHE and Bootstrapping. To demonstrate the usefulness of pIO we present two natural applications of the notion, which are arguably of independent interest.

Our first application (see Section 2.2) is to constructing fully homomorphic encryption schemes. Here we provide a natural construction of fully homomorphic encryption from pIO (or, in turn from sub-exponentially secure IO and puncturable PRFs.) In fact, we provide the *first* full-fledged FHE scheme that does not rely on circular security assumptions for encryption.

We proceed in two steps. First we show how to obtain leveled homomorphic encryption (LHE), where only a prespecified number of homomorphic operations can be made securely. The basic idea is to use pIO to transform an underlying encryption scheme with some mild structural properties (such as rerandomizability) into an LHE. We give a number of different instantiations of the general scheme, where each instantiation uses a different variant of pIO and a different type of encryption scheme as a starting point.

The second step transforms the resulting LHE into a full-fledged FHE, again assuming IO and puncturable PRFs. (All primitives from LHE to IO to PRFs are required to be slightly super-polynomially secure.) While this transformation works in general for any LHE with a-priori fixed polynomial decryption depth, it is particularly suitable for LHEs that result from our pIO based construction in that it uses the same underlying primitives and assumptions. These constructions use in an inherent way the concept of obfuscation of randomized circuits, and in particular probabilistic IO.

As a second application, discussed in Section 2.3, we consider variants of bootstrapping, transforming IO obfuscation (both probabilistic and not probabilistic) for weak classes (such as low-depth circuits) into obfuscation for arbitrary polynomial-size circuits.

Organization. Section 2 gives a detailed high-level and self-contained overview of the contributions of this paper, both at the definitional level, as well as in terms of applications.

Further down, Section 3 presents our definitions of pIO and studies relations among them. Moreover, it presents the construction of X-pIO from IO and puncturable PRFs. Section 4 present the application to FHE, whereas the application to bootstrapping IO is deferred to the full version for lack of space.

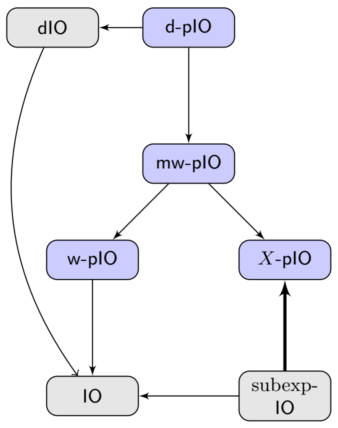

Fig. 1. Notions of obfuscation for probabilistic circuits: Arrows indicate implication, whereas lack of arrows among azure boxes implies a formal separation. The thicker line indicates that the implication holds under the assumption of subexponentially-hard puncturable PRFs.

2 Overview

We provide an overview of the definitions and results in this work.

2.1 Our Definitional Framework: IO for Probabilistic Circuits

The first contribution of this paper, found in Section 3, is the definition and study of IO notions for *probabilistic* circuits, or pIO. For our purposes, a probabilistic obfuscator $pi\mathcal{O}$ transforms a (usually probabilistic, i.e. randomized) circuit C into a *deterministic* circuit $\Lambda := pi\mathcal{O}(C)$ with the property that $\Lambda(x)$ is computationally indistinguishable from $C(x)$ the *first* time it is invoked, even when the circuits are invoked as oracles multiple times on *distinct* inputs. (Across multiple calls with the *same* input, $\Lambda(x)$ returns the same value over and over, whereas $C(x)$ returns a fresh random output.)

As for security, we want to ensure indistinguishability of $pi\mathcal{O}(C_0)$ and $pi\mathcal{O}(C_1)$ whenever $C_0(x)$ and $C_1(x)$ are computationally indistinguishable for every input x, rather than identical as in IO. However, formalizing this concept is challenging, due to the exponential number of inputs and the fact that C_0, C_1 are usually chosen from some distribution.

Four pIO notions. Following the approach of [9,2], we capture different pIO notions via classes of *samplers*, where such a sampler is a distributions D (parametrized by the security parameter) outputting triples (C_0, C_1, z), such that C_0, C_1 are circuits, and z is some (generally correlated) auxiliary input. Different pIO notions result from different requirement in terms of the class of samplers for which a certain obfuscator $pi\mathcal{O}$ guarantees indistinguishability of the obfuscations of C_0 and C_1 (given the auxiliary input z), in addition to the above correctness requirement. Concretely, we consider four different notions matching different approaches to formalizing the above computational indistinguishability requirement on all inputs:

X-Ind pIO (X-pIO). In the simplest notion, we require that for every *statically* chosen input x, the distributions of $C_0(x)$ and $C_1(x)$ are indistinguishable, given z, where the randomness is over the sampled (C_0, C_1, z). While this results in an unachievable notion, we additionally require the distinguishing advantage to be very small, smaller than $\mathsf{negl} \cdot X^{-1}$, for some negligible function, where X is the number of inputs of the circuits. The requirement on the small distinguishing gap seems stringent and leads to a weak notion, but we show that it is necessary.

Dynamic-input pIO (d-pIO). A d-pIO obfuscator is required to work on samplers D such that any PPT attacker, given a triple (C_0, C_1, z) sampled from D, cannot find (adaptively) an input x for which, when given additionally $C_b(x)$ for a random b, it can guess the value of b with noticeable advantage over random guessing.

Worst-case-input pIO (w-pIO and mw-pIO). A w-pIO obfuscator weakens the above notion by only working on samplers for which the above indistinguishability requirement holds for (much) stronger attackers where the choice of x after sampling (C_0, C_1, z) is made without any computational restrictions, whereas the final guess, *after* learning $C_b(x)$, is restricted to be polynomial-time. This captures the fact that the choice of the input x is *worst case* as to maximize the guessing probability in the second stage. The (stronger) notion where we enlarge our sampler class to only require indistinguishability for attackers not passing such state is referred to as *memory-less worst-case-input pIO* (or mw-pIO for short).

We prove that d-pIO implies mw-pIO, and mw-pIO implies both w-pIO and X-pIO, but the latter two notions do not imply each other. These relations are summarized in Figure 1 below. The fact that mw-pIO implies X-pIO is surprising at first, as on one hand we are *restricting* the power of the attacker, but on the other hand we are simplifying its task by choosing our barrier at negl/X advantage, and it is not clear what prevails.

The notion of d-pIO is a natural generalization of the notion of *differing inputs obfuscation* [7,13,2], and therefore directly suffers from recent implausibility results [22] in its most general form. In contrast, achievability of mw-pIO and the even weaker notion of w-pIO is not put in question by similar results, and the original IO notion is recovered from both w-pIO and mw-pIO when restricting them to deterministic circuits only. We in fact feel comfortable in conjecturing that w-pIO is achieved by a construction first transforming a randomized circuit C into a deterministic one $D^k(x) = C(x; \mathsf{PRF}(k, x))$ for a PRF key k, then applying an existing obfuscator $\mathcal{O}$ to D^k, such as those from [21,6,17].

X-Ind pIO from sub-exponential IO. The main technical result of this part is a proof that for X-pIO, the above approach indeed *provably* yields a secure obfuscator if the PRF is puncturable and if the obfuscator $\mathcal{O} = i\mathcal{O}$ is an IO, as long as additionally PRF and $i\mathcal{O}$ are *subexponentially secure*. In this context, sub-exponential means that no PPT attacker can achieve better than sub-exponential advantage, an assumption which we believe to be reasonable.

2.2 Application 1: Fully-Homomorphic Encryption

The first testbed for our pIO notions, discussed in Section 4, is a generic construction of leveled homomorphic-encryption (or LHE, for short) from a regular encryption scheme. We are then going to boost this to achieve fully-homomorphic encryption (FHE) without any circular security assumptions using a technique of independent interest.

The LHE construction. When trying to build a LHE scheme using ofuscation, the following natural and straightforward idea came up immediately. Starting from a CPA-secure encryption, we generate public-key and secret-key pairs for all levels $(\mathsf{pk}_0, \mathsf{sk}_0), \ldots, (\mathsf{pk}_L, \mathsf{sk}_L)$, and then, as part of the evaluation key, add for every level $i \in \{1, \ldots, L\}$, the pIO obfuscation of the circuit $\mathsf{Prog}^{(\mathsf{sk}_{i-1}, \mathsf{pk}_i)}$ which takes two ciphertexts $\alpha = \mathsf{Enc}(\mathsf{pk}_{i-1}, a)$ and $\beta = \mathsf{Enc}(\mathsf{pk}_{i-1}, b)$ (where a and b are bits), decrypts them using sk_{i-1}, and then outputs a fresh encryption $c = \mathsf{Enc}(\mathsf{pk}_i, a \text{ NAND } b)$. The outputs of this circuit, given sk_{i-1} and pk_i (but not sk_i) are computationally indistinguishable from those of a "trapdoor" circuit $\mathsf{tProg}^{(\mathsf{pk}_i)}$ which instead ignores its inputs, and simply outputs a fresh encryption $c = \mathsf{Enc}(\mathsf{pk}_i, 0)$ of 0. Note that this circuit is independent of sk_{i-1}. We therefore hope that by relying on some pIO notion for the sampler $D^{\mathsf{sk}_{i-1}}$ that outputs $(\mathsf{Prog}^{(\mathsf{sk}_{i-1}, \mathsf{pk}_i)}, \mathsf{tProg}^{(\mathsf{pk}_i)}, \mathsf{pk}_i)$ (and through a careful hybrid argument), one might transform the honest evaluation key to one that contains only obfuscations of the "trapdoor" circuits; in the latter case, since the evaluation key depends only on *public* keys, the semantic security of the LHE scheme reduces down to that of the underlying CPA scheme. The nice feature of this approach is that it builds on top of any already existing encryption scheme (say ElGamal), and that for all levels, ciphertexts are of the same type and size. A similar generic approach was for example abstracted in the work of Alwen et al. [1], and proved secure under ad-hoc obfuscation assumptions.

Unfortunately, it turns out that the above approach generically works for every CPA-secure scheme only when using d-pIO, which, as we discussed above, is somewhat brittle. Indeed, the above sampler $D^{\mathsf{sk}_{i-1}}$ is *not* contained in the classes associated with X-pIO and w-pIO. With respect to X-pIO there is no guarantee that encryptions (of values (a NAND b) or 0) are negl/X close to each other (note that here the domain size X corresponds to the length $|\alpha| + |\beta|$ of the *two* input ciphertexts). It seems that to fix the problem, one could simply re-encrypt under an encryption scheme which is negl/X secure (which exists assuming sub-exponentially secure CPA encryption), but this results in a longer output ciphertext of size $\text{poly}(\log X)$ (i.e., $\text{poly}(|\alpha| + |\beta|)$), leading to exponentially growing ciphertext with the depth.

With respect to w-pIO (and to mw-pIO also), the main challenge with the above sampler is that given the two circuits, the adversarial first stage is computationally unbounded and can (for example) find a secret key corresponding to the public key, and pass it on to the second stage, which proceeds in distinguishing encryptions (of values (a NAND b) and 0 again) using the secret key efficiently.

LHE via trapdoor encryption. We get around the above conundrum by using a generalization of CPA encryption—called trapdoor encryption: The idea here is that the encryption scheme can generate a *special* trapdoor key which is indistinguishable from a real public-key, but it does not guarantee decryption any more. In this way, we expect to be able to guarantee stronger ciphertext indistinguishability (even statistical) under a trapdoor key which cannot be satisfied by normal encryption scheme as long as correctness needs to be guaranteed. In particular, we modify the proof in the above approach as follows: In the hybrids, the obfuscations in the evaluation key are changed one by one in the reverse order; to change the obfuscation of circuit $\mathsf{Prog}^{(\mathsf{sk}_{i-1},\mathsf{pk}_i)}$, first replace the public key with a trapdoor key tpk_i, and then move to an obfuscation of a modified trapdoor circuit $\mathsf{tProg}^{(\mathsf{tpk}_i)}$ with the trapdoor key built in. Now thanks to the stronger ciphertext indistinguishability under the trapdoor key, it suffices to use weak notions of pIO. In this paper, we provide the following instantiations of this paradigm:

– **Lossy encryption + $\mathsf{w\text{-}pIO}$.** In order to instantiate the construction from $\mathsf{w\text{-}pIO}$, we consider encryption schemes which are statistically secure under a trapdoor key, so-called *lossy* encryption schemes [8]. Such schemes can be built using techniques from a variety of works [31,34,8,35], and admit instantiations from most cryptographic assumptions. This gives an LHE construction from $\mathsf{w\text{-}pIO}$ and any lossy encryption schemes.[4]
– **Re-randomizable encryption + sub-exponential IO.** Existing constructions of lossy encryption unfortunately do not allow a distinguishing gap of negl/X without having the ciphertext size growing polynomially in $\log X$. Instead, we construct a trapdoor encryption scheme with such a tiny distinguishing gap under the trapdoor key, from any re-randomizable (secret or public-key) encryption scheme: The (honest) public key of the trapdoor encryption scheme consists of two encryptions (c_0, c_1) of 0 and 1 of the underlying re-randomizable encryption scheme, and to encrypt a bit b, one simply re-randomizes c_b; the trapdoor key, on the other hand, simply consists of two encryptions (c_0, c_0') of both 0. By the semantic security of the underlying scheme, the honest and trapdoor keys are indistinguishable. Furthermore, if the re-randomizability of the underlying scheme guarantees that re-randomization of one ciphertext or another of the same plaintext yields identical distributions, then encryptions under the trapdoor keys are perfectly hiding. Many encryption schemes such as ElGamal, Goldwasser-Micali [27], Paillier [33], Damgård-Jurik [20], satisfy

[4] In fact, this instantiation only requires an even weaker $\mathsf{w\text{-}pIO}$ notion where sampler indistinguishability must hold against computationally unbounded adversaries in both stages.

the perfect re-randomizability. Therefore, when relying on such a scheme, the corresponding samplers is negl/X-indistinguishable, for any X; hence X-pIO suffices. Combined with the aforementioned construction of X-pIO, this also gives us leveled LHE from any re-randomizable encryption scheme and sub-exponentially hard IO and one-way functions.

We also note that the instantiation from d-pIO mentioned above from any CPA-secure encryption scheme is also a (trivial) application of the above general result.

From LHE to FHE. As a final contribution of independent interest, we use IO to turn an LHE scheme info an FHE scheme via techniques inspired by the recent works of Bitansky, Garg, and Telang [11], and of Lin and Pass [32].

The basic idea is to instantiate the above LHE construction on *super-polynomially* many levels, but to represent these keys *succinctly*. This is done by considering a circuit Γ that on input i genarates the i-th level evaluation key, i.e., the pIO obfuscation of $\mathsf{Prog}^{(\mathsf{sk}_{i-1},\mathsf{pk}_i)}$ (in the evaluation key for super-polynomially many levels), where the key pairs $(\mathsf{pk}_{i-1},\mathsf{sk}_{i-1})$ and $(\mathsf{pk}_i,\mathsf{sk}_i)$ are generated using pseudo-random coins $\mathsf{PRF}(k, i-1)$ and $\mathsf{PRF}(k,i)$ computed using a puncturable PRF on a hard-coded seed k; (the pIO obfuscations also use pseudo-random coins as well). The new succinct evaluation key is the *IO-obfuscation of this circuit* Γ, while the public key is pk_0 (generated using coins $\mathsf{PRF}(k,0)$) and the secret key is the PRF seed k. In order for this approach to be secure, we need the IO obfuscation to be slightly super-polynomially secure (not necessarily sub-exponentially secure), in order to accommodate for a number of hybrids in the proof which accounts to the (virtual) super-polynomial number of levels implicitly embedded in the succinct representation. In particular, we get this step almost for free (in terms of assumptions) when starting with our LHE constructions, either because we assume sub-exponential IO in the first place, or assuming just a slightly stronger form of w-pIO and d-pIO than what necessary above.

We also observe that this is a special case of a more general paradigm of using IO to turn any LHE with a fixed decryption depth (independent of the maximum evaluation level) into an FHE, which applies to almost all known LHE schemes (e.g. [23,18,16,15,24]). We believe that this general transformation is of independent interest, especially because it does not rely on any encryption scheme with circular security.

2.3 Application 2: Bootstrapping IO

Our second contribution is to use the notion of pIO to provide a simple way of bootstrapping (standard, deterministic) IO for weak circuit classes, such as $\mathbf{NC}^1$, into ones for all polynomial-size circuits. In the very first candidate construction of IO for $\mathbf{P}/poly$, Garg et al. [21] show how to obtain full fledged IO assuming the existence of indistinguishability obfuscation for a weak circuit class **WEAK**, as well as a fully homomorphic encryption scheme whose decryption can be computed in **WEAK** (given the known FHE schemes, one can think of **WEAK** as $\mathbf{NC}^1$). The natural question that remained is: Can we achieve bootstrapping without the FHE assumption?

We show a new way to bootstrap indistinguishability obfuscation, without assuming that FHE schemes exist. Instead, our assumption is the existence of sub-exponentially hard indistinguishability obfuscation for a complexity class **WEAK** and a sub-exponentially secure puncturable PRF computable in **WEAK**. Our technique is inspired by the recent work of Applebaum [3] that shows how to bootstrap VBB obfuscations from **WEAK** to $\mathbf{P}/poly$ using randomized encodings; however his transformation strongly relies on the fact that the starting point is a VBB obfuscation.

The idea is to apply the "randomized encodings" paradigm which was originally proposed in the context of multiparty computation [29,4] and has found many further uses ever since. A randomized encoding RE for a circuit family $\mathcal{C}$ is a probabilistic algorithm that takes as input a circuit $C \in \mathcal{C}$ and an input x, and outputs its randomized encoding $(\hat{C}, \hat{x})$. The key properties of RE are that: (1) given $\hat{C}$ and $\hat{x}$, one can efficiently recover $C(x)$; (2) given $C(x)$, one can efficiently simulate the pair $(\hat{C}, \hat{x})$, implying that the randomized encoding reveals no information beyond the output $C(x)$; and (3) computing RE is very fast in parallel. In particular, the work of Applebaum, Ishai and Kushilevitz [5], building on Yao's garbled circuits, showed a way to perform randomized encoding of any circuit in $\mathbf{P}/poly$ using a circuit $\mathsf{RE} \in \mathbf{NC}^0$, assuming a PRG in $\oplus\mathbf{L}/poly$ (which is implied by most cryptographic assumptions). The typical use of randomized encodings is to reduce computing a circuit C to the easier task of computing its randomized encoding $\mathsf{RE}(C, \cdot)$.

Therefore, to obfuscate a circuit $C \in \mathbf{P}/poly$, the natural idea is obfuscating its randomized encoding $\mathsf{RE}(C, x; r)$ using an appropriate pIO scheme for $\mathbf{NC}^0$. (Here, pIO comes into play naturally, since RE is a randomized circuit.) We show that, in fact, X-pIO suffices for this purpose: Assuming that randomized encoding is sub-exponentially secure, then for any two functionally equivalent circuits C_1 and C_2, their randomized encoding $\mathsf{RE}(C_1, x; r)$ and $\mathsf{RE}(C_2, x; r)$ have indistinguishable outputs for every input x, where the distinguishing gap is as small as $\mathsf{negl}(\lambda)2^{-|x|}$[5]. Therefore, obfuscating $\mathsf{RE}(C_1, \cdot)$ and $\mathsf{RE}(C_2, \cdot)$ using an X-pIO scheme $pi\mathcal{O}$ yields indistinguishable obfuscated programs, and hence $i\mathcal{O}(C) = pi\mathcal{O}(\mathsf{RE}(C, \cdot))$ is an indistinguishable obfuscator for all $\mathbf{P}/poly$. Since, our construction of X-pIO from sub-exponentially indistinguishable IO preserves the class of circuits modulo the complexity of the sub-exponentially indistinguishable puncturable PRF. Put together, we are able to bootstrap sub-exponentially indistinguishable IO for a weak class, say $\mathbf{NC}^1$, to IO for all of $\mathbf{P}/poly$, assuming a sub-exponentially indistinguishable PRF computable in the weak class of circuits.

Bootstrapping pIO. The same technique above can be applied to bootstrap worst-case-input pIO from $\mathbf{NC}^0$ to $\mathbf{P}/poly$, assuming the existence of a PRG in $\oplus\mathbf{L}/poly$. The key observation here is that since pIO handles directly randomized

[5] This can be done by using a sufficiently large security parameter when generating the randomized encoding.

circuits, it can be used to obfuscate the randomized encoding $\mathsf{RE}(C, \cdot)$ (without relying on pseudorandom functions). Furthermore, the security of the randomized encoding holds for any input and auxiliary information (even ones that are not efficiently computable). Then, given any two circuits $C_1(x;r), C_2(x;r)$ whose outputs are indistinguishable even for dynamically chosen worst-case inputs, their randomized encoding $C_1'(x;r,r') = \mathsf{RE}(C_1,(x,r);r')$ and $C_2'(x;r,r') = \mathsf{RE}(C_2,(x,r);r')$ are also indistinguishable on dynamically chosen worst case inputs. This is because, over the random choice of r and r', the distributions of $C_1'(x;r,r')$ and $C_2'(x;r,r')$ can be simulated using only $C_1(x;r)$ and $C_2(x;r)$, which are indistinguishable. Therefore a worst-case-input pIO scheme for $\mathbf{NC}^0$ suffices for obfuscating the circuit $C'(x;r,r') = \mathsf{RE}(C,(x,r);r)$, leading to a worst-case-input pIO scheme for all $\mathbf{P}/poly$. Following the same approach, we can bootstrap dynamic-input pIO for $\mathbf{NC}^0$ to dynamic-input pIO for $\mathbf{P}/poly$ assuming a PRG in $\oplus\mathbf{L}/poly$. Similarly, we can also bootstrap X-pIO for $\mathbf{NC}^0$ to X-pIO for $\mathbf{P}/poly$, but relying on the sub-exponential security of the PRG. The stronger security of PRG is needed so that the randomized encoding can be made $\mathsf{negl}(\lambda)/X(\lambda)$ indistinguishable.

3 IO for Probabilistic Circuits

3.1 IO for General Samplers over Probabilistic Circuits

We start with the notion of indistinguishability obfuscation for general classes of samplers over *potentially probabilistic* circuits, called pIO for samplers in class $\mathbf{S}$. Here, a sampler is a distribution ensemble over pairs of potentially randomized circuits, together with an auxiliary input. Below, we define various notions of obfuscation for probabilistic circuits instantiating the general definition with classes of samplers that produce pairs of probabilistic circuits satisfying different variants of our point-wise indistinguishability requirement.

More formally, let $\mathcal{C} = \{\mathcal{C}_\lambda\}_{\lambda\in\mathbb{N}}$ be a family of sets of (randomized) circuits, where $\mathcal{C}_\lambda$ contains circuits of size $\text{poly}(\lambda)$. Extending the notation of [9], a *circuit sampler* for $\mathcal{C}$ is a distribution ensemble $D = \{D_\lambda\}_{\lambda\in\mathbb{N}}$, where the distribution D_λ ranges over triples (C_0, C_1, z) with $C_0, C_1 \in \mathcal{C}_\lambda$ such that C_0, C_1 take inputs of the same length, and $z \in \{0,1\}^{\text{poly}(\lambda)}$. Moreover, a *class* $\mathbf{S}$ of samplers for $\mathcal{C}$ is a set of circuit samplers for $\mathcal{C}$.

The following definition captures the notion of pIO for a class of samplers.

Definition 1 (pIO for a Class of Samplers). *A uniform PPT machine $pi\mathcal{O}$ is an* indistinguishability obfuscator *for a class of samplers* $\mathbf{S}$ *over the (potentially randomized) circuit family $\mathcal{C} = \{\mathcal{C}_\lambda\}_{\lambda\in\mathbb{N}}$ if the following two conditions hold:*

Correctness: *$pi\mathcal{O}$ on input a (potentially probabilistic) circuit $C \in \mathcal{C}_\lambda$ and the security parameter $\lambda \in \mathbb{N}$ (in unary), outputs a deterministic circuit Λ of size* $\text{poly}(|C|, \lambda)$.

Furthermore, for every non-uniform PPT distinguisher $\mathcal{D}$, every (potentially probabilistic) circuit $C \in \mathcal{C}_\lambda$, and string z, we define the following two experiments:

- $\mathsf{Exp}^1_{\mathcal{D}}(1^\lambda, C, z)$: *$\mathcal{D}$ on input $1^\lambda, C, z$, participates in an unbounded number of iterations of his choice. In iteration i, it chooses an input x_i; if x_i is the same as any of the previously chosen input x_j for $j < i$, then abort; otherwise, $\mathcal{D}$ receives $C(x_i; r_i)$ using fresh random coins r_i ($r_i = null$ if C is deterministic). At the end of all iterations, $\mathcal{D}$ outputs a bit b. (Note that $\mathcal{D}$ can keep state across iterations.)*
- $\mathsf{Exp}^2_{\mathcal{D}}(1^\lambda, C, z)$: *Obfuscate circuit C to obtain $\Lambda \xleftarrow{\$} pi\mathcal{O}(1^\lambda, C; r)$ using fresh random coins r. Run $\mathcal{D}$ as described above, except that in each iteration, feed $\mathcal{D}$ with $\Lambda(x_i)$ instead.*

Overload the notation $\mathsf{Exp}^i_{\mathcal{D}}(1^\lambda, C, z)$ as the output of $\mathcal{D}$ in experiment $\mathsf{Exp}^i_{\mathcal{D}}$. We require that for every non-uniform PPT distinguisher $\mathcal{D}$, there is a negligible function μ, such that, for every $\lambda \in \mathbb{N}$, every $C \in \mathcal{C}_\lambda$, and every auxiliary input $z \in \{0,1\}^{\mathrm{poly}(\lambda)}$,

$$\mathsf{Adv}_{\mathcal{D}}(1^\lambda, C, z) = |\Pr[\mathsf{Exp}^1_{\mathcal{D}}(1^\lambda, C, z)] - \Pr[\mathsf{Exp}^2_{\mathcal{D}}(1^\lambda, C, z)]| = \mu(\lambda)\ .$$

Security with respect to S: *For every sampler $D = \{D_\lambda\}_{\lambda\in\mathbb{N}} \in \mathbf{S}$, and for every non-uniform PPT machine $\mathcal{A}$, there exists a negligible function μ such that*

$$\Big|\Pr[(C_1, C_2, z) \xleftarrow{\$} D_\lambda \ : \ \mathcal{A}(C_1, C_2, pi\mathcal{O}(1^\lambda, C_1), z) = 1] - \\ - \Pr[(C_1, C_2, z) \xleftarrow{\$} D_\lambda \ : \ \mathcal{A}(C_1, C_2, pi\mathcal{O}(1^\lambda, C_2), z) = 1]\Big| = \mu(\lambda)\ .$$

where μ is called the **distinguishing gap***.*

Furthermore, we say that $pi\mathcal{O}$ is **δ-indistinguishable** *if the distinguishing gap μ bounded by δ. Especially, $pi\mathcal{O}$ is* **sub-exponentially indistinguishable** *if $\mu(\lambda)$ is bounded by $2^{-\lambda^\epsilon}$ for a constant ϵ.*

Note that the sub-exponential indistinguishability defined above is weaker than usual sub-exponential hardness assumptions in that the distinguishing gap only needs to be small for PPT distinguishers, rather than sub-exponential ones.

An obvious (but important) remark is that an obfuscator $pi\mathcal{O}$ for the class $\mathbf{S}$ is also an obfuscator for any class $\mathbf{S}' \subseteq \mathbf{S}$, whereas conversely, if no obfuscator exists for $\mathbf{S}'$ (or its existence is implausible), then the same is true for $\mathbf{S} \supseteq \mathbf{S}'$.

3.2 Static-input pIO for Circuits

Arguably, the simplest way to formulate the property that two circuits are indistinguishable on every input is to require that this true for every statically chosen input, i.e., chosen independently of the random choice of the sampler. This results in the following definition, which we state for completeness, but that we will have to further restrict below to by-pass impossibility:

Definition 2 (*Static-input* Indistinguishable Samplers). *The class $\mathbf{S}^{\mathsf{s\text{-}Ind}}$ of* static-input indistinguishable samplers *for a circuit family $\mathcal{C}$ contains all circuit samplers $D = \{D_\lambda\}_{\lambda\in\mathbb{N}}$ for $\mathcal{C}$ with the following property: For all non-uniform PPT $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, the advantage of $\mathcal{A}$ in the following experiment is negligible.*

Experiment *static-input*-$\mathsf{IND}^{D}_{\mathcal{A}}(1^\lambda)$*:*

1. $(x, st) \xleftarrow{\$} \mathcal{A}_1(1^\lambda)$ *// $\mathcal{A}_1$ chooses challenge input x statically.*
2. $(C_0, C_1, z) \xleftarrow{\$} D_\lambda$
3. $y \xleftarrow{\$} C_b(x)$*, where* $b \xleftarrow{\$} \{0,1\}$.
4. $b' \xleftarrow{\$} \mathcal{A}_2(st, C_0, C_1, z, x, y)$

The advantage of $\mathcal{A}$ is $\Pr[b' = b] - 1/2$.

Unfortunately, we now show that pIO for general static-input indistinguishable samplers is (unconditionally) impossible, but we will see below that a further restriction of the class $\mathbf{S}^{\mathsf{s\text{-}Ind}}$ will bypass this impossibility.

Proposition 1. *There exists a static-input indistinguishable sampler D^* over deterministic circuits, such that, there is no* pIO *for D^*.*

Proof. Consider the following sampler D^*: D^*_λ samples (C_0, C_1, z) where C_0 is an all zero circuit, C_1 computes a point function that outputs 1 at a single point s chosen uniformly randomly, and z is set to s. Clearly, D^* is a static-input indistinguishable sampler. Indeed, for any fixed input x, with overwhelming probability D^* samples (C_0, C_1, s), with a differing input $s \neq x$. Thus, the outputs $C_0(x) = C_1(x) = 0$ cannot be distinguished.

However, any $pi\mathcal{O}$ achieving correctness cannot be secure for this sampler D^*: An adversary can easily tell apart $(C_0, C_1, s, \Lambda_0 = pi\mathcal{O}(C_0))$ from $(C_0, C_1, s, \Lambda_1 = pi\mathcal{O}(C_1))$ by simply evaluating Λ_0 and Λ_1 on input s. □

X-Ind pIO. To circumvent impossibility, we consider a smaller class of static-input indistinguishable samplers, $\mathbf{S}^{X\text{-}\mathsf{Ind}} \subset \mathbf{S}^{\mathsf{s\text{-}Ind}}$. In fact, in Section 3.6 we give a construction for such a pIO assuming sub-exponentially indistinguishable IO.

The samplers D we consider satisfy that the distinguishing gap of any PPT adversary in the above static-input-IND experiment is bounded by $\mathsf{negl} \cdot X^{-1}$, where X is the number of "differing inputs" that circuits C_0, C_1 sampled from D have, and negl is some negligible function. More precisely:

Definition 3 ((Static-input) X-Ind-Samplers). *Let $X(\lambda)$ be a function bounded by 2^λ. The class $\mathbf{S}^{X\text{-}\mathsf{Ind}}$ of* (static-input) X-Ind-samplers *for a circuit family $\mathcal{C}$ contains all circuit samplers $D = \{D_\lambda\}_{\lambda \in \mathbb{N}}$ for $\mathcal{C}$ with the following property: For every $\lambda \in \mathbb{N}$, there is a set $\mathcal{X} = \mathcal{X}_\lambda \subseteq \{0,1\}^*$ of size at most $X(\lambda)$ (called the differing domain), such that,*

X **differing inputs:** *With overwhelming probability over the choice of $(C_0, C_1, z) \xleftarrow{\$} D_\lambda$, for every input outside the differing domain, $x \notin \mathcal{X}$, it holds that $C_0(x'; r) = C_1(x'; r)$ for every random string r.*

X**-indistinguishability:** *For all non-uniform PPT $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, the advantage of $\mathcal{A}$ in the experiment static-input-$\mathsf{IND}^{D}_{\mathcal{A}}(1^\lambda)$ defined in Definition 2 is* $\mathsf{negl} X^{-1}$.

Definition 4 **(X-Ind pIO for Randomized Circuits).** *Let X be any function bounded by 2^λ. A uniform PPT machine X-piO is an X-pIO for randomized circuits, if it is a pIO for the class of X-Ind samplers $\mathbf{S}^{X\text{-}\mathsf{Ind}}$ over $\mathcal{C}$ that includes all randomized circuits of size at most λ.*

We note that the notion of a differing set is added for flexibility purposes, as our constructions below will allow for it. We stress that its definition is not allowed to depend on the circuits which are actually sampled, and must be fixed a-priori. Also, note that the notion encompasses the setting where $C_0(x)$ and $C_1(x)$ are identically distributed, or are statistically very close.

The notion of X-Ind pIO is the "best-possible" achievable with respect of static input. Indeed, one can modify the distribution D^* constructed in Proposition 1 to have $C_1(s)$ output 1 with probability $\frac{1}{p(\lambda)}$ for a polynomial p. The differing domain there is the whole domain, i.e., $\mathcal{X}_\lambda = \{0,1\}^\lambda$ (since the circuit *may* differ at any point.) This makes the sampler *exactly* $X \cdot p^{-1}$ indistinguishable for static adversaries, as $C_1(x) \neq C_0(x)$ with probability $X \cdot p^{-1}$ over the choice of (C_0, C_1, z). Yet, pIO for this sampler is impossible, as again, the circuits differ on input $z = s$ with probability $\frac{1}{p(\lambda)}$. This impossibility cannot be pushed any further, and indeed general X-pIO is possible, as shown in Section 3.6.

3.3 Dynamic-input pIO for Circuits

The above notion, while achievable, makes an unnaturally strong indistinguishability requirement. We explore alternative notions where the distinguishing gap is not required to be as small. We start with a natural sampler notion asking for indistinguishability on every input x *adaptively* chosen by a (PPT) adversary $\mathcal{A}_1$ on input (C_0, C_1, z).

Definition 5 **(*Dynamic-input* Indistinguishable Samplers).** *The class $\mathbf{S}^{\mathsf{d\text{-}Ind}}$ of* dynamic-input indistinguishable samplers *for a circuit family $\mathcal{C}$ contains all circuit samplers $D = \{D_\lambda\}_{\lambda\in\mathbb{N}}$ for $\mathcal{C}$ with the following property: For all non-uniform PPT $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, the advantage of $\mathcal{A}$ in the following experiment is negligible.*

Experiment *dynamic-input*-$\mathsf{IND}^D_{\mathcal{A}}(1^\lambda)$*:*

1. $(C_0, C_1, z) \overset{\$}{\leftarrow} D_\lambda$
2. $(x, st) \overset{\$}{\leftarrow} \mathcal{A}_1(C_0, C_1, z)$
3. $y \overset{\$}{\leftarrow} C_b(x)$*, where* $b \overset{\$}{\leftarrow} \{0,1\}$
4. $b' \overset{\$}{\leftarrow} \mathcal{A}_2(st, C_0, C_1, z, x, y)$

The advantage of $\mathcal{A}$ is $\Pr[b' = b] - 1/2$.

We note that the restriction to requiring indistinguishability on a single input is without loss of generality, as it follows from a standard hybrid argument that the advantage of any efficiency adversary is still negligible even if it receives samples from $C_b(x)$ for an unbounded number of *adaptively* chosen inputs.

We can now use the above sampler class to directly obtain the notion of Dynamic-input pIO for randomized circuits.

Definition 6 (Dynamic-input pIO for Randomized Circuits). *A uniform PPT machine* d-*pi$\mathcal{O}$ is a* dynamic-input pIO (or d-pIO) *for randomized circuits, if it is a* pIO *for the class of dynamic-input indistinguishable samplers* $\mathbf{S}^{\mathsf{d\text{-}Ind}}$ *over $\mathcal{C}$ that includes all randomized circuits of size at most λ.*

Differing-Input Indistinguishability Obfuscation. It is not hard to see that we can recover the notion of differing-inputs indistinguishability obfuscation (dIO) for circuits [7,13,2], by just restricting the above definition of d-pIO to the class $\mathcal{C}' = \{\mathcal{C}'_\lambda\}_{\lambda\in\mathbb{N}}$ of *deterministic* circuits.

This means that the notion of dynamic-input pIO generalizes dIO to randomized circuits. In a recent work by Garg et al. [22], it was shown that assuming strong obfuscation for a specific sampler of circuits and auxiliary inputs, it is impossible to construct differing-input IO for general differing-input samplers over circuits. Since dynamic-input pIO implies differing-input IO, a construction of d-pIO for general dynamic-input indistinguishable samplers is also implausible. However, a construction of d-pIO for specific dynamic-input indistinguishable samplers remains possible, as in the case of dIO.

3.4 Worst-case-input pIO for Circuits

In light of the implausibility of general dynamic-input pIO, we seek for a weaker notion, which is possibly achievable. The resulting notion is what we consider the most natural formalization of IO in the probabilistic setting, but in contrast to X-pIO above, we are only able to *conjecture* the existence of suitable obfuscators.

We first introduce the following class of samplers:

Definition 7 (*Worst-case-input* Indistinguishable Samplers). *The class* $\mathbf{S}^{\mathsf{w\text{-}Ind}}$ *of* worst-case-input indistinguishable samplers *for a circuit family $\mathcal{C}$ contains all circuit samplers $D = \{D_\lambda\}_{\lambda\in\mathbb{N}}$ for $\mathcal{C}$ with the following property: For all adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ where $\mathcal{A}_1$ is an unbounded non-uniform machine and $\mathcal{A}_2$ is PPT, the advantage of $\mathcal{A}$ in the following experiment is negligible.*

Experiment *worst-case-input*-$\mathsf{IND}^{D}_{\mathcal{A}}(1^\lambda)$*:*

```
1. $(C_0, C_1, z) \xleftarrow{\$} D_\lambda$
2. $(x, st) = \mathcal{A}_1(C_0, C_1, z)$            // $\mathcal{A}_1$ is unbounded.
3. $y \xleftarrow{\$} C_b(x)$, where $b \xleftarrow{\$} \{0,1\}$
4. $b' \xleftarrow{\$} \mathcal{A}_2(st, C_0, C_1, z, x, y)$            // $\mathcal{A}_2$ is PPT.
```

The advantage of $\mathcal{A}$ is $\Pr[b' = b] - 1/2$.

This directly yields the notion of worst-case-input pIO:

Definition 8 (Worst-case-input pIO for Randomized Circuits). *A uniform PPT machine* w-$pi\mathcal{O}$ *is a* worst-case-input pIO (or w-pIO) *for randomized circuits, if it is a* pIO *for the class of worst-case-input indistinguishable samplers* $\mathbf{S}^{\mathsf{w\text{-}Ind}}$ *over* $\mathcal{C}$ *that includes all randomized circuits of size at most* λ*.*

Note that in the above definition, since $\mathcal{A}_1$ is computationally unbounded, its best strategy on input (C_0, C_1, z) is to choose (x^*, st^*) that maximizes the guessing advantage of $\mathcal{A}_2$ and hence worst-case-input indistinguishable samplers can be seen as producing pairs of probabilistic circuits satisfying that no efficient adversary ($\mathcal{A}_2$) can distinguish their output $C_0(x)$ or $C_1(x)$ on *any* input x.

The above definition implies a limited form of multi-input indistinguishability: By a hybrid argument, for a worst-case-input sampler the advantage of any adversary $(\mathcal{A}_1, \mathcal{A}_2)$ in the above experiment is negligible even if $\mathcal{A}_1$ can choose a polynomial number of inputs $(x_1, \cdots, x_\ell, st)$ at once and $\mathcal{A}_2$ receives output samples $y_i \overset{\$}{\leftarrow} C_b(x_i)$ for all these inputs, i.e., it is given $(st, C_0, C_1, z, \{x_i\}, \{y_i\})$. However, we cannot prove an adaptive form of multi-input indistinguishability, due to the asymmetric computational powers of $\mathcal{A}_1$ and $\mathcal{A}_2$.

Memory-less worst-case-input pIO*: Forbidding state-passing.* Passing state between $\mathcal{A}_1$ and $\mathcal{A}_2$ in the above definition of worst-case-input indistinguishable samplers appears somewhat unavoidable for any "meaningful" way of defining $\mathbf{S}^{\mathsf{w\text{-}Ind}}$. Indeed, if we have a sampler $D \in \mathbf{S}^{\mathsf{w\text{-}Ind}}$, then for any length function ℓ, we also would like any sampler D' constructed as follows to be also in $\mathbf{S}^{\mathsf{w\text{-}Ind}}$: D'_λ samples (C_0, C_1, z) from the same distribution as D_λ, but instead returns a triple (C'_0, C'_1, z) where C'_b is a circuit such that $C'_b(x; x') = C_b(x)$ for any $x' \in \{0,1\}^{\ell(\lambda)}$, i.e., the last $\ell(\lambda)$ bits of the input are ignored. For such a pair, the adversary $\mathcal{A}_1$ can always use the last $\ell(\lambda)$ bits of the input (which are ignored by the circuit) to pass on some helpful, not efficiently computable, information to $\mathcal{A}_2$ that would help distinguish.

Explicitly forbidding state passing will however be useful when establishing the landscape of relationships among notions below. In particular, we define $\mathbf{S}^{\mathsf{mw\text{-}Ind}}$ as the class of memory-less worst-case-input indistinguishable samplers, which consists of all samplers D for which the advantage in worst-case-input-$\mathsf{IND}^D_{\mathcal{A}}(1^\lambda)$ is negligible for any $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ such that $\mathcal{A}_1$ is unbounded and outputs $st = \perp$, whereas $\mathcal{A}_2$ is PPT. Note that clearly $\mathbf{S}^{\mathsf{w\text{-}Ind}} \subseteq \mathbf{S}^{\mathsf{mw\text{-}Ind}}$. This then is used in the following definition.

Definition 9 (Memory-less worst-case-input pIO for Randomized Circuits). *A uniform PPT machine* mw-$pi\mathcal{O}$ *is a* memory-less worst-case-input pIO (or mw-pIO) *for randomized circuits, if it is a* pIO *for the class of memory-less worst-case-input indistinguishable samplers* $\mathbf{S}^{\mathsf{mw\text{-}Ind}}$ *over* $\mathcal{C}$ *that includes all randomized circuits of size at most* λ*.*

Indistinguishability Obfuscation. The notion of worst-case-input pIO is a direct generalization of IO to the case of randomized circuits. We can recover the original notion of indistinguishability obfuscation (IO) for circuits [7,21] by restricting Definition 8 of worst-case-input pIO to the class $\mathcal{C}' = \{\mathcal{C}'_\lambda\}_{\lambda \in \mathbb{N}}$ of *deterministic* circuits. Also, note that the classes $\mathbf{S}^{\mathsf{mw\text{-}Ind}}$ and $\mathbf{S}^{\mathsf{w\text{-}Ind}}$ are the same (and thus the notion of memory-less worst-case-input and worst-case-input pIO) when restricted to deterministic circuits.

3.5 Relations

In the full version, we prove a number of relations among notions, some of which are quite non-trivial to establish. They are summarized by the following theorem.

Theorem 1 (Relations among pIO notions).

- *A dynamic-input* pIO *obfuscator is also a memory-less worst-case-input* pIO *obfuscator for randomized circuits.*
- *A memory-less worst-case-input* pIO *obfuscator is also a worst-case-input* pIO *obfuscator.*
- *A memory-less worst-case-input* pIO *obfuscator is also an X-*Ind pIO *obfuscator.*

Moreover, all of these implications are strict, i.e., their converses are not true, assuming subexponentially-secure (trapdoor) one-way permutations exist.

3.6 Construction of X-Ind pIO from Sub-exp Indistinguishable IO

In this section, we prove the existence of a construction of an X-Ind pIO obfuscator (as in Definition 4) from sub-exponentially hard IO. It relies on sub-exponentially secure puncturable PRFs, which we now recall

Definition 10 (Puncturable PRFs). *A puncturable family of PRFs is given by a triple of uniform PPT machines* Key, Puncture, *and* PRF, *and a pair of computable functions $n(\cdot)$ and $m(\cdot)$, satisfying the following conditions:*

Correctness. *For all outputs K of* $\mathrm{Key}(1^\lambda)$, *all points $i \in \{0,1\}^{n(\lambda)}$, and $K_{-i} = \mathrm{Puncture}(K, i)$, we have that $\mathsf{PRF}(K_{-i}, x) = \mathsf{PRF}(K, x)$ for all $x \neq i$.*

Pseudorandom at punctured point. *For every PPT adversary $(\mathcal{A}_1, \mathcal{A}_2)$, there is a neligible function μ, such that in an experiment where $\mathcal{A}_1(1^\lambda)$ outputs a point $i \in \{0,1\}^{n(\lambda)}$ and a state σ, $K \xleftarrow{\$} \mathrm{Key}(1^\lambda)$ and $K_{-i} = \mathrm{Puncture}(K, i)$, the following holds*

$$\left|\Pr[\mathcal{A}_2(\sigma, K_{-i}, i, \mathsf{PRF}(K, i)) = 1] - \Pr[\mathcal{A}_2(\sigma, K_{-i}, i, U_{m(\lambda)}) = 1]\right| \leq \mu(\lambda)$$

As observed by [12,14,30], the GGM tree-based construction of PRFs [25] from PRGs yields puncturable PRFs. Furthermore, if the PRG underlying the GGM construction is sub-exponentially hard (and this can in turn be built from sub-exponentially hard OWFs), then the resulting puncturable PRF is sub-exponentially pseudo-random.

We are now ready to move to our theorem. Its formal proof is deferred to the full version, but we give a detailed description of the main ideas below.

Theorem 2 (Existence of X-Ind pIO.). *Assume the existence of a sub-exponentially indistinguishable indistinguishability obfuscator $i\mathcal{O}$ for circuits and a sub-exponentially secure puncturable PRF* (Key, Puncture, PRF)*. Then, there exists a X-Ind pIO obfuscator X-$pi\mathcal{O}$ for randomized circuits.*

We first describe our construction of X-Ind pIO, denoted as X-$pi\mathcal{O}$. Recall that by our assumption, both $i\mathcal{O}$ and the puncturable PRF (Key, Puncture, PRF) have a $2^{-\lambda^{\epsilon}}$ distinguishing gap for some constant $\epsilon \in (0,1)$ and any non-uniform PPT distinguisher. Also, in the following, we implicitly identify strings with integers (via their binary encoding) and vice versa.

Construction X-$pi\mathcal{O}$: On input 1^{λ} and a probabilistic circuit C of size at most λ, proceed as follows:

1. Let $\lambda' = \lambda'(\lambda) = (\lambda \log^2(\lambda))^{1/\epsilon}$. Sample a key of the PRF function $K \leftarrow \mathrm{Key}(1^{\lambda'})$.
2. Construct deterministic circuit $E^{(C,K)}$ which outputs $C(x\ ;\ \mathsf{PRF}(K,x))$. By construction the size of $E^{(C,K)}$ is bounded by a polynomial $p(\lambda') \geq \lambda'$ in λ'.
3. Let $\lambda'' = p(\lambda') \geq \lambda'$. Obfuscate $E^{(C,K)}$ using $i\mathcal{O}$, $\Lambda \overset{\$}{\leftarrow} i\mathcal{O}(1^{\lambda''}, E^{(C,K)})$.
4. Output Λ.

To see why the construction works, consider two circuits C_1, C_2 sampled satisfying the indistinguishability requirement imposed by X-Ind pIO, their obfuscation are the IO obfuscated programs Λ_1, Λ_2 of the two derandomized circuits D_1^k, D_2^k. The challenge lies in how to apply the security guaranetees of IO on two circuits D_1^k, D_2^k that have completely different functionality. Our hope is to leverage the fact that the original circuits C_1, C_2 are strongly indistinguishable together with the sub-exponential pseudo-randomness of PRF; indeed, when the PRF key is sufficiently long, it holds that for every x, the output pair $D_1^k(x)$ and $D_2^k(x)$ is $\frac{1}{X2^{\omega(\log(\lambda))}}$-indistinguishable. Thus by a simple union bound over all X inputs, even the entire truth tables $\left\{D_1^{k_1}(x)\right\}$, $\left\{D_2^{k_2}(x)\right\}$ are indistinguishable. However, even given such strong guarantees, it is still not clear how to apply IO.

We overcome the challenge by considering a sequence of $X+1$ hybrids $\{H_i\}$, in which we obfuscate a sequence of "hybrid circuits" $\left\{E_i^k(x)\right\}$ that "morph" gradually from D_1^k to D_2^k. More specifically, circuit E_i^k evaluates the first i inputs using D_2^k, and the rest using D_1^k. In any two subsequent hybrids, the circuits E_{i-1}^k and E_i^k only differ at whether the i'th input is evaluated using D_1^k or D_2^k. Consider additionally two auxiliary hybrids H_{i-1}^+, H_i^+ where two circuits $F_{i-1}^{k_{-i},y}, F_i^{k_{-i},y'}$ modified from E_{i-1}^k, E_i^k are obfuscated; they proceed the same as E_{i-1}^k, E_i^k respectively, except that they use internally a PRF key k_{-i} punctured at point i, and output directly y and y' for input i respectively. Then, when y and y' are programmed to be exactly $y = E_{i-1}^k(i) = D_1^k(i)$ and $y' = E_i^k(i) = D_2^k(i)$, the two circuits compute exactly the same functionality as E_{i-1}^k, E_i^k. By IO, these auxilary hybrids are indistinguishable from hybrids H_{i-1} and H_i respectively.

Then, by the fact that $y = E^k_{i-1}(i) = D^k_1(i)$ and $y' = E^k_i(i) = D^k_2(i)$ are indistinguishable (which in turn relies on the pseudo-randomness of the PRF function), the two auxiliary hybrids H^+_{i-1}, H^+_i are indistinguishable, and thus so are H_{i-1} and H_i. Furthermore, since the distinguishing gap of IO and PRF are bounded by $\frac{1}{X2^{\omega(\log \lambda)}}$, it follows from a hybrid argument that H_0 and H_X, which contain the IO obfuscations of $D^k_1(x)$ and $D^k_2(x)$, respectively, are $\frac{1}{2^{\omega(\log \lambda)}}$-indistinguishable.
Other notions. While we cannot prove this statement in any meaningful model, we also conjecture that the same construction is $\mathsf{w\text{-}pIO}$ obfuscator for randomized circuits.

4 Application 1: Fully Homomorphic Encryption

We now describe how to construct leveled and fully homomorphic encryption schemes using different notions of pIO. (See the Introduction for an overview of the constructions.)

4.1 Trapdoor Encryption Schemes

Trapdoor encryption schemes have two modes: In the honest mode, an honest public key is sampled and the encryption and decryption algorithms work as in a normal CPA-secure encryption scheme with semantic security and correctness; additionally, there is a "trapdoor mode", in which a indistinguishable "trapdoor public key" is sampled and the encryption algorithm produces ciphertexts that may have stronger indistinguishability properties than these in the honest mode, at the price of losing correctness. More precisely,

Definition 11 (Trapdoor Encryption Scheme). *We say that* $\Pi = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{tKeyGen})$ *is a trapdoor encryption scheme, if* $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *is a CPA-secure encryption scheme and the trapdoor key generation algorithm* $\mathsf{tKeyGen}$ *satisfies the following additionally properties:*

Trapdoor Public Keys: *The following two ensembles are indistinguishable:*

$$\left\{(\mathsf{pk}, \mathsf{sk}) \xleftarrow{\$} \mathsf{KeyGen}(1^\lambda) \ : \ \mathsf{pk}\right\}_\lambda \approx \left\{\mathsf{tpk} \xleftarrow{\$} \mathsf{tKeyGen}(1^\lambda) \ : \ \mathsf{tpk}\right\}_\lambda$$

Computational hiding: *The following ensembles are indistinguishable.*

$$\left\{\mathsf{tpk} \xleftarrow{\$} \mathsf{tKeyGen}(1^\lambda) \ : \ \mathsf{Enc}_{\mathsf{tpk}}(0)\right\}_\lambda \approx \left\{\mathsf{tpk} \xleftarrow{\$} \mathsf{tKeyGen}(1^\lambda) \ : \ \mathsf{Enc}_{\mathsf{tpk}}(1)\right\}_\lambda$$

The basic definition of trapdoor encryption scheme only requires encryption of different bits under a freshly generated trapdoor public key to be computationally indistinguishable. As discussed before, this definition is a generalization of CPA encryption in the following sense,

Lemma 1. *Let* $\Pi' = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *be a CPA-encryption scheme. Then* $\Pi = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{tKeyGen} = \mathsf{KeyGen})$ *is a trapdoor encryption scheme.*

The basic trapdoor encryption scheme does not provide any advantage in the trapdoor mode than the honest mode. Below, we consider two stronger security properties in the trapdoor mode.

Definition 12 (Statistical Trapdoor Encryption Scheme). *We say that trapdoor encryption scheme $\Pi = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{tKeyGen})$ is a statistical trapdoor encryption scheme, if the computational hiding property in Definition 11 is replaced by the following.*

Statistical hiding: *The following ensembles are statistically close.*

$$\left\{\mathsf{tpk} \overset{\$}{\leftarrow} \mathsf{tKeyGen}(1^\lambda) \ : \ \mathsf{Enc}_{\mathsf{tpk}}(0)\right\}_\lambda \approx_s \left\{\mathsf{tpk} \overset{\$}{\leftarrow} \mathsf{tKeyGen}(1^\lambda) \ : \ \mathsf{Enc}_{\mathsf{tpk}}(1)\right\}_\lambda$$

We note that any lossy encryption scheme as defined by Bellare, Hofheinz and Yilek [8] implies a statistical trapdoor encryption scheme. A lossy encryption scheme has a key generation algorithm KeyGen that takes as input the security parameter 1^λ and additionally a variable $m \in \{\mathsf{injective}, \mathsf{lossy}\}$ indicating whether to generate a key in the injective mode or in the lossy mode. A key generated in the injective mode ensures decryption correctness and semantic security, whereas a key generated in the lossy mode statistically loses information of the plaintexts, that is, encryption of different bits are statistically close. Therefore, we have:

Lemma 2. *Let $\Pi' = (\mathsf{Gen}', \mathsf{Enc}, \mathsf{Dec})$ be a lossy encryption scheme. Then $\Pi = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{tKeyGen})$ where $\mathsf{KeyGen}(1^\lambda) = \mathsf{Gen}'(1^\lambda, \mathsf{injective})$ and $\mathsf{KeyGen}(1^\lambda) = \mathsf{Gen}'(1^\lambda, \mathsf{lossy})$, is a statistical trapdoor encryption scheme.*

Definition 13 (μ-Hiding Trapdoor Encryption Scheme). *Let μ be any function We say that trapdoor encryption scheme $\Pi = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{tKeyGen})$ is a μ-Lossy trapdoor encryption scheme, if the computational hiding property in Definition 11 is replaced by the following.*

μ-hiding: *For any non-uniform PPT adversary $\mathcal{A}$, the following holds:*

$$\Big| \Pr[\mathsf{tpk} \overset{\$}{\leftarrow} \mathsf{tKeyGen}(1^\lambda) \ : \ \mathcal{A}(\mathsf{Enc}_{\mathsf{tpk}}(0)) = 1] \\ - \Pr[\mathsf{tpk} \overset{\$}{\leftarrow} \mathsf{tKeyGen}(1^\lambda) \ : \ \mathcal{A}(\mathsf{Enc}_{\mathsf{tpk}}(1)) = 1]\Big| \leq \mu(\lambda)$$

where μ is called the distinguishing gap.

One of the instantiations of our general transformation for obtaining FHE relies on sub-exponentially indistinguishable IO and a μ-hiding trapdoor encryption scheme where μ is bounded by $\mathsf{negl}(\lambda)2^{-2l(\lambda)}$ and $l(\lambda)$ is an upper bound on the length of the ciphertext. In other words, the distinguishing gap is much smaller than the inverse exponentiation of the ciphertext length. We construct such a μ-hiding trapdoor encryption scheme using a μ-rerandomizable encryption. In fact, our construction achieves the stronger property of perfect hiding, that is, $\mu = 0$.

Definition 14 (μ-Rerandomizable Encryption Scheme). *We say that a quadruple of uniform PPT algorithms $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{reRand})$ is a μ-rerandomizable encryption scheme, if $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ is a CPA-secure encryption scheme, and additionally the algorithm reRand satisfies the following property:*

μ-Rerandomizability: *For every non-uniform PPT adversary $\mathcal{A}$, the following holds for every $\lambda \in \mathbb{N}$.*

$$\Big| \Pr[\mathcal{A}(\mathsf{pk}, c_0, c_1, \mathsf{reRand}_{\mathsf{pk}}(c_0)) = 1] - \Pr[\mathcal{A}(\mathsf{pk}, c_0, c_1, \mathsf{reRand}_{\mathsf{pk}}(c_1)) = 1] \Big| \le \mu(\lambda)$$

where $(\mathsf{pk}, \mathsf{sk}) \xleftarrow{\$} \mathsf{Gen}(1^\lambda)$, $c_0 \xleftarrow{\$} \mathsf{Enc}_{\mathsf{pk}}(b)$ and $c_1 \xleftarrow{\$} \mathsf{Enc}_{\mathsf{pk}}(b)$.

We way that Π is perfectly re-randomizable, if the distinguishing gap μ above is zero.

Many encryption scheme such as ElGamal, Goldwasser-Micali [27], Paillier [33], Damgård-Jurik [20], are in fact perfectly rerandomizable as per [36,28] and satisfy our definition. Furthermore, we show that

Lemma 3. *Let μ be a negligible function. Every μ-rerandomizable CPA encryption scheme can be transformed into a μ-hiding trapdoor encryption scheme.*

An overview of the construction was provided in the Introduction (See "LHE via trapdoor encryption"). We defer the formal construction and proof to the full version [19].

4.2 From Trapdoor Encryption to Leveled Homomorphic Encryption

In this section, we present our general transformation from a trapdoor encyrption scheme $\Pi = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{tKeyGen})$ to a leveled fully homomorphic encryption scheme LHE, relying on a pIO scheme $pi\mathcal{O}$ for a specific class $\mathbf{S}^{\Pi}$ of samplers defined by Π as described in Figure 3; (more explanation on the class is provided in the proof of semantic security).

Proposition 2. *Let Π be any trapdoor encryption scheme. Assume the existence of pIO for the class of samplers $\mathbf{S}^{\Pi}$ defined by Π as in Figure 3. Then, Π can be transformed into a leveled homomorphic encryption scheme.*

Below we first describe our construction and then prove its correctness and semantic security in Lemma 4 and 5. Without loss of generality, we assume that the public, secret keys and ciphertexts of Π have lengths bounded by $l(\lambda)$. Below we first describe our construction.

Construction of LHE: Let $L = L(\lambda)$ be the depth of the circuits that we want to evaluate. The four algorithms of the scheme proceed as follows:

– **Key generation:** $\mathsf{LHE.Keygen}(1^\lambda, 1^L)$ does the following for every level i from 0 to L.
  - samples a pair of keys $(\mathsf{pk}_i, \mathsf{sk}_i) \overset{\$}{\leftarrow} \mathsf{KeyGen}(1^\lambda)$ of Π;
  - for $i \geq 1$, obfuscate the circuit $P_i = \mathsf{Prog}^{(\mathsf{sk}_{i-1}, \mathsf{pk}_i)}$ as described in Figure 2, that is, sample $\Lambda_i \overset{\$}{\leftarrow} pi\mathcal{O}(1^s, P_i)$ where the security parameter $s = s(\lambda)$ for obfuscation is an upper-bound on the size of all P_i's. [6]

  Finally outputs $\mathsf{pk} = \mathsf{pk}_0$, $\mathsf{sk} = \mathsf{sk}_L$, $\mathsf{evk} = \{P_i\}_{0 \leq i \leq L}$.
– **Encryption:** $\mathsf{LHE.Enc}_{\mathsf{pk}}(m)$ outputs a fresh encryption of m under $\mathsf{pk} = \mathsf{pk}_0$ using Π, $c \overset{\$}{\leftarrow} \mathsf{Enc}_{\mathsf{pk}_0}(m)$.
– **Decryption:** $\mathsf{LHE.Dec}_{\mathsf{sk}}(c)$ decrypts c using the secret key $\mathsf{sk} = \mathsf{sk}_L$ to obtain $m = \mathsf{Dec}_{\mathsf{sk}_L}(c)$.
– **Homomorphic evaluation:** $\mathsf{LHE.Eval}_{\mathsf{evk}}(C, c_1, \ldots, c_\ell)$ on input a layered circuit C (consisting of only NAND gates) of depth at most L, evaluate C layer by layer; in iteration i, layer $i \in [L]$ is evaluated (the first layer is connected with the input wires): At the onset of this iteration, the values of the input wires of layer i has been homomorphically evaluated in the previous iteration and encrypted under key pk_{i-1} (in the first iteration, these encryptions are simply $c_1, \cdots, c_\ell$); for each NAND gate g in this layer i, let $\alpha(g), \beta(g)$ be encryption of the values of its input wires; evaluate g homomorhpically by computing $\gamma(g) = \Lambda_i(\alpha(g), \beta(g))$ to obtain an encryption of the value of g's output wire under public key pk_i. At the end, output the encryptions generated in the last iteration L.

sk, pk, tpk, α, and β are strings of length $l(\lambda)$.

Circuit $\mathsf{Prog}^{(\mathsf{sk},\mathsf{pk})}(\alpha, \beta)$**:** Decrypt α and β to obtain $a = \mathsf{Dec}_{\mathsf{sk}}(\alpha)$ and $b = \mathsf{Dec}_{\mathsf{sk}}(\beta)$; output $\gamma \overset{\$}{\leftarrow} \mathsf{Enc}_{\mathsf{pk}}(a \ \mathsf{NAND}\ b)$.
Circuit $\mathsf{tProg}^{(\mathsf{tpk})}(\alpha, \beta)$**:** Output $\gamma \overset{\$}{\leftarrow} \mathsf{Enc}_{\mathsf{tpk}}(0)$.

Both circuits are padded to their maximum size. Let $s(\lambda)$ be an upper bound on their sizes.

Fig. 2. Circuits used in the construction of LHE and its analysis

It follows from the correctness of pIO and Π that the scheme LHE is correct; we refer the reader to the full version [19] for a formal proof.

Lemma 4. *If* pIO *and* Π *are correct, then* LHE *has homomorphism.*

Proof of Semantic Security of LHE Towards establishing the semantic security of LHE, we rely on the security property of pIO for the class of

[6] This is because the obfuscator $pi\mathcal{O}(1^\lambda, C)$ works with classes of circuits $\mathcal{C}_\lambda$ of size at most λ.

samplers $\mathbf{S}^{\Pi}$ defined by the trapdoor encryption scheme Π used in LHE. Roughly speaking, samplers in $\mathbf{S}^{\Pi}$ samples pairs of circuits where one of them is identical the "honest" program used for generating the evaluation key in LHE, except that a trapdoor public key tpk (instead of an honest public key) is hardwired in (that is, $\mathsf{Prog}^{(\mathsf{sk},\mathsf{tpk})}$), and the other one is a "trapdoor" program $\mathsf{tProg}^{(\mathsf{tpk})}$ as described in Figure 2 that always generates a ciphertext of 0 under the "trapdoor" public key hardwired inside. More precisely, we describe the class of samplers in Figure 3.

$\Pi = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{tKeyGen})$ is a trapdoor encryption scheme, $SK = \{sk_\lambda\}$ is a sequence of strings of length $l(\lambda)$, and $s(\lambda)$ is an upper bound on the sizes of programs $\mathsf{Prog}^{(sk,\mathsf{tpk})}$ and $\mathsf{tProg}^{(\mathsf{tpk})}$.

The Sampler D^{SK}: The distribution D_s^{SK} samples a trapdoor public key $\mathsf{tpk} \overset{\$}{\leftarrow} \mathsf{tKeyGen}(1^\lambda)$, and outputs $C_0 = \mathsf{Prog}^{(sk,\mathsf{tpk})}$, $C_1 = \mathsf{tProg}^{(\mathsf{tpk})}$ and $z = \mathsf{tpk}$, where $sk = \mathsf{sk}_\lambda$.

The Class $\mathbf{S}^{\Pi}$: Let $\mathbf{S}^{\Pi}$ be the class of samplers that include distribution ensembles D^{SK} for all sequence of strings SK of length $l(\lambda)$.

Fig. 3. The class of samplers for proving the semantic security of LHE.

Next we show that LHE is semantic secure. We note that for the proof to go through, we only rely on the fact that $pi\mathcal{O}$ is a pIO for the above described class $\mathbf{S}^{\Pi}$ and the fact that trapdoor public keys of the trapdoor encryption scheme Π are indistinguishable from honest public keys. The proof actually does not depend on any hiding property in the trapdoor mode, which will only play a role later when instantiating pIO for $\mathbf{S}^{\Pi}$.

Lemma 5. *Assume that Π is a trapdoor encryption scheme and $pi\mathcal{O}$ is a* pIO *for the class of samplers $\mathbf{S}^{\Pi}$ in Figure 3. Then,* LHE *is semantically secure.*

Proof. Fix any polynomial time adversary $\mathcal{A}$. We want to show that for every $\lambda \in \mathbb{N}$, it holds that, the advantage of the adverary $\mathrm{Adv}_{\mathrm{CPA}}[\mathcal{A}]$ is negligible.

$$|\Pr[\mathcal{A}(\mathsf{pk}, \mathsf{evk}, \mathsf{LHE.Enc}_{\mathsf{pk}}(0)) = 1] - \Pr[\mathcal{A}(\mathsf{pk}, \mathsf{evk}, \mathsf{LHE.Enc}_{\mathsf{pk}}(1)) = 1]| < \mathsf{negl}(\lambda) ,$$

where $(\mathsf{pk}, \mathsf{evk}, \mathsf{sk}) \leftarrow \mathsf{LHE.Keygen}(1^\lambda)$.

Towards this, we consider two sequences of hybrids $H_0^b, \cdots, H_L^b$ for $b \in \{0, 1\}$. H_0^b is exactly an honest CPA game with the adversary $\mathcal{A}$ where it receives a challenge ciphertext that is an encryption of b; in intermediate hybrids, the adversary $\mathcal{A}$ participates in a modified game. We show that for every two subsequent hybrids H_i^b, H_{i+1}^b, as well as H_L^0, H_L^1, the view of $\mathcal{A}$ is indistinguishable. Below we formally describe all the hybrids.

Hybrid H_0^b: Hybrid H_0^b is an honest CPA game with $\mathcal{A}$, where $\mathcal{A}$ receives $(\mathsf{pk}, \mathsf{evk}, c^* = \mathsf{LHE.Enc}_{\mathsf{pk}}(b))$ for freshly sampled $(\mathsf{pk}, \mathsf{evk}, \mathsf{sk}) \xleftarrow{\$} \mathsf{LHE.Keygen}(1^\lambda)$. By construction of LHE, the view of $\mathcal{A}$ is,

$$\mathsf{view}[\mathcal{A}]_0^b = \big(\mathsf{pk} = \mathsf{pk}_0, \mathsf{evk} = (\Lambda_1, \cdots, \Lambda_L), c_b = \mathsf{Enc}_{\mathsf{pk}_0}(b)\big)$$

Hybrid H_i^b for $i > 0$: Hybrid H_i^b proceeds identically to H_0^b except that the evaluation key evk is sampled in a different way. Recall that in H_0^b, evk consists of the obfuscated circuits $\Lambda_1, \cdots, \Lambda_L$ of circuits $P_1, \cdots, P_L$, where $P_j = \mathsf{Prog}^{(sk_{j-1}, pk_j)}$. In H_i^b, the last i circuits $P_{L-i+1}, \cdots, P_L$ are replaced with $tP_{L-i+1}, \cdots, tP_L$, where $tP_j = \mathsf{tProg}^{(\mathsf{tpk}_j)}$ (see Figure 2) hardwired with a freshly sampled "trapdoor" public key $\mathsf{tpk}_j \xleftarrow{\$} \mathsf{tKeyGen}(1^\lambda)$. Let $t\Lambda_{L-i+1}, \cdots, t\Lambda_L$ be the obfuscated circuits of $tP_{L-i+1}, \cdots, tP_L$. Then evk_i in H_i^b consists of $\mathsf{evk}_i = \Lambda_1, \cdots, \Lambda_{L-i}, t\Lambda_{L-i+1}, \cdots, t\Lambda_L$. The view of A in H_i^b is

$$\mathsf{view}[\mathcal{A}]_i^b = \big(\mathsf{pk}_0, \mathsf{evk}_i = (\Lambda_1, \cdots, \Lambda_{L-i}, t\Lambda_{L-i+1}, \cdots, t\Lambda_L), c_b = \mathsf{Enc}_{\mathsf{pk}_0}(b)\big)$$

To show that the $\mathcal{A}$ cannot distinguish the two CPA games, it is equivalent to show that $\mathcal{A}$ cannot distinguish hybrids H_0^0 and H_0^1. Towards this, it suffices to prove that $\mathcal{A}$ cannot distinguish any of the neighboring hybrids, that is,

- The views of $\mathcal{A}$ in H_L^0 and H_L^1 are indistinguishable,
- For every b and $0 \leq i \leq L$, the views of $\mathcal{A}$ in H_i^b and H_{i+1}^b are indistinguishable,

Towards showing the first indistinguishability, we observe that in H_L^0 and H_L^1, the evaluation key evk_L consists of only obfuscation of the "trapdoor" programs $\{t\Lambda_i \xleftarrow{\$} pi\mathcal{O}(1^s, \mathsf{tProg}^{(\mathsf{tpk}_j)})\}$ which does not depend on any secret key sk_j. Thus by the semantic security of Π, encryption $\mathsf{Enc}_{\mathsf{pk}_0}(0)$ and $\mathsf{Enc}_{\mathsf{pk}_0}(1)$ are indistinguishable, and hence so are the views of $\mathcal{A}$ in H_L^0 and H_L^1.

Towards showing the second indistinguishability, we observe that the only difference between H_i^b and H_{i+1}^b lies in whether the evaluation key contains an obfuscation Λ_{L-i} of the honest program $\mathsf{Prog}^{(\mathsf{sk}_{L-i-1}, \mathsf{pk}_{L-i})}$ for layer $L-i$, or an obfuscation $t\Lambda_{L-i}$ of the trapdoor program $\mathsf{tProg}^{(\mathsf{tpk}_{L-i})}$. Furthermore, in both H_i^b and H_{i+1}^b the generation of the evaluation key does not depend on sk_{L-i}, and hence neither do the views of $\mathcal{A}$. Thus to show the indistinguishability of the views of $\mathcal{A}$ it suffices to show the indistinguishability of the following ensembles, from which the views of A in H_i^b and H_{i+1}^b can be reconstructed.

$$\big\{\Lambda_{L-i}, \mathsf{pk}_{L-i}, \mathsf{pk}_{L-i-1}, \big)\big\}_\lambda \approx \big\{t\Lambda_{L-i}, \mathsf{tpk}_{L-i}, \mathsf{pk}_{L-i-1}\big)\big\}_\lambda$$

where in the above distributions $(\mathsf{pk}_{L-i}, \mathsf{sk}_{L-i})$ and $(\mathsf{pk}_{L-i-1}, \mathsf{sk}_{L-i-1})$ are all randomly sampled honest keys of Π, tpk_{L-i} is a randomly sampled trapdoor public key, and Λ_{L-i} and $t\Lambda_{L-i}$ are obfuscations of the honest program or the trapdoor program as in H_i^b and H_{i+1}^b. We argue why the views of $\mathcal{A}$ in H_i^b and H_{i+1}^b can be reconstructed from the left and right random variables respectively: This is because Λ_{L-i} and $t\Lambda_{L-i}$ correspond respectively to the $(L-i)$'th obfuscation in the evaluation key in H_i^b and H_{i+1}^b, and the other obfuscated programs $\Lambda_1, \cdots \Lambda_{L-i-1}, t\Lambda_{L-i+1}, \cdots, t\Lambda_L$ in the evaluation key can be sampled efficiently given pk_{L-i-1} together with pk_{L-i} or tpk_{L-i}; finally, encryption of b under pk_0 can be sampled independently.

We show the above indistinguishability in two steps, via an intermediate hybrid where an obfuscation Λ'_{L-i} of the hybrid program $\mathsf{Prog}^{(\mathsf{sk}_{L-i-1}, \mathsf{tpk}_{L-i})}$ is sampled; the hybrid program is the same as the honest program except that a trapdoor public key tpk_{L-i} is hardwired.

$$\left\{\boxed{\Lambda_{L-i}, \mathsf{pk}_{L-i}}, \mathsf{pk}_{L-i-1}, \right\}_\lambda \approx \left\{\boxed{\Lambda'_{L-i}, \mathsf{tpk}_{L-i}}, \mathsf{pk}_{L-i-1}, \right\}_\lambda \tag{1}$$

$$\left\{\boxed{\Lambda'_{L-i}}, \mathsf{pk}_{L-i}, \mathsf{pk}_{L-i-1}, \right\}_\lambda \approx \left\{\boxed{t\Lambda_{L-i}}, \mathsf{tpk}_{L-i}, \mathsf{pk}_{L-i-1}\right\}_\lambda \tag{2}$$

Equation (1) follows directly from the fact that a randomly sampled trapdoor public key is indistinguishable from an honest public key.

Equation (2) holds following the pIO security for the class of samplers $\mathbf{S}^\Pi$. More specifically, to show the equation, it suffices to show that it holds for every fixed sequence of pairs $S = \left\{(\mathsf{pk}_{L-i-1}, \mathsf{sk}_{L-i-1})\right\}$ of length $l(\lambda)$ each. Fix such a sequence S and let $SK = \{\mathsf{sk}_{L-i-1}\}$ be the sequence of secret keys only. Notice that the sampler D^{SK} described in Figure 3 produces exactly the hybrid and trapdoor programs as above, that is,

$$(C_0 = \mathsf{Prog}^{(\mathsf{sk}_{L-i-1}, \mathsf{tpk}_{L-i})}, C_1 = \mathsf{tProg}^{(\mathsf{tpk}_{L-i})}, z = \mathsf{tpk}_{L-i}) \xleftarrow{\$} D_s^{SK}$$

Thus for the fixed sequence S, Equation (2) is equivalent to the following:

$$\left\{(C_0, C_1, z) \xleftarrow{\$} D_s^{\mathsf{sk}_{L-i-1}} \; : \; (C_0, C_1, pi\mathcal{O}(1^s, C_0), z)\right\}_\lambda$$
$$\approx \left\{(C_0, C_1, z) \xleftarrow{\$} D_s^{\mathsf{sk}_{L-i-1}} \; : \; (C_0, C_1, pi\mathcal{O}(1^s, C_1), z)\right\}_\lambda$$

This indistinguishability follows directly from the premise that $pi\mathcal{O}$ is a pIO for the sampler D^{SK}. Thus the views of $\mathcal{A}$ in H_i^b and H_{i+1}^b are indistinguishable.

Instantiation of LHE We show how to instantiate our general transformation from any trapdoor encryption scheme to a LHE scheme, more precisely, how to realize the premise of Proposition 2.

Instantiation 1: Rerandomizable Encryption + Sub-exponential IO. The first instantiation uses a ν-hiding trapdoor encryption scheme Π and a X-Ind pIO for appropriate functions ν and X. Let us specify the functions: First, set $\nu(\lambda) = \mathsf{negl}(\lambda) 2^{-2l(\lambda)}$, where $l(\lambda)$ is an upper bound on the lengths of the ciphertexts of Π. Second, to set the function X, recall that every sampler D_s^{SK} [7] in the class $\mathbf{S}^\Pi$ produces circuits $C_0 = \mathsf{Prog}^{(\mathsf{sk}, \mathsf{tpk})}$ and $C_1 = \mathsf{tProg}^{(\mathsf{tpk})}$ of size $s(\lambda)$ and input

[7] We remind the reader that all variables related with the encryption scheme Π, such as $\mathsf{pk}, \mathsf{sk}, \mathsf{tpk}$, are generated using security parameter λ, while the pIO scheme $pi\mathcal{O}$ and the related samplers all use security parameter $s = s(\lambda)$.

length $2l(\lambda)$; by setting $X(s(\lambda)) = 2^{2l(\lambda)}$, we have that the two sampled circuits C_0, C_1 differ at most $X(s)$ inputs and the output distributions of C_0 and C_1 are $\mathsf{negl}(\lambda)X(\lambda)^{-1}$-indistinguishable following from the ν-hiding property of Π. Therefore D^{SK} is an X-Ind sampler.

Therefore, any X-Ind pIO scheme is a pIO scheme matching the ν-hiding trapdoor encryption scheme Π. Furthermore, by Lemma 3, the existence of a ν-rerandomizable encryption scheme (in particular, a perfectly rerandomizable one) implies that of a ν-hiding trapdoor encryption scheme. By Theorem 2, X-Ind pIO can be constructed from any sub-exponentially indistinguishable IO and sub-exponentially secure OWFs. Therefore, following Proposition 2, we have

Corollary 1 (LHE from rerandomizable encryption and sub-exp secure IO and OWF.). *Let Π be a perfectly rerandomizable encryption scheme. Assume the existence of sub-exponentially indistinguishable IO for circuits and sub-exponentially secure one-way functions. Π can be turned into a leveled homomorphic encryption scheme.*

Instantiation 2: Lossy Encryption + worst-case-input pIO*.* The second instantiation combines a lossy encryption scheme, which by Lemma 2 directly implies a statistical trapdoor encryption scheme Π, with a worst-case-input pIO. By the statistical hiding property of the trapdoor mode of Π, every sampler D^{SK} in the class $\mathbf{S}^{\Pi}$ corresponding to Π samples circuits $C_0 = \mathsf{Prog}^{(\mathsf{sk},\mathsf{tpk})}$ and $C_1 = \mathsf{tProg}^{(\mathsf{tpk})}$ with statistically close output distributions for every input. Therefore, D^{SK} is a worst-case-input indistinguishable sampler. In other words, any worst-case-input pIO is a pIO for the class $\mathbf{S}^{\Pi}$. Following Proposition 2,

Corollary 2 (LHE from lossy encryption and worst-case-input pIO). *Let Pi be a lossy encryption scheme. Assume the existence of a worst-case-input* pIO *scheme pi$\mathcal{O}$. Then, Π can be transformed into a leveled homomorphic encryption scheme.*

Instantiation 3: CPA Encryption + Dynamic-input pIO *for Specific Class.* Finally, we observe that any CPA encryption Π can be turned into a LHE, if there exits a strong notion of pIO, namely dynamic-input pIO for $\mathbf{S}^{\Pi}$. As observed in Lemma 1, any CPA encryption scheme $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ directly implies a trapdoor encryption scheme $\Pi' = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{tKeyGen} = \mathsf{Gen})$ with a computationally hiding trapdoor mode. This implies that every sampler D^{SK} in the matching class $\mathbf{S}^{\Pi}$ is a dynamic-input indistinguishable sampler. Therefore,

Corollary 3. *Let Π be any CPA encryption scheme and Π' the corresponding trapdoor encryption scheme. Assume the existence of a dynamic-input* pIO *scheme pi$\mathcal{O}$ for $\mathbf{S}^{\Pi'}$. Then, Π can be transformed into a leveled homomorphic encryption scheme.*

We note that although general pIO for all dynamic-input indistinguishable samplers is implausible by [22], pIO for the specific class of samplers $\mathbf{S}^{\Pi'}$ circumvents the implausibility result. This is because the implausibility of [22] applies only to a specific class of samplers that produce (C_0, C_1, z) where z is an obfuscated program that essentially distinguishes circuits with the same functionality as C_0 from ones with the same functionality as C_1 using only their I/O interfaces. However, samplers in $\mathbf{S}^{\Pi'}$ produce auxiliary input that is a public key pk of Π, which cannot be used to tell apart circuits of functionalities identical to $\mathsf{Prog}^{(sk,pk)}$ or $\mathsf{tProg}^{(pk)}$ through only their I/O interfaces, due to the semantic security of Π. Therefore, dynamic-input pIO for $\mathbf{S}^{\Pi'}$ circumvents the implausibility. We consider the same construction of X-Ind pIO as a potential candidate construction of dynamic-input pIO for $\mathbf{S}^{\Pi'}$.

4.3 From LHE to FHE

In this section, we show how to transform any leveled homomorphic encryption scheme LHE with a fixed decryption depth into a fully homomorphic one, *without relying on circular security.* More specifically,

– we say that a LHE scheme $\mathsf{LHE} = (\mathsf{LHE.Keygen}, \mathsf{LHE.Enc}, \mathsf{LHE.Dec}, \mathsf{LHE.Eval})$ has a *fixed decryption depth* $D_{\mathsf{LHE.Dec}}(\cdot)$, if for every polynomial depth L, every $(\mathsf{pk}, \mathsf{sk}, \mathsf{evk})$ in the support of $\mathsf{LHE.Keygen}(1^\lambda, 1^{L(\lambda)})$, every freshly generated or homomorphically evaluated ciphertext c^* in the support of $\mathsf{LHE.Enc}(\mathsf{pk}, \cdot)$ or $\mathsf{LHE.Eval}(\mathsf{pk}, (C, \cdots))$ with a depth $L(\lambda)$ circuit C, the decryption algorithm $\mathsf{LHE.Dec}_{\mathsf{sk}}(c^*)$ has depth bounded by $D_{\mathsf{LHE.Dec}}(\lambda)$.

We now sketch a general transformation that turns any LHE scheme with a fixed decryption depth into a FHE. The transformation proceeds in two steps.

A "imaginary" FHE with a non-succinct evaluation key: In a first step, imagine a FHE scheme with an evaluation key evk that consists of a super-polynomial number $\mathcal{L}(\lambda)$ of layer evaluation keys each of size $\mathrm{poly}(\lambda)$. Each layer $\ell \in [\mathcal{L}]$ is associated with a key tuple $(\mathsf{pk}_\ell, \mathsf{sk}_\ell, \mathsf{evk}_\ell)$ of LHE that supports evaluating circuits of depth $D' = D_{\mathsf{LHE.Dec}} + 1$; moreover, for each layer, an encryption of the secret key $\mathsf{sk}_{\ell-1}$ under the public key pk_ℓ is released, that is, $\Lambda_\ell = (\mathsf{pk}_\ell, \mathsf{evk}_\ell, c_\ell)$ where $(\mathsf{pk}_\ell, \mathsf{sk}_\ell, \mathsf{evk}_\ell) \overset{\$}{\leftarrow} \mathsf{LHE.Keygen}(1^\lambda, 1^{D'})$ for $D' = D_{\mathsf{LHE.Dec}} + 1$ and $c_\ell = \mathsf{LHE.Enc}_{\mathsf{pk}_\ell}(\mathsf{sk}_{\ell-1})$.
Each Λ_ℓ is a layer evaluation key: Given two ciphertexts α, β of bits a and b under $\mathsf{pk}_{\ell-1}$, we can obtain an encryption γ of a NAND b under pk_ℓ, by evaluating homomorphically over c_ℓ the function $f_{\alpha,\beta}(\mathsf{sk}_{\ell-1})$ that decrypts α, β using $\mathsf{sk}_{\ell-1}$ and computes NAND of the decrypted bits. Since $f_{\alpha,\beta}$ has depth exactly $D_{\mathsf{LHE.Dec}}+1$, the homomorphic computation yields a ciphertext γ of a NAND b under pk_ℓ correctly.
Therefore by publishing a super-polynomially number $\mathcal{L}$ of layer evaluation keys $\mathsf{evk} = (\Lambda_1, \cdots \Lambda_{\mathcal{L}})$, the scheme supports homomorphic evaluation of any polynomial depth circuits.

"Compress" the size of the evaluation key: The next step is to "compress" the size of the super-polynomially long evaluation key to obtain a FHE with succinct evaluation key. This step relies on an IO for circuits and a puncturable PRF. The idea is to obfuscate a master circuit Γ that on input

$\ell \in [\mathcal{L}]$ computes the ℓ'th layer evaluation key Λ_ℓ produced using pseudo-random coins generated with a puncturable PRF and hardwired PRF keys k, k'. That is,

$$\begin{aligned}\Lambda_\ell = \Gamma^{(k,k')}(\ell), \quad \text{where } (\mathsf{pk}_\ell, \mathsf{sk}_\ell, \mathsf{evk}_\ell) &= \mathsf{LHE.Keygen}(1^\lambda, 1^{D'}; \mathsf{PRF}(k, \ell)),\\ (\mathsf{pk}_{\ell-1}, \mathsf{sk}_{\ell-1}, \mathsf{evk}_{\ell-1}) &= \mathsf{LHE.Keygen}(1^\lambda, 1^{D'}; \mathsf{PRF}(k, \ell - 1)),\\ c_\ell &= \mathsf{LHE.Enc}_{\mathsf{pk}_\ell}(\mathsf{sk}_{\ell-1}; \mathsf{PRF}(k', \ell))\\ \Lambda_\ell &= (\mathsf{pk}_\ell, \mathsf{evk}_\ell, c_\ell)\end{aligned}$$

Since the size of the master program $\Gamma^{(k,k')}$ is a fixed polynomial in λ, the new evaluation key $\mathsf{evk} \overset{\$}{\leftarrow} i\mathcal{O}(1^s, \Gamma^{k,k'})$ is succinct, of a fixed polynomial size in λ (where s an upperbound on the size of Γ and k, k' are randomly sampled PRF keys). It follows from a careful hybrid argument over the virtual super-polynomial number of levels (similar to that in [11,32]) that the semantic security of LHE remains even when the new evaluation key is additionally released, provided that all primitives from LHE, to $i\mathcal{O}$ to PRF all have a slightly inverse super-polynomially small distinguishing gap $\mu(\lambda) = \mathsf{negl}(\lambda)\mathcal{L}(\lambda)^{-1}$.

Finally, we note that any LHE scheme with decryption in $\mathbf{NC}^1$ have a fixed decryption depth (in particular, the depth is bounded by λ). Many known constructions, for example [23,18,16,15,24] satisfy this property. Thus, if these constructions are slightly super-polynomially secure, by assuming slightly stronger underlying assumptions (for instance the LHE scheme of [18] can be made slightly super-polynomially secure if assuming that the underlying learning with error assumption is slightly super-polynomially secure), they can be directly transformed into a FHE assuming slightly super-polynomially secure IO and OWFs (without assuming circular security).

We also note that our LHE scheme constructed in Section 4.2 has a fixed decryption depth, since its decryption algorithm is identical to that of the underlying trapdoor encryption scheme. It can also be transformed into a FHE using the above general transformation. In the full version [19], we provide a formal description and security proof of the FHE transformed from our LHE.

Acknowledgements Ran Canetti's research is supported by the Check Point Institute for Information Security, ISF grant 1523/14, the NSF MACS Frontier project, and NSF Algorithmic Foundations grant 1218461. Huijia Lin's research is partially supported by a gift from the Gareatis Foundation. Stefano Tessaro's research was partially supported by NSF Grant CNS-1423566 and a gift from the Gareatis Foundation. Vinod Vaikuntanathan's research was supported by DARPA Grant number FA8750-11-2-0225, an Alfred P. Sloan Research Fellowship, an NSF CAREER Award CNS-1350619, NSF Frontier Grant CNS-1414119, a Microsoft Faculty Fellowship, and a Steven and Renee Finn Career Development Chair from MIT.

References

1. Joël Alwen, Manuel Barbosa, Pooya Farshim, Rosario Gennaro, S. Dov Gordon, Stefano Tessaro, and David A. Wilson. On the relationship between functional encryption, obfuscation, and fully homomorphic encryption. In *Cryptography and Coding - 14th IMA International Conference, IMACC 2013, Oxford, UK, December 17-19, 2013. Proceedings*, pages 65–84, 2013.
2. Prabhanjan Ananth, Dan Boneh, Sanjam Garg, Amit Sahai, and Mark Zhandry. Differing-inputs obfuscation and applications. Cryptology ePrint Archive, Report 2013/689, 2013. http://eprint.iacr.org/.
3. Benny Applebaum. Bootstrapping obfuscators via fast pseudorandom functions. *IACR Cryptology ePrint Archive*, 2013:699, 2013.
4. Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. Cryptography in NC^0. In *FOCS*, pages 166–175. IEEE Computer Society, 2004.
5. Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. Computationally private randomizing polynomials and their applications. *Computational Complexity*, 15(2):115–162, 2006.
6. Boaz Barak, Sanjam Garg, Yael Tauman Kalai, Omer Paneth, and Amit Sahai. Protecting obfuscation against algebraic attacks. In Phong Q. Nguyen and Elisabeth Oswald, editors, *EUROCRYPT*, volume 8441 of *Lecture Notes in Computer Science*, pages 221–238. Springer, 2014.
7. 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.
8. Mihir Bellare, Dennis Hofheinz, and Scott Yilek. Possibility and impossibility results for encryption and commitment secure under selective opening. In *EUROCRYPT*, pages 1–35, 2009.
9. Mihir Bellare, Igors Stepanovs, and Stefano Tessaro. Poly-many hardcore bits for any one-way function and a framework for differing-inputs obfuscation. In *ASIACRYPT*, 2014.
10. Nir Bitansky, Ran Canetti, Henry Cohn, Shafi Goldwasser, Yael Tauman Kalai, Omer Paneth, and Alon Rosen. The impossibility of obfuscation with auxiliary input or a universal simulator. In *CRYPTO*, pages 71–89, 2014.
11. Nir Bitansky, Sanjam Garg, and Sidharth Telang. Succinct randomized encodings and their applications. Cryptology ePrint Archive, Report 2014/771, 2014. http://eprint.iacr.org/.
12. Dan Boneh and Brent Waters. Constrained pseudorandom functions and their applications. In Kazue Sako and Palash Sarkar, editors, *ASIACRYPT (2)*, volume 8270 of *Lecture Notes in Computer Science*, pages 280–300. Springer, 2013.
13. Elette Boyle, Kai-Min Chung, and Rafael Pass. On extractability obfuscation. In *TCC*, pages 52–73, 2014.
14. Elette Boyle, Shafi Goldwasser, and Ioana Ivan. Functional signatures and pseudorandom functions. In Hugo Krawczyk, editor, *PKC*, volume 8383 of *Lecture Notes in Computer Science*, pages 501–519. Springer, 2014.
15. Zvika Brakerski. Fully homomorphic encryption without modulus switching from classical gapsvp. In *CRYPTO*, pages 868–886, 2012.
16. Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (leveled) fully homomorphic encryption without bootstrapping. In *ITCS*, pages 309–325, 2012.
17. Zvika Brakerski and Guy N. Rothblum. Obfuscating conjunctions. In Ran Canetti and Juan A. Garay, editors, *CRYPTO (2)*, volume 8043 of *Lecture Notes in Computer Science*, pages 416–434. Springer, 2013.

18. Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. In *FOCS*, pages 97–106, 2011. References are to full version: http://eprint.iacr.org/2011/344.
19. Ran Canetti, Huijia Lin, Stefano Tessaro, and Vinod Vaikuntanathan. Obfuscation of probabilistic circuits and applications. Cryptology ePrint Archive, Report 2014/882, 2014. http://eprint.iacr.org/.
20. Ivan Damgård and Mads Jurik. A generalisation, a simplification and some applications of paillier's probabilistic public-key system. In *PKC*, pages 119–136, 2001.
21. Sanjam Garg, Craig Gentry, Shai Halevi, Amit Sahai, Mariana Raikova, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *FOCS*, 2013.
22. Sanjam Garg, Craig Gentry, Shai Halevi, and Daniel Wichs. On the implausibility of differing-inputs obfuscation and extractable witness encryption with auxiliary input. In *CRYPTO*, pages 518–535, 2014.
23. Craig Gentry. Fully homomorphic encryption using ideal lattices. In *STOC*, pages 169–178, 2009.
24. 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.
25. Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *J. ACM*, 33(4):792–807, 1986.
26. Shafi Goldwasser and Yael Tauman Kalai. On the impossibility of obfuscation with auxiliary input. In *FOCS*, pages 553–562. IEEE Computer Society, 2005.
27. Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *J. Comput. Syst. Sci.*, 28(2):270–299, 1984.
28. Brett Hemenway, Benoît Libert, Rafail Ostrovsky, and Damien Vergnaud. Lossy encryption: Constructions from general assumptions and efficient selective opening chosen ciphertext security. In *ASIACRYPT*, pages 70–88, 2011.
29. Yuval Ishai and Eyal Kushilevitz. Randomizing polynomials: A new representation with applications to round-efficient secure computation. In *FOCS*, pages 294–304. IEEE Computer Society, 2000.
30. Aggelos Kiayias, Stavros Papadopoulos, Nikos Triandopoulos, and Thomas Zacharias. Delegatable pseudorandom functions and applications. In Ahmad-Reza Sadeghi, Virgil D. Gligor, and Moti Yung, editors, *CCS*, pages 669–684. ACM, 2013.
31. Gillat Kol and Moni Naor. Games for exchanging information. In *STOC*, pages 423–432, 2008.
32. Huijia Lin and Rafael Pass. Succinct garbling schemes and applications. Cryptology ePrint Archive, Report 2014/766, 2014. http://eprint.iacr.org/.
33. Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *EUROCRYPT*, pages 223–238, 1999.
34. Chris Peikert, Vinod Vaikuntanathan, and Brent Waters. A framework for efficient and composable oblivious transfer. In *CRYPTO*, pages 554–571, 2008.
35. Chris Peikert and Brent Waters. Lossy trapdoor functions and their applications. *SIAM J. Comput.*, 40(6):1803–1844, 2011.
36. Manoj Prabhakaran and Mike Rosulek. Rerandomizable RCCA encryption. In *CRYPTO*, pages 517–534, 2007.
37. Amit Sahai and Brent Waters. How to use indistinguishability obfuscation: deniable encryption, and more. In David B. Shmoys, editor, *STOC*, pages 475–484. ACM, 2014.