Obfuscation of Probabilistic Circuits and Applications

Ran Canetti* Huijia Lin† Stefano Tessaro‡ Vinod Vaikuntanathan§

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, using different approaches to formalizing the above security requirement, and study non-trivial relations among them. Moreover, we give a construction of one of our pIO variants from sub-exponentially hard indistinguishability obfuscation (for deterministic circuits) and one-way functions, and conjecture this construction to be a good candidate for other pIO variants. We then move on to show a number of applications of pIO:

- We give a general and natural methodology to achieve leveled homomorphic encryption (LHE) from variants of semantically secure encryption schemes and of pIO. In particular, we propose instantiations from lossy and re-randomizable encryption schemes, assuming the two weakest notions of pIO.
- We enhance the above constructions to obtain a full-fledged (i.e., non-leveled) FHE scheme under the same (or slightly stronger) assumptions. In particular, this constitutes the **first construction of full-fledged FHE that does not rely on encryption with circular security.**
- Finally, we show that 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.

*Boston University and Tel Aviv University, canetti@bu.edu
†UC Santa Barbara, rachel.lin@cs.ucsb.edu
‡UC Santa Barbara, tessaro@cs.ucsb.edu
§MIT CSAIL, vinodv@csail.mit.edu

Contents

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) [BGI+12] 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 [BGI+12, GK05, BCC+14].

All this changed with the recent breakthrough results of [GGH+13, SW14]. 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 [BGI+12]. Indeed, following [GGH+13, SW14] 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 it 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. [BGI+12] take a first stab at this challenging task 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.

The basic idea is to compile a randomized circuit, namely a circuit C that expects an input x and a uniformly chosen random input r, returning the random variable $C(x, r)$, into a *deterministic* obfuscated circuit $\Lambda = \mathcal{O}(C)$ that has essentially the same functionality 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 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 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 follow the approach of [BST14, ABG+13] 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). With this framework at hand, 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—they are correspondingly called *dynamic-input pIO, worst-case-input pIO, memory-less worst-case-input pIO, and X-indistinguishable static-input pIO or X-pIO for short* (see definitions in Section 1.1).

A construction for X-pIO. For X-pIO, as our first main result, we prove the existence of a X-pIO scheme, given any IO scheme. 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. Noting that subexponentially-secure puncturable PRFs are implied by subexponentially-hard one-way functions, we obtain the following theorem.

Theorem 1 (Informal) *Assume the existence of a subexponentially-hard indistinguishability obfuscator for (deterministic) circuits and one-way functions. Then, X-*pIO *exist.*

Furthermore, one can also 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 1.2) is to construct 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.** More precisely, we obtain the following theorem:

Theorem 2 (Informal) *Assume the existence of a sub-exponentially secure indistinguishability obfuscator for (deterministic) circuits, and sub-exponentially secure one-way functions.*

- *Any perfectly rerandomizable encryption scheme can be transformed into a* leveled *homomorphic encryption scheme.*
- *Any perfectly rerandomizable encryption scheme that is slightly super-polynomially secure can be transformed into a* fully *homomorphic encryption scheme.*

The above theorem is, in fact, a special case of a *general transformation* that constructs FHE schemes in two steps. In the first step, 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 1.3, 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. The rest of this introduction 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 2 presents our definitions of pIO and studies relations among them. Moreover, it presents the construction of X-pIO from IO and puncturable PRFs. Section 3 and 4 present the application to FHE and bootstrapping IO.

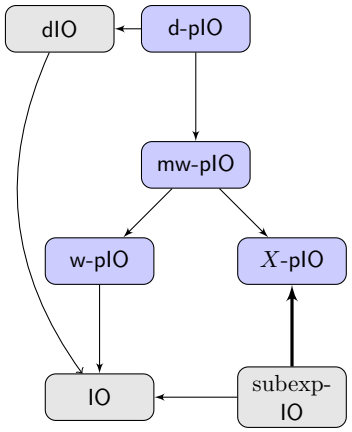

Figure 1: **Notions of obfuscation for probabilistic circuits:** Gray boxes correspond to existing notions, whereas purple ones correspond to notions introduced in this paper. Arrows indicate implication, whereas lack of arrows among azure boxes implies a formal separation. The thicker line indicates that the implication holds under the assumption of subexponentially-hard puncturable PRFs, which in turn, follow from subexponentially-hard one-way functions.

1.1 Our Definitional Framework: IO for Probabilistic Circuits

The first contribution of this paper, found in Section 2, 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 [BST14, ABG+13], 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 main sampler classes matching different approaches to formalizing the above computational indistinguishability requirement on all inputs, resulting in four different notions:

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. It turns out that it makes a difference whether the first, computationally unbounded stage can pass a state (in addition to the chosen input x) to the computationally bounded 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).

X-Ind pIO (X-pIO). The last notion takes a different path. Here, we weaken the adversary to choose the input x *before* seeing (C_0, C_1, z). While this alone enlarges the class too much, resulting in an unachievable notion, we additionally require the guessing advantage of the attacker 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. However, we show that this notion is essentially optimal in the sense that when static input choices are considered, there are specific samplers with distinguishing gaps $1/\mathrm{poly} \cdot X^{-1}$ for which indistinguishability obfuscation is impossible (unconditionally).

In a sequence of steps, 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* [BGI+12, BCP14, ABG+13], and therefore directly suffers from recent implausibility results [GGHW14] 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 [GGH+13, BGK+14, BR13].

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.

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 $y = E_{i-1}^k(i) = D_1^k(i)$ and $y' = E_i^k(i) = D_2^k(i)$ 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 $D_1^k(x)$ and $D_2^k(x)$, respectively, are $\frac{1}{2^{\omega(\log\lambda)}}$- indistinguishable.

We emphasize that the reason that we require the underlying primitives, namely IO and puncturable PRF, to be subexponentially secure is because the above hybrid argument essentially "enumerates" over all inputs in the domain of C_1 and C_2. In fact, with a closer examination, for two specific circuits C_1, C_2, it suffices to "enumerate" over all "differing inputs" for which the distribution of $C_1(x)$ and $C_2(x)$ are different.

1.2 Application 1: Fully-Homomorphic Encryption

The first testbed for our pIO notions, discussed in Section 3, 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 $c = \mathsf{Enc}(\mathsf{pk}_i, a \;\mathsf{NAND}\; b)$. 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. [ABF+13], 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 $\mathsf{d\text{-}piO}$, 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 $\mathsf{w\text{-}piO}$. 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 $\mathrm{poly}(\log X)$ (i.e., $\mathrm{poly}(|\alpha| + |\beta|)$), leading to exponentially growing ciphertext with the depth.

With respect to $\mathsf{w\text{-}piO}$ (and to $\mathsf{mw\text{-}piO}$ 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 [BHY09]. Such schemes can be built using techniques from a variety of works [KN08, PVW08, BHY09, PW11], and admit instantiations from most cryptographic assumptions. This gives an LHE construction from $\mathsf{w\text{-}piO}$ and any lossy encryption schemes.[1]

- **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,

[1] 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.

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 [GM84], Paillier [Pai99], Damgård-Jurik [DJ01], satisfy 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, in Section 3.4.1 we use IO to turn an LHE scheme info an FHE scheme via techniques inspired by the recent works of Bitansky, Garg, and Telang [BGT14], and of Lin and Pass [LP14].

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. [Gen09, BV11, BGV12, Bra12, GSW13]). We believe that this general transformation is of independent interest, especially because it does not rely on any encryption scheme with circular security.

1.3 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. [GGH+13] 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 [App13] 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 [IK00, AIK04] 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 [AIK06], 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 $\mathsf{negl}(\lambda)2^{-|x|}$[2]. 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 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

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

of PRG is needed so that the randomized encoding can be made $\mathsf{negl}(\lambda)/X(\lambda)$ indistinguishable.

2 IO for Probabilistic Circuits

In this section, we extend the notion of indistinguishability obfuscation (IO) to probabilistic circuits. To this end, extending [BST14], we first introduce in Section 2.1 a general notion of IO for general *samplers* outputting triples (C_0, C_1, z), where z is some auxiliary input, and we are going to seek for obfuscation algorithms which, given such a triple from the sampler, have the property that the obfuscations of C_0 and of C_1 are computationally indistinguishable, given z, C_0, and C_1. We then exercise this framework to derive our IO notions for probabilistic circuits in terms of different sampler classes – the smaller the class, the weaker the resulting security assumption. Concretely, we will define four different notions of pIO corresponding to different classes of samplers.

2.1 IO for General Samplers over Probabilistic Circuits

In this section, we define 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. Different notions of obfuscation (with indistinguishability-based security) in the literature, can be derived by instantiating the general definition with a specific class of samplers. For instance, IO for circuits [BGI+12, SW14] is an IO for the class of samplers that sample pairs of deterministic circuits with the same functionality (and some arbitrary auxiliary input). Later, to define various notions of obfuscation for probabilistic circuits, we will instantiate the general definition with classes of samplers that produce pairs of probabilistic circuits satisfying different properties.

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 $\mathrm{poly}(\lambda)$. Extending the notation of [BST14], 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 $z \in \{0,1\}^{\mathrm{poly}(\lambda)}$. 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 2.1 (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 $\mathrm{poly}(|C|, \lambda)$.*

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 above security condition holds with a 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.*

We note that the sub-exponential indistinguishability defined above is weaker than usual sub-exponential hardness assumptions in that the distinguishing gap only need to be small for PPT distinguishers, rather than sub-exponential time distinguishers.

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 $\mathbf{S} \supseteq \mathbf{S}'$.

2.2 Dynamic-input piO for Circuits

We start with the most natural way of formalizing a sampler outputting a triple (C_0, C_1, z) for which C_0 and C_1 are indistinguishable on every input. Namely, we ask indistinguishability on every input x adaptively chosen by a (PPT) adversary $\mathcal{A}_1$ on input (C_0, C_1, z). This defines the largest class of samplers we consider in the following, denoted $\mathbf{S}^{\mathsf{d\text{-}Ind}}$.

Definition 2.2 (*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. $(C_0, C_1, z) \xleftarrow{\$} D_\lambda$
2. $(x, st) \xleftarrow{\$} \mathcal{A}_1(C_0, C_1, z)$
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 $\Pr[b' = b] - 1/2$.

The above definition states that dynamic-input indistinguishable samplers produce pairs of probabilistic circuits satisfying that no efficiency adversary can distinguish their output $C_0(x)$ or $C_1(x)$ on an input chosen adaptively given C_0, C_1, z.

Remark 2.1 (On multi-input indistinguishability)**.** *The above definition of* $\mathbf{S}^{\mathsf{d\text{-}Ind}}$ *only allows the adversary to receive output* $C_b(x)$ *for one input* x*. We note that this is with loss of generality, as it follows from a standard hybrid argument that the above definition implies that the advantage of any efficiency adversary is still negligible even if it receives samples from* $C_b(x)$ *for an unbounded number of* dynamically and adaptively *chosen inputs. More precisely, this means that the advantage of an adversary* $\mathcal{A}^{\mathcal{O}_b}(C_0, C_1, z)$ *with access to an oracle* $\mathcal{O}_b$ *in guessing* b *is negligible, where* $\mathcal{O}_b$ *on input* x *samples* $y \xleftarrow{\$} C_b(x)$ *for* $\mathcal{A}$*. For simplicity, we therefore adopt the above definition, as a multi-input indistinguishability requirement does not restrict the class any further.*

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

Definition 2.3 (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. We can recover the notion of differing-inputs indistinguishability obfuscation (dIO) for circuits [BGI+12, BCP14, ABG+13], by just restricting the above definition of d-piO to the class $\mathcal{C}' = \{\mathcal{C}'_\lambda\}_{\lambda\in\mathbb{N}}$ of *deterministic* circuits. In this case, if a circuit sampler $D = \{D_\lambda\}_{\lambda\in\mathbb{N}}$ is in the class $\mathbf{S}^{\mathsf{d\text{-}Ind}}$, then it must be that given a randomly sampled tuple $(C_0, C_1, z) \xleftarrow{\$} D_\lambda$, it is hard to find an input x that makes the two deterministic circuits output different values. Therefore, we can re-define the dIO notion as follows within our framework .

Definition 2.4 (Differing-input IO for Circuits [BGI+12, BCP14, ABG+13])**.** *A uniform PPT machine di*$\mathcal{O}$ *is a* differing input IO *for 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* deterministic *circuits of size at most* λ*.*

In other words, the notion of dynamic-input piO is a direct generalization of differing-input IO to the case of randomized circuits. In a recent work by Garg et al. [GGHW14], 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.

2.3 Worst-case-input piO for Circuits

In light of the implausibility of general dynamic-input piO, we seek for a more restrictive class of samplers which bypasses this result. To this end, we introduce the notion of worst-case-input indistinguishable samplers and denote the resulting class as $\mathbf{S}^{\mathsf{w\text{-}Ind}}$.

Definition 2.5 (*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, summarized in the following definition.

Definition 2.6 (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 $\mathcal{A}_2$.

$$(x^*, st^*) = \arg\max_{x,st \in \{0,1\}^{\mathrm{poly}(\lambda)}} \left(\Pr[b \xleftarrow{\$} \{0,1\}, y \xleftarrow{\$} C_b(x) \ : \ b = \mathcal{A}_2(st, C_0, C_1, z, x, y))] \right)$$

Since the above definition quantifies over all adversaries $(\mathcal{A}_1, \mathcal{A}_2)$, it implies that worst-case-input indistinguishable samplers produce 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.

Remark 2.2 (On Multi-input indistinguishability)**.** *As in Definition 2.3, the above definition only allows the adversary to choose one input x. We note that this implies a limited form of multi-instance indistinguishability: 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 $y_i \xleftarrow{\$} C_b(x_i)$ for all these inputs, i.e., it is given $(st, C_0, C_1, z, \{x_i\}, \{y_i\})$. This form of multi-instance indistinguishability follows from the above definition by a standard hybrid argument, but note that the ability for $\mathcal{A}_1$ to pass on state to $\mathcal{A}_2$ is clearly necessary for this proof to be valid.*

Also note that we cannot consider the stronger form of multi-instance indistinguishability considered for dynamic-input indistinguishable samplers; this is because here $\mathcal{A}_1$ and $\mathcal{A}_2$ have different computational powers and cannot be "collapsed" into one adversary.

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 $st = \bot$, 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 2.7 (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. We can recover the notion of indistinguishability obfuscation (IO) for circuits [BGI+12, GGH+13] by restricting Definition 2.6 of worst-case-input piO to the class $\mathcal{C}' = \{\mathcal{C}'_\lambda\}_{\lambda\in\mathbb{N}}$ of *deterministic* circuits. In this case, if a circuit sampler $D = \{D_\lambda\}_{\lambda\in\mathbb{N}}$ is in the class $\mathbf{S}^{\mathsf{w\text{-}Ind}}$, then it must be that given a randomly sampled tuple $(C_0, C_1, z) \xleftarrow{\$} D_\lambda$, C_0 and C_1 agrees on every input x. Thus, the definition of IO can be restated in our framework as follows.

Definition 2.8 (IO for Circuits [BGI+12, GGH+13])**.** *A uniform PPT machine* $i\mathcal{O}$ *is a* IO *for 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* deterministic *circuits of size at most* λ*.*

In other words, the notion of worst-case-input piO is a direct generalization of IO to the case of randomized 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.

Relations. It follows directly from the definitions that every worst-case-input indistinguishable sampler is also a dynamic-input indistinguishable sampler. This is true also if the sampler is only memory-less worst-case-input indistinguishable sampler. Therefore:

Proposition 2.1 (d-piO $\Rightarrow$ mw-piO $\Rightarrow$ w-piO)**.** *A dynamic-input* piO *obfuscator for randomized circuits is also a memory-less worst-case-input* piO *obfuscator for randomized circuits. Moreover, a memory-less worst-case-input* piO *obfuscator is also a worst-case-input* piO *obfuscator.*

Proof. It is not hard to see that $\mathbf{S}^{\mathsf{w\text{-}Ind}} \subseteq \mathbf{S}^{\mathsf{d\text{-}Ind}}$ and obviously $\mathbf{S}^{\mathsf{w\text{-}Ind}} \subseteq \mathbf{S}^{\mathsf{mw\text{-}Ind}}$. It is a bit harder to prove that $\mathbf{S}^{\mathsf{mw\text{-}Ind}} \subseteq \mathbf{S}^{\mathsf{d\text{-}Ind}}$. Indeed, assume for a sampler $D \in \mathbf{S}^{\mathsf{w\text{-}Ind}}$ there exists a non-uniform PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ with non-negligible advantage in Experiment dynamic-input-$\mathsf{IND}^D_{\mathcal{A}}(1^\lambda)$. Then, we can assume without loss of generality that $\mathcal{A}$ is deterministic by non-uniformity. Moreover, we can build an memory-less worst-case-input adversary $\mathcal{A}' = (\mathcal{A}'_1, \mathcal{A}'_2)$ where $\mathcal{A}'_1(C_0, C_1, z)$ runs $\mathcal{A}_1(C_0, C_1, z)$ to compute the input x (and ignores the state st output by $\mathcal{A}_1$), whereas $\mathcal{A}'_2$, given $(C_0, C_1, x, y = C_b(x), z)$, first re-runs $\mathcal{A}_1(C_0, C_1, z)$ to obtain (x, st), and

then runs $\mathcal{A}_2(st, C_0, C_1, x, y, z)$ to compute the output bit. Clearly $\mathcal{A}'$ achieves the same advantage as $\mathcal{A}$, which is non-negligible, leading to a contradiction. □

A plausible candidate. We stress that existing implausibility results do not apply to this class (as much as IO for deterministic circuits appears feasible). We conjecture here that for a suitable PRF construction, obfuscating a randomized circuit C by applying one of the existing IO obfuscators [GGH+13, BR13, BGK+14] to the derandomized circuit $D^K(x) = C(x; \mathsf{PRF}_K(x))$ for some hard-coded PRF seed K, is indeed a secure $\mathsf{w\text{-}pIO}$ obfuscator.

Conjecture 1. *There exists a* $\mathsf{w\text{-}pIO}$ *obfuscator for randomized circuits.*

Below, we will prove that the above construction is indeed a secure obfuscator for an alternative, weak class of indistinguishable samplers, which we move to introduce next.

2.4 Static-input pIO for Circuits

Given the above formulations of dynamic-input pIO and worst-case-input pIO, it is natural to ask whether there exists a notion of pIO corresponding to static input selection. In this section, we formulate the class of static-input indistinguishable samplers and define the notion of static-input pIO for circuit. We show that general static-input pIO is impossible. However, pIO for a more restricted class of static-input indistinguishable samplers, namely these that are "X-strongly indistinguishable" (defined shortly below), is possible; in fact, in Section 2.6 we give a construction for such a pIO assuming sub-exponentially indistinguishable IO.

Definition 2.9 (*Static-input* Indistinguishable Samplers)**.** *The class* $\mathbf{S}^{\mathsf{s\text{-}Ind}}$ *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 *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* $\Pr[b' = b] - 1/2$.

In the above definition, the adversary chooses the challenge input x statically before seeing the circuit pairs and auxiliary input sampled by the sampler. This definition is equivalent to stating that for every input x, no efficient adversary can distinguish the outputs of two randomly sampled circuits from a static-input indistinguishable sampler D on input x.

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 2.2. *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.

D^* is a static-input indistinguishable sampler. 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.

On the other hand, we show that any machine $pi\mathcal{O}$ that has correctness cannot be secure for this sampler D^*. This is because 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. As mentioned above, we now consider a smaller class of static-input indistinguishable samplers, $\mathbf{S}^{X\text{-}\mathsf{Ind}} \subset \mathbf{S}^{\mathsf{s\text{-}Ind}}$, that circumvents the impossibility of Proposition 2.2. 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 2.10 ((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 2.9 is $\mathsf{negl}X^{-1}$.*

Definition 2.11 (X-Ind piO for Randomized Circuits)**.** *Let X be any function bounded by 2^λ. A uniform PPT machine X-$pi\mathcal{O}$ 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 2.2 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 circuit 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: For every function X, we provide a construction of X-piO for randomized circuits in Section 2.6, assuming sub-exponentially secure IO for deterministic circuits, which we believe to be a reasonable assumption.

Other X-Ind notions. One could indeed formulate X-Ind versions of dynamic-input, memory-less worst-case-input, and worst-case-input piO, where the distinguishing advantage is required to be very small. Indeed, these notions would all be (in analogy of the above) *weaker* (and hence implied) by X-piO, and can thus all be achieved by our construction presented below. (Note that our possibility result below does not contradict the implausibility of [GGHW14], as the corresponding sampler from their proof would enable a larger distinguishing gap.)

Relation to worst-case-input piO. We are going to now show that every X-Ind indistinguishable sampler is also a memory-less worst-case-input indistinguishable one, i.e., it belongs to $\mathbf{S}^{\mathsf{mw\text{-}Ind}}$. Hence, if memory-less worst-case-input piO exists, then so does X-Ind piO. This result appears at first somewhat surprising (and indeed, its proof requires some technical work), because static adversaries are the weakest, as opposed to memory-less worst-case-input adversaries being among the strongest. This really highlights the power of requiring a $\mathsf{negl}X^{-1}$ distinguishing gap. Moreover, it is important to note that the result *does not* extend to worst-case-input piO, which we show to be incomparable to X-Ind piO below.

Proposition 2.3. *Let $X = X(\lambda) \leq 2^{\mathrm{poly}(\lambda)}$. Then, $\mathbf{S}^{X\text{-}\mathsf{Ind}} \subseteq \mathbf{S}^{\mathsf{mw\text{-}Ind}}$.*

The proof of the proposition is provided in Appendix B.1.

Corollary 2.4. *If* d-piO *or* mw-piO *for randomized circuits exists, then X-piO for randomized circuits exists.*

2.5 Strict Inclusions

We note that all relations we proved above result into strict inclusions of the corresponding sampler classes into each other. Moreover, we also note that the two notions of w-piO and X-piO are incomparable. This follows by the three following separations of sampler classes:

mw-piO $\not\Rightarrow$ d-piO. Let π_λ be a permutation on $\{0,1\}^\lambda$. Consider the sampler D such that D_λ outputs triples (C_0, C_1, z) such that $C_0(x) = 0$ for all inputs $x \in \{0,1\}^\lambda$, $z = \pi_\lambda(x^*)$ for a random x^* in $\{0,1\}^\lambda$, and finally, C_1, on input x, checks whether $\pi_\lambda(x) = z$, and if so outputs 1, and it outputs 0 otherwise. Then, it is not hard to see that $D \in \mathbf{S}^{\mathsf{d\text{-}Ind}}$ as long as $\pi = \{\pi_\lambda\}_{\lambda\in\mathbb{N}}$ is one-way, yet $D \notin \mathbf{S}^{\mathsf{mw\text{-}Ind}}$, as a computationally unbounded $\mathcal{A}_1$ can simply recover x^* from z, and output this value, and this leads to easy distinguishing for $\mathcal{A}_2$.

X-piO $\not\Rightarrow$ w-piO. We consider a bit-commitment scheme which is computationally hiding (against non-uniform adversaries) and statistically binding. Then, we consider the pair of circuits C_0, C_1 with input length λ such that $C_b(x)$ ignores the λ-bit inout x and outputs $\mathsf{Com}(b; r)$, for a randomly chosen commitment randomness which we set to be λ bits without loss of generality. Note that these circuits differ on every input. Then, consider D which outputs (C_0, C_1). First of all, we have $D \in \mathbf{S}^{\mathsf{w\text{-}Ind}}$, because regardless *for all* x, the outputs $C_0(x)$ and $C_1(x)$ are computationally indistinguishable. However, $D \notin \mathbf{S}^{X\text{-}\mathsf{Ind}}$ for $X = 2^\lambda$: A distinguisher can just guess the randomness r, and check which bit was committed to (and if the randomness is incorrect, just output a random bit). This strategy achieves advantage $2^{-\lambda} = 1/X$.

w-piO $\not\Rightarrow$ X-piO. Here, we consider a sampler which outputs a triple (C_0, C_1, z), where $z = \mathsf{pk}$ is the public key of a bit-encryption scheme, and $C_b(x) = \mathsf{Enc}(\mathsf{pk}, b)$ for all x. (Note that no

matter how one defines the differing sets, all x are in it, since the circuits behave completely differently on all inputs.) Here, we assume that x is λ-bit long, and that the encryption scheme can be distinguished with advantage at most $\mathsf{negl}(\lambda)\cdot 2^{-\lambda}$ by an attacker. (Such a scheme can be obtained from any subexponentially-secure bit-encryption scheme.) Then, clearly, $D \in \mathbf{S}^{X\text{-}\mathsf{Ind}}$, but however, $D \notin \mathbf{S}^{\mathsf{w}\text{-}\mathsf{Ind}}$, since an unbounded $\mathcal{A}_1$ can compute a secret key from z and pass it on to $\mathcal{A}_2$.

We stress that the last counterexample does not work against mw-pIO (thus possibly violating Proposition 2.3) because in order to achieve such high security, the secret key of the exponentially-secure encryption scheme must be *longer* than λ bits, and thus cannot be embedded in a carefully chosen x.

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

The following theorem yields a construction of an X-Ind pIO obfuscator (as in Definition 2.11) from sub-exponentially hard IO. It relies on sub-exponentially secure puncturable PRF, as defined in Appendix A.1.

C, C_1 and C_2 are probabilistic circuits of size at most λ, K is a key of the PRF function and K_{-i} a punctured key at input x_i, and y is a string of length at most λ. In particular, $x_1, \ldots, x_X$ for $X = X(\lambda)$ are the elements of the differing domain, canonically ordered (e.g., with respect to the lexicographic ordering of strings).

Circuit $E^{(C,K)}(x)$**:** Output $C(x\ ;\ \mathsf{PRF}(K,x))$.

Circuit $E_i^{(C_1,C_2,K)}(x)$**:** If $x < x_i$, output $C_1(x\ ;\ \mathsf{PRF}(K,x))$; otherwise, if $x \geq x_i$, output $C_2(x\ ;\ \mathsf{PRF}(K,x))$.

Circuit $E_i^{(C_1,C_2,K,y)}(x)$**:** If $x < x_i$, output $C_1(x\ ;\ \mathsf{PRF}(K_{-i},x))$, and if $x = x_i$, output y; otherwise, if $x > x_i$, output $C_2(x\ ;\ \mathsf{PRF}(K_{-i},x))$.

All circuits $E^{(C,K)}, E_i^{(C_1,C_2,K)}, E_i^{(C_1,C_2,K,y)}$ are padded to their maximum size.

Figure 2: Circuits used in the construction of X-*pi*$\mathcal{O}$ and its analysis

Theorem 2.5 (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* $(\mathrm{Key}, \mathrm{Puncture}, \mathsf{PRF})$*. Then, there exists a X-Ind pIO obfuscator X-pi$\mathcal{O}$ for randomized circuits.*

Below we describe only our construction of X-Ind pIO, denoted as X-*pi*$\mathcal{O}$, and leave the proof of its correctness and security to Appendix B.2. Recall that by our assumption, both $i\mathcal{O}$ and the puncturable PRF $(\mathrm{Key}, \mathrm{Puncture}, \mathsf{PRF})$ 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 $K \leftarrow \text{Key}(1^{\lambda'})$.
2. Construct deterministic circuit $E^{(C,K)}$ as described in Figure 2. 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 $i\mathcal{O}$, $\Lambda \xleftarrow{\$} i\mathcal{O}(1^{\lambda''}, E^{(C,K)})$.
4. Output Λ.

3 Application 1: Fully Homomorphic Encryption

3.1 Our Results

In this section, we show how to construct leveled and fully homomorphic encryption schemes using different notions of pIO. One of the main corollary of our construction is that assuming sub-exponentially indistinguishable IO and sub-exponential secure OWF, any rerandomizable CPA encryption scheme (that is slightly inverse super-polynomial indistinguishable) can be transformed into a FHE scheme. More precisely,

Theorem 3.1 (LHE and FHE from sub-exp indistinguishable IO and sub-exp OWFs)**.** *Assume the existence of a sub-exponentially indistinguishable IO for circuits, and a sub-exponentially secure OWF.*

- *Any perfectly rerandomizable encryption scheme can be transformed into a* leveled *homomorphic encryption scheme.*
- *Any perfectly rerandomizable encryption scheme that is μ-indistinguishable for any inverse super-polynomial function μ, can be transformed into a* fully *homomorphic encryption scheme.*

At a high-level, the above theorem is obtained through a general two-step approach.

Step 1: A general construction of LHE. We first show a *general transformation* that turns any, so called, trapdoor encryption scheme Π (introduced shortly), into a LHE scheme, assuming the existence of pIO for a specific class of samplers defined by Π. Our transformation follows a very natural idea for enabling homomorphic evaluation: Obfuscate the program that on input some input ciphertexts, decrypt them, compute NAND and re-encrypt under a different key. Since the program to be obfuscated is probabilistic, pIO comes into play natural. But as seen in Section 2, pIO for different classes of samplers leads to various notions with different power. The challenge is to only rely on weak notions of pIO, namely, worst-case-input pIO and X-Ind pIO. We achieve this by instantiating the trapdoor encryption scheme Π with specific constructions that have certain special properties.

Step 2: A general transformation from LHE to FHE. Assume that the LHE scheme constructed in Step 1 is slightly inverse super-polynomially indistinguishable (i.e., the advantage of any PPT adversary in violating semantic security is bounded by an inverse super-polynomial function μ, for example, $2^{\log \lambda \log\log \lambda}$); this can be achieved assuming slightly stronger

underlying trapdoor encryption scheme and piO. Then additionally assuming an inverse super-polynomially indistinguishable IO for deterministic circuits and puncturable PRF, we can transform the LHE scheme into a FHE scheme without relying on circular security. In fact, this transformation is general and applies to not only our LHE scheme based on piO, but any LHE scheme that has a fixed decryption depth, which includes almost all known LHE schemes (e.g. [Gen09, BV11, BGV12, Bra12, GSW13]).

Below we describe the two steps in more details.

Roughly speaking, a trapdoor encryption scheme has two-modes: In the honest mode, it acts as a CPA secure encryption scheme, whereas in the trapdoor mode, an *indistinguishable* trapdoor public key is sampled, under which encryption is no longer guaranteed to have correctness. So far, we haven't put any security requirement on the trapdoor mode. In this work, we will consider three different security properties in the trapdoor mode: (i) Computationally hiding, that is, encryption under trapdoor public keys are negligibly indistinguishable, or (ii) ν-hiding, that is, encryption is indistinguishable with a distinguishing gap $\nu(\lambda) = \mathsf{negl}(\lambda)2^{2l(\lambda)}$ where l is the length of the ciphertext, or (iii) statistically hiding.

The advantage of using a trapdoor encryption scheme is that first it is a generalization of CPA encryption (as any CPA encryption automatically gives a trapdoor encryption scheme with computationally hiding trapdoor mode), and second the trapdoor mode may have strong hiding properties, such as ν-hiding and statistically-hiding, that are impossible for normal CPA encryption schemes. The strong hiding properties are, in fact, the key towards basing our LHE scheme on weak notions of piO, in particular, the statistically hiding property of the trapdoor mode enables our LHE scheme to rely only on worst-case-input piO, while ν-hiding enables it to rely only on X-Ind piO, leading to different instantiations of our general construction of LHE.

Summarizing, our general transformation is stated informally in the following proposition (see Proposition 3.2 for the formal statement), followed by its instantiations.

Informal Proposition (Step 1: General Construction of LHE). *Let* Π *be a trapdoor encryption scheme. Assume the existence of a* piO *for an appropriate class of samplers* $\mathbf{S}^{\Pi}$ *defined by* Π*. Then,* Π *can be transformed into a* leveled *homomorphic encryption scheme.*

By instantiating the two key components—the trapdoor encryption scheme and a matching piO—we obtain the following three instantiations:

1. **Rerandomizable encryption + sub-exponentially indistinguishable IO:** We show that perfectly rerandomizable encryption scheme can be converted into a trapdoor encryption scheme whose trapdoor mode is ν-hiding. (In fact, as we see later, a weaker notion of rerandomizability suffices.) The ν-hiding property of the trapdoor mode implies that the matching piO can be instantiated with X-Ind piO, which in turn is implied by sub-exponentially indistinguishable IO and OWF by our construction in Section 2.6. This instantiation corresponds to the first LHE construction in Theorem 3.1.

2. **Lossy encryption + worst-case-input piO:** In [BHY09] Bellare, Hofheinz and Yilek introduced lossy encryption, based on dual-mode encryption in [PVW08] and meaningful/meaningless encryption in [KN08]. As we see below, a lossy encryption is exactly a trapdoor encryption scheme with statistically hiding trapdoor mode, which implies that the matching piO can be instantiated with worst-case-input piO. Recall that, so far, no implausibility results apply to worst-case-input piO and a candidate construction is described in Conjecture 1.

3. **CPA encryption + dynamic-input piO:** Any CPA encryption scheme trivially implies a trapdoor encryption scheme whose trapdoor mode is computationally hiding (by simply setting the trapdoor mode to be identical to the honest mode). The computational hiding property in the trapdoor mode implies that the matching piO can be instantiated using dynamic-input piO. Though dynamic-input piO for general classes of samplers is implausible [GGHW14], dynamic-input piO for the *specific* class defined by a CPA encryption for our LHE scheme circumvents the implausibility result. Furthermore, the same construction in Conjecture 1 can also be a candidate of dynamic-input piO for that specific class.

In the next step, we show a general transformation that turns any LHE scheme with a fixed decryption depth (i.e., the decryption algorithm has depth $\mathrm{poly}(\lambda)$ independent of the maximum evaluation level) into a FHE, provided that the encryption of the LHE scheme is slightly inverse super-polynomially indistinguishable and assuming IO and puncturable PRFs with also inverse super-polynomial distinguishing gap.

Informal Proposition (Step 2: General transformation from LHE to FHE). *Let μ be any inverse super-polynomial function. Assume a μ-indistinguishable IO for deterministic circuits and puncturable PRF. Then any μ-indistinguishable LHE scheme Π, can be transformed into a FHE scheme.*

The general transformation from LHE to FHE directly applies to our LHE scheme which has a decryption algorithm identical to the underlying trapdoor encryption scheme, (hence, the decryption depth is independent of the maximum evaluation depth). When the LHE scheme is instantiated with a rerandomizable encryption and sub-exponentially indistinguishable IO, the transformation leads to an FHE without assuming any additional assumptions, which corresponds to the construction of FHE in Theorem 3.1. Furthermore, the general transformation also applies to many known lattice based LHE schemes such as [Gen09, BV11, BGV12, Bra12, GSW13]; in particular, this give FHE based on slightly super-polynomial hardness of the learning with error problem, and slightly super-polynomially secure IO and OWFs.

3.1.1 Organization of the Section.

Below, we first formally define trapdoor encryption scheme in Section 3.2. Then we proceed to formally describe the general transformation from a trapdoor encryption scheme to a LHE scheme, and discuss about the three concrete instantiations in Section 3.3. In Sections 3.4, we show how to further turn the construction of LHE into a FHE.

3.2 Trapdoor Encryption Schemes

In this section, we define the notion of trapdoor encryption schemes. They 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 3.1 (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,

Claim 3.1.1. *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.

3.2.1 Statistical Trapdoor Encryption Scheme

Definition 3.2 (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 3.1 is replaced by the following.*

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

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

We note that any lossy encryption scheme as defined by Bellare, Hofheinz and Yilek [BHY09] 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:

Claim 3.1.2. *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.*

3.2.2 μ-Hiding Trapdoor Encryption Scheme

Definition 3.3 (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 3.1 is replaced by the following.*

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

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

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. Note that such a small distinguishing gap can only be achieved in the trapdoor mode where correctness is not required. 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, $\mu = 0$.

Below we define the notion of rerandomizable encryption scheme we need and provide the construction. Roughly speaking, a μ-rerandomizable encryption scheme is one such that given freshly sampled public key pk and two encryptions c_0, c_1 of the same bit b, the distribution of a rerandomized ciphertext from c_0 and that from c_1 are μ-indistinguishable. We note that our notion of rerandomizable encryption is slightly different from previous notions in the literature, for example [PR07, HLOV11], where re-randomizability means that the distribution of a rerandomized ciphertext is statistically close or identical to the distribution of a freshly generated ciphertext. Here, we do not require the rerandomized ciphertext to distribute close to a fresh ciphertext, but rather, require the weaker property that their distributions are μ-indistinguishable no matter which original ciphertexts they are derived from. Many encryption scheme such as ElGamal, Goldwasser-Micali [GM84], Paillier [Pai99], Damgård-Jurik [DJ01], are in fact perfectly rerandomizable as per [PR07, HLOV11] and hence satisfy our definition.

Definition 3.4 (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 $\lambda \in \mathbb{N}$.*

$$\Big|\Pr[(\mathsf{pk},\mathsf{sk}) \xleftarrow{\$} \mathsf{Gen}(1^\lambda),\ c_0 \xleftarrow{\$} \mathsf{Enc}_{\mathsf{pk}}(b),\ c_1 \xleftarrow{\$} \mathsf{Enc}_{\mathsf{pk}}(b)\ :\ \mathcal{A}(\mathsf{pk}, c_0, c_1, \mathsf{reRand}_{\mathsf{pk}}(c_0)) = 1]$$
$$-\Pr[(\mathsf{pk},\mathsf{sk}) \xleftarrow{\$} \mathsf{Gen}(1^\lambda),\ c_0 \xleftarrow{\$} \mathsf{Enc}_{\mathsf{pk}}(b),\ c_1 \xleftarrow{\$} \mathsf{Enc}_{\mathsf{pk}}(b)\ :\ \mathcal{A}(\mathsf{pk}, c_0, c_1, \mathsf{reRand}_{\mathsf{pk}}(c_1)) = 1]\Big| \le \mu(\lambda)$$

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

Claim 3.1.3. *Let μ be any negligible function. Every μ-rerandomizable CPA encryption scheme Π' can be transformed into a μ-hiding trapdoor encryption scheme Π.*

To construct a μ-hiding trapdoor encryption scheme Π from a μ-rerandomizable encryption Π', the idea is to generate an (honest) public key that contains a public key pk of Π', and a pair of ciphertexts (c_0, c_1) of bits 0 and 1 under pk; to encrypt a bit b, simply rerandomize the b'th ciphertext c_b in the public key. On the other hand, a trapdoor public key contains instead two ciphertexts c'_0, c'_1 of 0. It follows from the μ-rerandomizability Π' that rerandomized ciphertexts from c'_0 and c'_1 are μ-indistinguishable; therefore, under the trapdoor public key, ciphertexts of 0 and 1 are μ-indistinguishable. Now we provide the complete proof.

Proof of Claim 3.1.3. Fix any μ-rerandomizable CPA encryption scheme $\Pi' = (\mathsf{KeyGen}', \mathsf{Enc}', \mathsf{Dec}', \mathsf{reRand}')$. We show how to transform it into a μ-hiding trapdoor encryption scheme $\Pi = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{tKeyGen})$.

- $\mathsf{KeyGen}(1^\lambda)$: Sample a pair of keys $(\mathsf{pk}', \mathsf{sk}') \xleftarrow{\$} \mathsf{KeyGen}'(1^\lambda)$ of Π', and two ciphertexts of 0 and 1, $c_0 \xleftarrow{\$} \mathsf{Enc}'_{\mathsf{pk}'}(0)$ and $c_1 \xleftarrow{\$} \mathsf{Enc}'_{\mathsf{pk}'}(1)$. Output $\mathsf{pk} = (\mathsf{pk}', c_0, c_1)$ and $\mathsf{sk} = \mathsf{sk}'$.
- $\mathsf{Enc}_{\mathsf{pk}}(b)$: To encrypt a bit $b \in \{0,1\}$, rerandomize the b'th ciphertext c_b in the public key pk, to obtain the ciphertext $c \xleftarrow{\$} \mathsf{reRand}'_{\mathsf{pk}'}(c_b)$.
- $\mathsf{Dec}_{\mathsf{sk}}(c)$: Decrypt using Dec' with secret key $\mathsf{sk} = \mathsf{sk}'$, $b = \mathsf{Dec}'_{\mathsf{sk}'}(c)$.
- $\mathsf{tKeyGen}(1^\lambda)$: Sample a pair of keys $(\mathsf{pk}', \mathsf{sk}') \xleftarrow{\$} \mathsf{KeyGen}'(1^\lambda)$ of Π', and two ciphertexts of 0, $c_0 \xleftarrow{\$} \mathsf{Enc}'_{\mathsf{pk}'}(0)$ and $c_1 \xleftarrow{\$} \mathsf{Enc}'_{\mathsf{pk}'}(0)$. Output $\mathsf{tpk} = (\mathsf{pk}', c_0, c_1)$.

It follows from the μ-rerandomizability and the semantic security of Π' that Π is also semantically secure. Moreover, it follows from the semantic security of Π that a trapdoor public key (consisting of pk' and two ciphertexts of 0) is indistinguishable from an honest public key (consisting of pk', one ciphertext of 0 and one of 1). Finally, it follows again from the μ-rerandomizability of Π' that under a trapdoor public key, distributions of encryption of 0 or 1 are μ-indistinguishable. This concludes that Π is a μ-hiding trapdoor encryption scheme. □

3.3 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 4; (more explanation on the class is provided in the proof of semantic security).

Proposition 3.2. *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 4. 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 3.3 and 3.4. 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 $(\mathsf{pk}_i, \mathsf{sk}_i) \xleftarrow{\$} \mathsf{KeyGen}(1^\lambda)$ of Π;
  - for $i \geq 1$, obfuscate the circuit $P_i = \mathsf{Prog}^{(\mathsf{sk}_{i-1}, \mathsf{pk}_i)}$ as described in Figure 3, that is, sample $\Lambda_i \xleftarrow{\$} pi\mathcal{O}(1^s, P_i)$ where the security parameter $s = s(\lambda)$ for obfuscation is an upper-bound on the size of all P_i's. [3]

  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, $c \xleftarrow{\$} \mathsf{Enc}_{\mathsf{pk}_0}(m)$.

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

- **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 $\gamma \xleftarrow{\$} \mathsf{Enc}_{\mathsf{pk}}(a \text{ NAND } b)$.

Circuit $\mathsf{tProg}^{(\mathsf{tpk})}(\alpha, \beta)$**:** Output $\gamma \xleftarrow{\$} \mathsf{Enc}_{\mathsf{tpk}}(0)$.

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

Figure 3: 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. That is,

Lemma 3.3. *If* pIO *and* Π *are correct, then* LHE *has homomorphism.*

Proof. Fix any sequence of circuits C_λ of depth at most $L(\lambda)$ and inputs $m_1, \ldots, m_\ell \in \{0,1\}$ (where $\ell = \ell(\lambda)$). We want to show that

$$\Pr\left[\mathsf{LHE.Dec}_{\mathsf{sk}}(\mathsf{LHE.Eval}_{\mathsf{evk}}(C_\lambda, c_1, \ldots, c_\ell)) \neq C_\lambda(m_1, \ldots, m_\ell)\right] = \mathsf{negl}(\lambda) \ ,$$

where $(\mathsf{pk}, \mathsf{evk}, \mathsf{sk}) \leftarrow \mathsf{LHE.Keygen}(1^\lambda)$ and $c_i \leftarrow \mathsf{LHE.Enc}_{\mathsf{pk}}(m_i)$.

Assume for contradiction that the above condition does hold, that is the probability above is at least $1/p(\lambda)$ for some polynomial. Towards reaching a contradiction, we consider a sequence of hybrids, $H_0, \cdots, H_L$: In H_i, the $\mathsf{LHE.Eval}_{\mathsf{evk}}(C_\lambda)$ procedure is modified to $\mathsf{LHE.Eval}^i_{\mathsf{evk}}(C_\lambda)$ as follow: The first i layers of C_λ are homomorphically evaluated using the obfuscated circuits Λ_j, and the rest layers are evaluated using the circuits P_j directly (without obfuscation). In H_0, the whole circuit C_λ is evaluated using solely $\{P_j\}$, whereas in H_ℓ, C_λ is evaluated using solely $\{\Lambda_j\}$ as in $\mathsf{LHE.Eval}$. Define p_i to be the probability that $\mathsf{LHE.Eval}^i$ produces the wrong output, that is,

$$p_i = \Pr\left[\mathsf{LHE.Dec}_{\mathsf{sk}}(\mathsf{LHE.Eval}^i_{\mathsf{evk}}(C_\lambda, c_1, \ldots, c_\ell)) \neq C_\lambda(m_1, \ldots, m_\ell)\right]$$

It follows directly from the correctness of the encryption scheme Π that, this probability p_0 in H_0 is negligible. Thus there must exist an $0 \leq i < L$, such that $|p_i - p_{i+1}| \geq 1/2Lp(\lambda)$. Moreover, there must exist a set of keys $\{pk_i, sk_i\}$, such that conditioned on $\mathsf{LHE.Keygen}$ sampling these keys in H_i and H_{i+1}, it still holds that $|p_i - p_{i+1}| \geq 1/2Lp(\lambda)$. Recall that H_i and H_{i+1} only differ at

whether layer $i+1$ is evaluated using P_{i+1} or $\Lambda_{i+1} \xleftarrow{\$} pi\mathcal{O}(1^s, P_{i+1})$. We show how to construct an distinguisher $\mathcal{D}$ that violates the correctness of the pIO obfuscator $pi\mathcal{O}$.

The distinguisher $\mathcal{D}$ on input $(1^s, P_{i+1}, z)$, with $z = (\{pk_i, sk_i\}, \{m_i\}, C_\lambda)$, participates externally either experiment $\mathsf{Exp}^1_{\mathcal{D}}(1^s, P_{i+1}, z)$ or $\mathsf{Exp}^2_{\mathcal{D}}(1^s, P_{i+1}, z)$ (where it receives outputs of P_{i+1} in the former and that of Λ_{i+1} in the latter). Internally, $\mathcal{D}$ tries to homomorphically evaluate $C_\lambda(m_1, \cdots, m_\ell)$ as in H_i or H_{i+1} as follows. It encrypts $m_1, \cdots, m_\ell$ honestly obtaining ciphertexts $c_1, \cdots, c_\ell$, and obfuscates P_1 to P_i using $pi\mathcal{O}$ obtaining Λ_1 to Λ_i. Then it evaluates the first i layers of C_λ over $c_1, \cdots, c_\ell$ homomorphically using Λ_1 to Λ_i, and layer $i+2$ to L using P_{i+2} to P_L; to evaluate layer $i+1$, for each NAND gate g, it forwards externally the encryptions $\alpha(g), \beta(g)$ of the values of g's input wires and receives an encryption $\gamma(g)$ of the value of g's output wire. Finally, $\mathcal{D}$ decrypts the output encryptions using sk_L and outputs 1 if the decrypted value does not equal to $C_\lambda(m_1, \cdots, m_\ell)$.

By construction of $\mathcal{D}$, when it is participating externally in experiment $\mathsf{Exp}^1_{\mathcal{D}}(1^s, P_i, z)$, it homomorphically evaluates C_λ exactly as in H_i, thus the probability that it outputs 1 is exactly p_i; on the other hand, when it is participating externally in $\mathsf{Exp}^2_{\mathcal{D}}(1^s, P_i, z)$, it homomorphically evaluates C_λ as in H_{i+1} and the probability that it outputs 1 is p_{i+1}. Since $|p_i - p_{i+1}|$ is non-negligible, $\mathcal{D}$ distinguishes the two experiments and thus violates the correctness of $pi\mathcal{O}$, which gives a contradiction. □

3.3.1 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 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 3 that always generates a ciphertext of 0 under the "trapdoor" public key hardwired inside. More precisely, we describe the class of samplers in Figure 4.

$\Pi = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{tKeyGen})$ 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^{SK}_s samples a trapdoor public key $\mathsf{tpk} \xleftarrow{\$} \mathsf{tKeyGen}(1^\lambda)$, 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 $l(\lambda)$.

Figure 4: 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 3.4. *Assume that* Π *is a trapdoor encryption scheme and piO is a* pIO *for the class of samplers* $\mathbf{S}^{\Pi}$ *in Figure 4. 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,

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

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 3) 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,

$$\begin{aligned}\Big\{\mathsf{view}[\mathcal{A}]_L^0 = \Big(\mathsf{pk}_0, \mathsf{evk}_L = (t\Lambda_1, \cdots, t\Lambda_{L-i}, t\Lambda_{L-i+1}, \cdots, t\Lambda_L), \boxed{\mathsf{Enc}_{\mathsf{pk}_0}(0)}\Big)\Big\}_\lambda \\ \approx \Big\{\mathsf{view}[\mathcal{A}]_L^1 = \Big(\mathsf{pk}_0, \mathsf{evk}_L = (t\Lambda_1, \cdots, t\Lambda_{L-i}, t\Lambda_{L-i+1}, \cdots, t\Lambda_L), \boxed{\mathsf{Enc}_{\mathsf{pk}_0}(1)}\Big)\Big\}_\lambda\end{aligned}$$

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

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

where the difference in the views of $\mathcal{A}$ in neighboring hybrids are highlighted in box.

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.

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

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 4 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. □

3.3.2 Instantiation of LHE

In this section, 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 3.2. Towards this, the focus is to instantiate the pIO scheme matching the trapdoor encryption scheme Π. Since Π defines the class of samplers for which pIO security is required. Properties of Π determines how strong the corresponding pIO is. As we will see, by varying the properties of Π, it leads to different instantiations of pIO.

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} [4] 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 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.

Claim 3.4.1. *Let Π be a ν-hiding trapdoor encryption scheme, where $\nu = \mathsf{negl}(\lambda)2^{-2l(\lambda)}$ and $l(\lambda)$ is an upperbound on the length of the ciphertexts of Π, and X is a function such that $X(s(\lambda)) = 2^{2l(\lambda)}$. Every sampler $D^{SK} \in \mathbf{S}^{\Pi}$ is a X-Ind sampler.*

Therefore, any X-Ind pIO scheme is a pIO scheme matching the ν-hiding trapdoor encryption scheme Π. Plugging them in the general transformation for LHE, that is, Proposition 3.2, we obtain:

Corollary 3.5 (LHE from ν-hiding trapdoor encryption scheme and X-Ind pIO)**.** *Let Π, ν, X be defined as in Claim 3.4.1. Assume the existence of an X-Ind pIO scheme $pi\mathcal{O}$. Then, Π can be transformed into a leveled homomorphic encryption scheme.*

By Claim 3.1.3, the existence of a ν-rerandomizable encryption scheme (in particular, a perfectly rerandomizable one) implies that of a ν-hiding trapdoor encryption scheme. Furthermore, by Theorem 2.5, X-Ind pIO can be constructed from any sub-exponentially indistinguishable IO and sub-exponentially secure OWFs. Therefore, we further obtain:

Corollary 3.6 (LHE from rerandomizable encryption and sub-exponentially 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 Claim 3.1.2 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

[4] 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 $s = s(\lambda)$.

the class $\mathbf{S}^{\Pi}$. Plugging a lossy encryption and worst-case-input piO in our general transformation Proposition 3.2, we obtain:

Corollary 3.7 (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 piO. 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 Claim 3.1.1, 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 3.8. *Let* Π *be any CPA encryption scheme and* Π' *the corresponding trapdoor encryption scheme. Assume the existence of a dynamic-input* piO *scheme piO 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 [GGHW14], piO for the specific class of samplers $\mathbf{S}^{\Pi'}$ circumvents the implausibility result. This is because the implausibility of [GGHW14] 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 construction in Conjecture 1 a potential candidate construction of dynamic-input piO for $\mathbf{S}^{\Pi'}$.

3.4 From LHE to FHE

In this section, we show how to transform the construction of leveled homomorphic encryption scheme LHE in Section 3.3 into a fully homomorphic one, *without relying on circular security.*

Overview: The first observation we have is that in the leveled FHE scheme LHE, the only dependency on the maximum level L of evaluation is the length of the evaluation key: The evaluation key evk consists of L obfuscated programs $\{\Lambda_i\}_{i\in[L]}$, one for each layer—call them layer evaluation keys. Except from the number of layer evaluation keys in evk, all other parameters in the scheme are independent of L. For instance, the size of each Λ_i is a fixed polynomial in λ, as well as the length of the public, secret keys $\mathsf{pk}_0, \mathsf{sk}_L$ and the length of the ciphertexts $\mathsf{Enc}_{\mathsf{pk}_0}(b)$; their concretes sizes depend only on the parameters of the underlying encryption scheme Π.

Towards enabling fully homomorphic operations, the naive idea is to publish in evk a slightly super-polynomial number of layer evaluation keys, that is, $\mathsf{evk} = \{\Lambda_i\}_{i\in[\mathcal{L}]}$, where $\mathcal{L} = \mathcal{L}(\lambda) = 2^{\omega(\log\lambda)}$ can be any super-polynomial function. Then to homomorphically evaluate a circuit of an arbitrary polynomial depth, one can simply perform the computation layer by layer as in LHE. However, this results in an evk of super-polynomial size.

The challenge is how to "compress" the size of the evaluation key evk. For this we use an idea developed in the recent works by Lin and Pass [LP14] and Bitansky, Garg and Telang [BGT14]. They used IO obfuscation to construct a succinct garbling scheme for bounded space Turing machine. Towards their goal, they first work with a non-succinct garbled program for bounded space Turing machine whose size is proportional to both the space and time complexity of the TM under consideration. They then "compress" the size of the non-succinct garbled TM, by generating using IO a "master program" that enables producing the large garbled TM in a "piecemeal fashion" at evaluation time (as opposed to generating and outputting them at the garbling time). The master program then becomes the final succinct garbled TM.

In our context, to "compress" the size of evk, we can apply the same idea: Instead of publishing the super-polynomially long sequence of layer evaluation keys in the evaluation key, we just publish a "master evaluation key" MEvk that enables the evaluator to generate each layer evaluation keys "on the fly", by evaluating MEvk on the index of the layer, $\Lambda_i = \mathsf{MEvk}(i)$ for every $i \in [\mathcal{L}]$. Then, to evaluate a circuit of arbitrary depth, the evaluator still proceeds layer by layer; for each layer ℓ, it first generates the layer evaluation key Λ_ℓ using the master program, and then evaluates that layer using Λ_ℓ as in LHE.

Recall that each evaluation key Λ_i is a piO obfuscation of the program $\mathsf{Prog}^{(\mathsf{sk}_{i-1},\mathsf{pk}_i)}$ with randomly generated keys sk_{i-1} and pk_i of Π. To avoid hurting the semantic security of Π, the randomness used for generating Λ_i and $\mathsf{sk}_{i-1}, \mathsf{pk}_i$ must be hidden. To ensure this, the master evaluation key MEvk itself will be the obfuscation of a program MProg that generates Λ_i (and in turn $\mathsf{sk}_{i-1}, \mathsf{pk}_i$). It seems that we can simply use piO to obfuscate the randomized program MProg. Unfortunately, this does not go through, because for different inputs, say i and $i+1$, the computation of $\mathsf{MProg}(i)$ and $\mathsf{MProg}(i+1)$ relies on correlated randomness in order to generate consistently $\mathsf{pk}_i, \mathsf{sk}_i$ needed for both Λ_i and Λ_{i+1} (as they are obfuscation of $\mathsf{Prog}^{(\mathsf{sk}_{i-1},\mathsf{pk}_i)}$ and $\mathsf{Prog}^{(\mathsf{sk}_i,\mathsf{pk}_{i+1})}$ with matching $\mathsf{pk}_i, \mathsf{sk}_i$). To overcome this problem, as in [LP14, BGT14], we directly obfuscate using IO a de-randomized version of MProg that internally uses pseudo-random coins generated using a puncturable PRF in a coordinated way to ensure consistency between the outputs for different inputs.

Furthermore, to show that the obfuscated program indeed does not reveal more information than the layer evaluation keys it produces and hence semantic security holds, as in [LP14, BGT14], we will require all underlying primitives, from the LHE, to IO, to puncturable PRF to have a distinguishing gap that is much smaller than the inverse of the domain size, namely, bounded by $\mathsf{negl}(\lambda)\mathcal{L}(\lambda)^{-1}$. Since the domain has a super-polynomial size $\mathcal{L}(\lambda)$, all primitives needs to be slightly super-polynomially secure. The reason that this is needed is because the proof of security goes through a sequence of hybrids that enumerate over all inputs in the domain of the obfuscated program (much like the exponentially long sequence of hybrids in the security proof of our construction of X-Ind piO from sub-exponentially secure IO and OWFs); for the hybrid argument to go through, we require all underlying primitives to have a distinguishing gap much smaller than the inverse size of the domain.

3.4.1 The Construction of FHE

We now describe our construction of FHE. The construction is identical to the level homomorphic scheme LHE, except from how the evaluation key is generated and how the evaluation procedure proceeds. Let $\mathcal{L}$ be any super-polynomial function $\mathcal{L}(\lambda) = 2^{\omega(\log \lambda)}$. Our construction of FHE relies on the following primitives: A trapdoor secure encyrption scheme $\Pi = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$,

a pIO for the matching class $\mathbf{S}^{\Pi}$, an IO for deterministic circuits $i\mathcal{O}$, and a puncturable PRF $(\mathrm{Key}, \mathrm{Puncture}, \mathsf{PRF})$, where all primitives are $\mathsf{negl}(\lambda)\mathcal{L}(\lambda)^{-1}$ indistinguishable.

sk_0, k, k' are strings of length $l(\lambda)$. i is an integer in $[\mathcal{L}]$.

Circuit $\mathsf{MProg}^{(sk_0,k,k')}(i)$**:**

Compute pseudo-random coins $r_{i-1} = \mathsf{PRF}(k, i-1)$, $r_i = \mathsf{PRF}(k, i)$ and $r'_i = \mathsf{PRF}(k', i)$;
compute key pairs $(\mathsf{pk}_{i-1}, \mathsf{sk}_{i-1}) = \mathsf{Gen}(1^\lambda; r_{i-1})$ and $(\mathsf{pk}_i, \mathsf{sk}_i) = \mathsf{Gen}(1^\lambda; r_i)$;
compute and output the obfuscated program $\Lambda_i = pi\mathcal{O}(1^s, P_i; r'_i)$, where $P_i = \mathsf{Prog}^{(\mathsf{sk}_{i-1}, \mathsf{pk}_i)}$.

Circuit $\mathsf{tMProg}^{(k,k')}(i)$**:**

Compute pseudo-random coins $r_i = \mathsf{PRF}(k, i)$ and $r'_i = \mathsf{PRF}(k', i)$;
compute $\mathsf{tpk}_i = \mathsf{tKeyGen}(1^\lambda;\ r_i)$;
output obfuscated program $t\Lambda_i = pi\mathcal{O}(1^s, tP_i)$, where $tP_i = \mathsf{tProg}^{(\mathsf{tpk}_i)}$.

Circuit $\mathsf{hMProg}_\ell^{(k,k')}(i)$**:**

1. If $i < \mathcal{L} - \ell$, compute and output Λ_i as in $\mathsf{MProg}^{(\mathsf{sk}_0,k,k')}$.
2. If $i = \mathcal{L} - \ell$, do:
   compute pseudo-random coins $r_{i-1} = \mathsf{PRF}(k, i-1)$, $r_i = \mathsf{PRF}(k, i)$ and $r'_i = \mathsf{PRF}(k', i)$;
   compute key pairs $(\mathsf{pk}_{i-1}, \mathsf{sk}_{i-1}) = \mathsf{Gen}(1^\lambda; r_{i-1})$ and $\boxed{\mathsf{tpk}_i = \mathsf{tKeyGen}(1^\lambda; r_i)}$;
   compute and output the obfuscated program $h\Lambda_i = pi\mathcal{O}(1^s, P_i; r'_i)$, where $P_i = \mathsf{Prog}^{(\mathsf{sk}_{i-1}, \mathsf{tpk}_i)}$.
3. If $i > \mathcal{L} - \ell$, compute and output $t\Lambda_i$ as in $\mathsf{tMProg}^{(k,k')}$.

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

Figure 5: Circuits used in the construction of FHE and its analysis

Construction of FHE: The four algorithms of the scheme proceed as follows:

- **Key generation:** $\mathsf{FHE.Keygen}(1^\lambda, 1^L)$ does the following:
  - sample a pair of keys $(\mathsf{pk}_0, \mathsf{sk}_0) \xleftarrow{\$} \mathsf{KeyGen}(1^\lambda)$ of Π;
  - sample two PRF keys $k, k' \xleftarrow{\$} \mathrm{Key}(1^\lambda)$.
  - obfuscate *using IO* the circuit $\mathsf{MProg}^{(\mathsf{sk}_0,k,k')}$ as described in Figure 5, that is, sample $\mathsf{MEvk} \xleftarrow{\$} i\mathcal{O}(1^{s'}, \mathsf{MProg}^{(\mathsf{sk}_0,k,k)})$ where the security parameter $s' = s'(\lambda)$ for obfuscation is an upper-bound on the size of $\mathsf{MProg}^{(\mathsf{sk}_0,k,k')}$.

  Finally outputs $\mathsf{pk} = \mathsf{pk}_0$, $\mathsf{sk} = (\mathsf{sk}_0, k)$, $\mathsf{evk} = \mathsf{MEvk}$.

  Note: *The PRF keys k, k' are used to generate the pseudo-random coins for sampling the key pairs $(\mathsf{pk}_i, \mathsf{sk}_i)$ and the layer evaluation keys Λ_i for every layer $i \in [\mathcal{L}]$. We define,*

$$\forall i \in [\mathcal{L}], \quad \mathsf{pk}_i, \mathsf{sk}_i \triangleq \mathsf{Gen}(1^\lambda; \mathsf{PRF}(k, i)),\ \ \Lambda_i \triangleq pi\mathcal{O}(1^s, \mathsf{Prog}^{(\mathsf{sk}_{i-1}, \mathsf{pk}_i)}; \mathsf{PRF}(k', i)) \tag{3}$$

We emphasize that these keys are merely well-defined, and are not actually generated nor published.

- **Encryption:** $\mathsf{FHE.Enc}_{\mathsf{pk}}(m)$ outputs a fresh encryption of m under $\mathsf{pk} = \mathsf{pk}_0$ using Π, $c \xleftarrow{\$} \mathsf{Enc}_{\mathsf{pk}_0}(m)$, and outputs ciphertext $(c, 0)$, where 0 indicates that it is a freshly generated ciphertext at level 0.
- **Decryption:** $\mathsf{FHE.Dec}_{\mathsf{sk}}((c, i))$ decrypts c, by first generating sk_i as in equation (3) (if $i = 0$, sk_0 is used), and then obtaining $m = \mathsf{Dec}_{\mathsf{sk}_i}(c)$.
- **Homomorphic evaluation:** $\mathsf{FHE.Eval}_{\mathsf{evk}}(C, c_1, \ldots, c_m)$ on input a layered circuit C (consisting of only NAND gates) of an arbitrary depth $\ell \leq \mathcal{L}(\lambda)$, evaluate C layer by layer; in iteration i, layer $i \in [\ell]$ is evaluated (the first layer is connected with the input wires) as follow:
  - 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 some key pk_{i-1}; in the first iteration, these encryptions are simply $c_1, \cdots, c_m$;
  - generate the i'th layer evaluation key Λ_i by evaluating $\Lambda_i = \mathsf{MEvk}(i)$.
  - for each NAND gate g in this layer i, let $\alpha(g), \beta(g)$ be the encryptions 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 encryption c^* generated in the last iteration together with the depth ℓ, (c^*, ℓ).

  Note: *Jumping ahead, as we show in the proof of homomorphism, when the evaluation proceeds correctly,* $\mathsf{pk}_{i-1}, \Lambda_i, \mathsf{pk}_i$ *above are exactly as defined in equation* (3).

  It follows from the correctness of Π, $pi\mathcal{O}$ and $i\mathcal{O}$ that FHE is correct.

Lemma 3.9. *If* Π, $pi\mathcal{O}$, $i\mathcal{O}$ *are correct, then* FHE *has homomorphism.*

Proof. Fix any polynomial $\ell(\lambda)$ and $n(\lambda)$, and any sequence of circuits C_λ of depth $\ell(\lambda)$ and inputs $m_1, \ldots, m_n \in \{0, 1\}$ (where $n = n(\lambda)$). We want to show that

$$\Pr\left[\mathsf{FHE.Dec}_{\mathsf{sk}}(\mathsf{FHE.Eval}_{\mathsf{evk}}(C_\lambda, c_1, \ldots, c_n)) \neq C_\lambda(m_1, \ldots, m_n)\right] = \mathsf{negl}(\lambda) \ ,$$

where $(\mathsf{pk}, \mathsf{evk}, \mathsf{sk}) \leftarrow \mathsf{FHE.Keygen}(1^\lambda)$ and $(c_i, 0) \leftarrow \mathsf{FHE.Enc}_{\mathsf{pk}}(m_i)$.

For every $i \in [\ell]$ (where $\ell = \ell(\lambda)$), it follows from the correctness of $i\mathcal{O}$ that with overwhelming probability over the choice of the random coins of $i\mathcal{O}$ in the key generation algorithm, the master evaluation key MEvk satisfies that

$$\Pr[(\mathsf{pk}, \mathsf{sk}, \mathsf{evk} = \mathsf{MEvk}) \xleftarrow{\$} \mathsf{FHE.Keygen}(1^\lambda) \ : \ \mathsf{MEvk}(i) \neq \Lambda_i] = \mathsf{negl}(\lambda) \ ,$$

where Λ_i is exactly as in equation (3). That is,

$$\mathsf{pk}_i, \mathsf{sk}_i = \mathsf{Gen}(1^\lambda; \mathsf{PRF}(k, i)), \quad \Lambda_i = pi\mathcal{O}(1^s, \mathsf{Prog}^{(\mathsf{sk}_{i-1}, \mathsf{pk}_i)}; \mathsf{PRF}(k', i))$$

By a union bound, it holds that

$$\Pr[(\mathsf{pk}, \mathsf{sk}, \mathsf{evk} = \mathsf{MEvk}) \xleftarrow{\$} \mathsf{FHE.Keygen}(1^\lambda) \ : \ \forall i \in [\ell], \ \mathsf{MEvk}(i) \neq \Lambda_i] = \mathsf{negl}(\lambda) \ .$$

Therefore, the homomorphic evaluation procedure computes $\Lambda_1, \cdots, \Lambda_\ell$ correctly with overwhelming probability; in this case, the evaluation procedure proceeds identically to that of LHE with a level bound ℓ, except that the layer evaluation keys $\{\Lambda_i\}$ are generated using pseudo-random coins, as opposed to truly random coins. Furthermore, the decryption procedure uses sk_ℓ to decrypt the homomorphism evaluated ciphertext, just as in LHE, except from, again, that sk_ℓ is generated using pseudo random coins. Then it follows from the homomorphism of LHE and the pseudo-randomness of PRF that the probability that decrypted value is not the correct output is negligible. □

3.4.2 Proof of Semantic Security of FHE

As in the proof of the semantic security of LHE in Section 3.3.1, the semantic security of FHE relies on that Π is a trapdoor encryption scheme, and $pi\mathcal{O}$ is a piO for the specific class of samplers $\mathbf{S}^{\Pi}$. Additionally, to ensure that it is secure to release the master evaluation key which produces the layer evaluation keys of LHE, we rely on an IO for circuits and a puncturable PRF. We note that since the master public key allows producing a slightly super-polynomial number $\mathcal{L}$ of layer keys, to prove it is secure to release it, we need to require all primitives $pi\mathcal{O}$, Π, $i\mathcal{O}$ and the PRF to have distinguishing gaps bounded by a slightly inverse super-polynomial $\mathsf{negl}(\lambda)\mathcal{L}^{-1}(\lambda)$. More precisely,

Lemma 3.10. *Let $\mathcal{L}$ be any slightly super-polynomial function, and μ any function, such that, $\mu(\lambda) = \mathsf{negl}(\lambda) \cdot \mathcal{L}(\lambda)^{-1}$. Assume the following primitives with distinguishing gaps bounded by μ:*

- *Π is a trapdoor encryption scheme,*
- *$pi\mathcal{O}$ is a piO for the class of samplers $\mathbf{S}^{\Pi}$,*
- *$i\mathcal{O}$ is an IO for circuits, and*
- *$(\mathrm{Key}, \mathrm{Puncture}, \mathsf{PRF})$ is a puncturable PRF.*

Then, the scheme FHE described in Section 3.4.1 is semantic secure.

Proof. Fix any polynomial time adversary $\mathcal{A}$. We want to show that for every $\lambda \in \mathbb{N}$, it holds that,

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

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

Towards this, we consider two sequences of hybrids $H_0^b, \cdots, H_{\mathcal{L}}^b$ for $b \in \{0, 1\}$, each of *super-polynomial* length $\mathcal{L} = \mathcal{L}(\lambda)$. 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 $H_{\mathcal{L}}^0, H_{\mathcal{L}}^1$, the views of $\mathcal{A}$ are μ'-indistinguishable, for a distinguishing gap $\mu' = c\mu$ for a sufficiently large constant c. Below we formally describe all the hybrids.

Hybrid real^b: Hybrid real^b is an honest CPA game with $\mathcal{A}$, where $\mathcal{A}$ receives $(\mathsf{pk}, \mathsf{evk}, c^* \xleftarrow{\$} \mathsf{FHE.Enc}_{\mathsf{pk}}(b))$ for freshly sampled $(\mathsf{pk}, \mathsf{evk}, \mathsf{sk}) \xleftarrow{\$} \mathsf{FHE.Keygen}(1^\lambda)$. By construction of FHE, $\mathsf{pk} = \mathsf{pk}_0$, $\mathsf{evk} = \mathsf{MEvk}$ which is an obfuscation of $\mathsf{MProg}^{(\mathsf{sk}_0, k, k')}$ for randomly sampled k, k', and c^* contains a ciphertext c_b of Π and level 0; thus, the view of $\mathcal{A}$ is,

$$\mathsf{view}[\mathcal{A}]^b_{\mathsf{real}} = \left(\mathsf{pk}_0,\ \mathsf{MEvk} \xleftarrow{\$} i\mathcal{O}(1^{s'}, \mathsf{MProg}^{(\mathsf{sk}_0, k, k')}),\ (\mathsf{Enc}_{\mathsf{pk}_0}(b), 0)\right) \ ,$$

where $\mathsf{pk}_0, \mathsf{sk}_0, k, k'$ are all randomly sampled.

Hybrid ideal^b: Hybrid ideal^b is identical to real^b except that the evaluation key is sampled in a different way. More specifically, instead of obfuscating the honest master program $\mathsf{MProg}^{(sk_0,k,k')}$, obfuscate the trapdoor master program $\mathsf{tMProg}^{(k,k')}$ as described in Figure 5 to obtain a trapdoor evaluation key tMEvk. Thus, the view of $\mathcal{A}$ is,

$$\mathsf{view}[\mathcal{A}]^b_{\mathsf{ideal}} = \left(\mathsf{pk}_0,\ \mathsf{tMEvk} \xleftarrow{\$} i\mathcal{O}(1^{s'}, \mathsf{tMProg}^{(k,k')}),\ (\mathsf{Enc}_{\mathsf{pk}_0}(b), 0)\right) ,$$

where $\mathsf{pk}_0, \mathsf{sk}_0, k, k'$ are all randomly sampled.

Note that in this hybrid, the view of the adversary is completely independent of the secret key sk_0, and the only difference between the view $\mathsf{view}[\mathcal{A}]^0_{\mathcal{L}}$ and $\mathsf{view}[\mathcal{A}]^1_{\mathcal{L}}$ is which bit $b = 0$ or 1 is encrypted. It follows from the semantic security of Π that these views are μ-indistinguishable.

$$\left\{\mathsf{view}[\mathcal{A}]^0_{\mathsf{ideal}}\right\}_\lambda \overset{\mu}{\approx} \left\{\mathsf{view}[\mathcal{A}]^1_{\mathsf{ideal}}\right\}_\lambda$$

Intermediate Hybrids H^b_ℓ for $0 \le \ell \le \mathcal{L}$: Hybrid H^b_ℓ proceeds identically to H^b_0 except that the evaluation key is sampled in a different way. More specifically, instead of obfuscating the honest master program $\mathsf{MProg}^{(sk_0,k,k')}$, obfuscate the hybrid master program $\mathsf{hMProg}^{(sk_0,k,k')}_\ell$ as described in Figure 5 to obtain a hybrid evaluation key hMEvk_ℓ. Thus, the view of $\mathcal{A}$ is,

$$\mathsf{view}[\mathcal{A}]^b_\ell = \left(\mathsf{pk}_0,\ \mathsf{hMEvk}_\ell \xleftarrow{\$} i\mathcal{O}(1^{s'}, \mathsf{hMProg}^{(\mathsf{sk}_0,k,k')}_\ell),\ (\mathsf{Enc}_{\mathsf{pk}_0}(b), 0)\right) ,$$

where $\mathsf{pk}_0, \mathsf{sk}_0, k, k'$ are all randomly sampled.

It is easy to see that since $\mathsf{hMProg}^{(sk_0,k,k')}_0$ has the same functionality as $\mathsf{MProg}^{(sk_0,k,k')}$ and $\mathsf{hMProg}^{(sk_0,k,k')}_{\mathcal{L}}$ has the same functionality as $\mathsf{tMProg}^{(k,k')}$. Therefore, it follows from the μ-indistinguishability of $i\mathcal{O}$ that

$$\left\{\mathsf{view}[\mathcal{A}]^b_0\right\}_\lambda \overset{\mu}{\approx} \left\{\mathsf{view}[\mathcal{A}]^b_{\mathsf{real}}\right\}_\lambda \qquad \left\{\mathsf{view}[\mathcal{A}]^b_{\mathcal{L}}\right\}_\lambda \overset{\mu}{\approx} \left\{\mathsf{view}[\mathcal{A}]^b_{\mathsf{ideal}}\right\}_\lambda$$

Therefore, given the indistinguishability of the ideal views of $\mathcal{A}$ when $b = 0$ or 1 shown for hybrids ideal^b, to show the indistinguishability of the real views of $\mathcal{A}$, it suffices to show the indistinguishability of views of A in neighboring hybrids H^b_i and H^b_{i+1}. More formally,

Claim 3.10.1. *For every $b \in \{0,1\}$ and $i \ge 0$,*

$$\left\{\mathsf{view}[\mathcal{A}]^0_\ell\right\}_\lambda \overset{\mu'}{\approx} \left\{\mathsf{view}[\mathcal{A}]^1_{\ell+1}\right\}_\lambda$$

Before proving Claim 3.10.1, note that by a hybrid argument over the sequence of experiments, $\mathsf{real}^b, H^b_1, \cdots H^b_{\mathcal{L}}, \mathsf{ideal}^b$ and the indistinguishability of ideal^0 and ideal^1. We have that the real views of $\mathcal{A}$ are indistinguishable and hence semantic security holds.

Proof of Claim 3.10.1. Fix any $b \in \{0,1\}$ and $i \ge 0$. We want to show that the views of $\mathcal{A}$ in H^b_i and H^b_{i+1} are indistinguishable.

$$\left\{\mathsf{view}[\mathcal{A}]^b_\ell = \left(\mathsf{pk}_0,\ \mathsf{hMEvk}_\ell \xleftarrow{\$} i\mathcal{O}(1^{s'}, \boxed{\mathsf{hMProg}^{(\mathsf{sk}_0,k,k')}_\ell}),\ (\mathsf{Enc}_{\mathsf{pk}_0}(b), 0)\right)\right\}_\lambda$$
$$\overset{\mu'}{\approx} \left\{\mathsf{view}[\mathcal{A}]^b_{\ell+1} = \left(\mathsf{pk}_0,\ \mathsf{hMEvk}_{\ell+1} \xleftarrow{\$} i\mathcal{O}(1^{s'}, \boxed{\mathsf{hMProg}^{(\mathsf{sk}_0,k,k')}_{\ell+1}}),\ (\mathsf{Enc}_{\mathsf{pk}_0}(b), 0)\right)\right\}_\lambda$$

The only difference between the views of $\mathcal{A}$ lies in which hybrid master program is obfuscated; furthermore, notice that $C_0 = \mathsf{hMProg}_{\ell}^{(\mathsf{sk}_0,k,k')}$ and $C_1 = \mathsf{hMProg}_{\ell+1}^{(\mathsf{sk}_0,k,k')}$ differ at only two inputs $w = \mathcal{L} - \ell$ and $w-1 = \mathcal{L} - \ell - 1$:

- At w, C_0 outputs a pIO obfuscation of the hybrid program $\mathsf{Prog}^{(\mathsf{sk}_{w-1},\mathsf{tpk}_w)}$, where as C_1 outputs an obfuscation of the trapdoor program $\mathsf{tProg}^{(\mathsf{tpk}_w)}$, all random variables are generated with pseudo-random coins produced using k, k'.
- At $w-1$, C_0 outputs an obfuscation of the honest program $\mathsf{Prog}^{(\mathsf{sk}_{w-2},\mathsf{pk}_{w-1})}$, whereas C_1 outputs an obfuscation of the hybrid program $\mathsf{Prog}^{(\mathsf{sk}_{w-2},\mathsf{tpk}_{w-1})}$.

At all other inputs, C_0 and C_1 outputs the same.

To show the above indistinguishability, it suffices to show that for every fixed sequence of keys $\{(\mathsf{pk}_0, \mathsf{sk}_0)\}$, the following are indistinguishable.

$$\left\{k, k' \xleftarrow{\$} \mathrm{Key}(1^\lambda), \boxed{C_0 = \mathsf{hMProg}_{\ell}^{(\mathsf{sk}_0,k,k')}} \ : \ \mathsf{hMEvk}_\ell \xleftarrow{\$} i\mathcal{O}(1^{s'}, C_0)\right\}_\lambda$$
$$\overset{\mu'}{\approx} \left\{k, k' \xleftarrow{\$} \mathrm{Key}(1^\lambda), \boxed{C_1 = \mathsf{hMProg}_{\ell+1}^{(\mathsf{sk}_0,k,k')}} \ : \ \mathsf{hMEvk}_{\ell+1} \xleftarrow{\$} i\mathcal{O}(1^{s'}, C_1)\right\}_\lambda$$

Towards this, we will consider a sequence of hybrids G_0 to G_6, where G_0 and G_6 samples obfuscations of programs $\Gamma_0 = C_0$ and $\Gamma_6 = C_1$ as in the above two ensembles. The intermediate G_i hybrids produces the obfuscation of some hybrid program Γ_i as described below:

Hybrid G_1: Hybrid G_1 obfuscates a program Γ_1 constructed as follows: After sampling two PRF keys k and k'; puncture the PRF key k at points w and $w-1$, and k' at w:

$$k(w, w-1) = \mathrm{Puncture}(k, \{w, w-1\}), \quad k'(w) = \mathrm{Puncture}(k', w) \ ;$$

furthermore, let r_{w-1}, r_w and r'_w be the outputs of the PRF with key k at points $w-1$ and w, and with k' at w, that is,

$$r_{w-1} = \mathsf{PRF}(k, w-1), \ r_w = \mathsf{PRF}(k, w), \ r'_w = \mathsf{PRF}(k', w)$$

Note that in Γ_0, r_{w-1} and r_w are only used to generate key pairs for layers $w-1$ and w, while r'_w is only used to generate the w'th layer evaluation key. Directly compute the variables that depend on r_{w-1}, r_w, r'_w:

$$(\mathsf{pk}_{w-1}, \mathsf{sk}_{w-1}) = \mathsf{Gen}(1^\lambda; r_{w-1}), \ \mathsf{tpk}_w = \mathsf{tKeyGen}(1^\lambda; r_w),$$
$$h\Lambda_w = pi\mathcal{O}(1^s, P_w; r'_w), \ P_w = \mathsf{Prog}^{(sk_{w-1},\mathsf{tpk}_w)}$$

The program Γ_1 is identical to Γ_0, except from the following two modifications: (1) instead of having full keys k, k' hardwired inside, Γ_1 has the punctured keys $k(w, w-1)$, $k'(w)$ hardwired in, together with values $\mathsf{pk}_{w-1}, h\Lambda_w$; (2) Γ_1 proceeds identically to Γ_0 for all inputs except from $w-1, w$; for input w, it directly outputs $h\Lambda_w$, and for $w-1$, it uses the hardwired key pk_{w-1} to compute Λ_{w-1} as in Γ_0.

Since Γ_0 and Γ_1 has the same functionality, it follows from the μ-indistinguishability guarantees of $i\mathcal{O}$ that their obfuscation is μ-indistinguishable.

Hybrid G_2: Hybrid G_2 proceeds identically to G_1, except that it computes the keys and obfuscated program to be hardwired using truly random coins, instead of pseudo-random coins, that is,

$$(\mathsf{pk}_{w-1}, \mathsf{sk}_{w-1}) \xleftarrow{\$} \mathsf{Gen}(1^\lambda),\ \mathsf{tpk}_w \xleftarrow{\$} \mathsf{tKeyGen}(1^\lambda),$$
$$h\Lambda_w \xleftarrow{\$} pi\mathcal{O}(1^s, P_w),\ P_w = \mathsf{Prog}^{(sk_{w-1}, \mathsf{tpk}_w)}$$

G_2 then obfuscates the program Γ_2, which is identical to Γ_1 except that it contains $\mathsf{pk}_{w-1}, h\Lambda_w$ generated using truly random strings as above.

It follows from the pseudo-randomness of the puncturable PRF that G_4 and G_5 are μ-indistinguishable.

Hybrid G_3: Hybrid G_3 proceeds identically to G_2, except that instead of sampling $h\Lambda_w$ as in G_2, it samples $t\Lambda_w$ that is an obfuscation of the trapdoor program $tP_w = \mathsf{tProg}^{(\mathsf{tpk}_w)}$ (instead of the hybrid program).

$$t\Lambda_w \xleftarrow{\$} pi\mathcal{O}(1^s, tP_w),\ tP_w = \mathsf{tProg}^{(\mathsf{tpk}_w)}$$

G_3 then obfuscates Γ_3 that is identical to Γ_2 except that $t\Lambda_w$ (together with pk_{w-1}) is hardwired in instead of $h\Lambda_w$.

It follows from the pIO security for the class $\mathbf{S}^\Pi$ that $t\Lambda_w$ and $h\Lambda_w$ are indistinguishable. More precisely, consider any fixed sequence of $\{\mathsf{pk}_{w-1}, \mathsf{sk}_{w-1}\}$, and let $SK = \{\mathsf{sk}_{w-1}\}$. The distribution D^{SK} samples the following tuple:

$$(P_w = \mathsf{Prog}^{(\mathsf{sk}_{w-1}, \mathsf{tpk}_w)},\ tP_w = \mathsf{tProg}^{(\mathsf{tpk}_w)}, z = \mathsf{tpk}_w) \xleftarrow{\$} D_s^{SK}$$

Thus by pIO security w.r.t. D^{SK}, the following two ensembles are μ-indistinguishable.

$$\left\{P_w, tP_w, h\Lambda_w \xleftarrow{\$} pi\mathcal{O}(1^s, P_w), \mathsf{tpk}_w\right\}_\lambda \overset{\mu}{\approx} \left\{P_w, tP_w, t\Lambda_w \xleftarrow{\$} pi\mathcal{O}(1^s, tP_w), \mathsf{tpk}_w\right\}_\lambda$$

Since this indistinguishability holds for every sequence of $\{\mathsf{pk}_{w-1}, \mathsf{sk}_{w-1}\}$, it directly implies the μ-indistinguishability of hybrids G_3 and G_2.

Hybrid G_4: Hybrid G_4 obfuscates a program Γ_4 that is identical to Γ_3 except that the hardwired honest public key pk_{w-1} is replaced by a trapdoor public key tpk_{w-1}

$$\mathsf{tpk}_{w-1} \xleftarrow{\$} \mathsf{tKeyGen}(1^\lambda)$$

It follows directly from the μ-indistinguishability between the trapdoor and honest keys of Π that G_4 and G_3 are μ-indistinguishable.

Hybrid G_5: Hybrid G_5 obfuscates a program Γ_5 that is identical to Γ_4 except that the values hardwired in Γ_5, namely $\mathsf{tpk}_{w-1}, t\Lambda_w$ are generated using pseudo-random strings r_{w-1}, r_w, r'_w as in hybrid G_1:

$$\mathsf{tpk}_{w-1} = \mathsf{tKeyGen}(1^\lambda; r_{w-1}),\ \mathsf{tpk}_w = \mathsf{tKeyGen}(1^\lambda; r_w),$$
$$t\Lambda_w = pi\mathcal{O}(1^s, tP_w; r'_w),\ tP_w = \mathsf{Prog}^{(\mathsf{tpk}_w)}$$

It follows from the pseudo-randomness of the puncturable PRF that G_4 and G_5 are μ-indistinguishable.

Hybrid G_6: Hybrid G_6 outputs an obfuscation of $\Gamma_6 = C_1 = \mathsf{hMProg}_{\ell+1}^{(\mathsf{sk}_0,k,k')}$, which has the same functionality as Γ_5. Therefore it follows from the indistinguishability guarantee of $i\mathcal{O}$ that G_5 is μ-indistinguishable from G_6

It then follows from a hybrid argument that obfuscation of C_0 and C_1 are μ'-indistinguishable, where $\mu'(\lambda) = c\mu(\lambda)$ with a sufficiently large constant c. This concludes the claim. □

□

3.4.3 On the Generality of the Transformation to FHE

Our method of using a slightly super-polynomially secure IO and puncturable PRF to compress the size of a long sequence of layer evaluation keys can actually be applied generally to any LHE scheme that has a *fixed decryption depth* (independent of the maximum level of evaluation). 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 follows the same two step approach as the transformation for our specific LHE based on pIO.

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,

$$\Lambda_\ell = (\mathsf{pk}_\ell, \mathsf{evk}_\ell, c_\ell), \quad \text{where} \quad (\mathsf{pk}_\ell, \mathsf{sk}_\ell, \mathsf{evk}_\ell) \xleftarrow{\$} \mathsf{LHE.Keygen}(1^\lambda, 1^{D'}),\ D' = D_{\mathsf{LHE.Dec}} + 1,$$
$$\text{and} \quad c_\ell = \mathsf{LHE.Enc}_{\mathsf{pk}_\ell}(\mathsf{sk}_{\ell-1})$$

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. Following the same approach as before, 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,

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

Since the size of the master program $\Gamma^{(k,k')}$ is a fixed polynomial in λ, the new evaluation key $\mathsf{evk} \xleftarrow{\$} i\mathcal{O}(1^s, \Gamma^{k,k'})$ 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 essentially the same proof as in Lemma 3.10 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 $\mu(\lambda) = \mathsf{negl}(\lambda)\mathcal{L}(\lambda)^{-1}$.

The concrete construction of FHE and its proof of correctness and security are essentially the same as that for the FHE scheme from our specific construction of LHE from piO in Section 3.4; therefore, we here defer the details to the full version.

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 [Gen09, BV11, BGV12, Bra12, GSW13] satisfy this property. Thus, if these constructions are slightly super-polynomially secure, by assuming slightly stronger underlying assumptions (for instance the LHE scheme of [BV11] 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 3.3 has a fixed decryption depth, since its decryption algorithm is identical to that of the underlying trapdoor encryption scheme. Therefore, the specific transformation described in Section 3.4 can be derived as a special case of the above general transformation.

4 Application 2: Bootstrapping Indistinguishability Obfuscation

Putting together a randomized encoding for $\mathbf{P}/poly$ and our construction of X-Ind piO from sub-exponentially secure IO and OWFs in Section 2.6, we obtain a way to bootstrap an indistinguishability obfuscator for relatively weak (deterministic) circuit classes (such as $\mathbf{TC}^0$ or $\mathbf{NC}^1$) into an indistinguishability obfuscator for $\mathbf{P}/poly$. We note that Garg et al. [GGH+13] show how to do this assuming the existence of indistinguishability obfuscation for some weak circuit class **WEAK** (to be defined below), as well as a fully homomorphic encryption scheme whose decryption can be computed in **WEAK**. Such an FHE scheme can be instantiated based on the LWE assumption [BV11]. Our goal is to perform bootstrapping for indistinguishability obfuscation, without assuming that FHE schemes exist (but instead assuming sub-exponentially hard indistinguishability obfuscation for **WEAK** and sub-exponentially hard puncturable PRF in **WEAK**.) Applebaum [App13] shows how to bootstrap VBB obfuscations from **WEAK** to $\mathbf{P}/poly$ using randomized encodings: our technique below is inspired by his transformation.

Let **WEAK** be a class of circuits with the following properties: (1) $\mathbf{NC}^0 \subseteq \mathbf{WEAK}$; (2) (Closure under concatenation) If each output bit of a multi-output function f is computable in **WEAK** then so is f; (3) (Closure under composition) If $f \in \mathbf{WEAK}$ and $g \in \mathbf{NC}^0$, then $g \circ f \in \mathbf{WEAK}$, where $\circ$ denotes function composition. This definition is the same as in [App13]. Clearly, **WEAK** includes well-known low-depth classes such as $\mathbf{TC}^0$ and $\mathbf{NC}^1$.

Theorem 4.1. *Assume the existence of a sub-exponentially hard indistinguishability obfuscator $i\mathcal{O}_{\mathbf{WEAK}}$ for a circuit class* **WEAK** *as above, a sub-exponentially hard puncturable PRF computable in* **WEAK**. *Then, there is an indistinguishability obfuscator for $\mathbf{P}/poly$.*

Proof. Let C be a polynomial size circuit that we wish to obfuscate, and let $\mathsf{RE}(1^\lambda, C, x)$ be the function that takes as input C of size bounded by λ and an input x and produces a randomized encoding $(\hat{C}, \hat{x})$. From the modern constructions of garbled circuits (see, e.g., [AIK06]), we know that RE can be implemented in $\mathbf{NC}^0$, assuming the existence of a pseudorandom generator in $\oplus\mathbf{L}/poly$. Our obfuscation of C is exactly an X-Ind piO obfuscator for the probabilistic circuit $\mathsf{RE}(1^{\lambda'}, C, \cdot)$ using a sufficiently large security parameter $\lambda' = \mathrm{poly}(\lambda)$. That is,

$$\bar{C} \xleftarrow{\$} i\mathcal{O}(1^\lambda, C) \ : \ \bar{C} \xleftarrow{\$} pi\mathcal{O}(1^s, P) \text{ where } P = \mathsf{RE}(1^{\lambda'}, C, \cdot)),\ s = s(\lambda) = |P|$$

The function X needs to be sufficiently large; we set it in the security proof of $i\mathcal{O}$ below.

To show the security of $i\mathcal{O}$, consider a sampler $(C_1, C_2, z) \xleftarrow{\$} D_\lambda$ over deterministic circuits of size at most λ, such that, with overwhelming probability C_1 and C_2 are functionally equivalent. We want to show that the following ensembles are indistinguishable:

$$\left\{(C_1, C_2, z) \xleftarrow{\$} D_\lambda \ : \ (C_1, C_2, i\mathcal{O}(C_1), z)\right\}_\lambda \approx \left\{(C_1, C_2, z) \xleftarrow{\$} D_\lambda \ : \ (C_1, C_2, i\mathcal{O}(C_2), z)\right\}_\lambda$$

This follows from the following two observations:

- Assuming that the PRG underlying the randomized encoding RE is subexponentially hard, then by using a sufficiently large security parameter $\lambda'(\lambda) = \mathrm{poly}(\lambda)$, the following ensembles are μ-indistinguishable for any input $x \in \{0,1\}^\lambda$, where $\mu(\lambda) = \mathsf{negl}(\lambda)2^\lambda$.

$$\{(C_1, C_2, z) \xleftarrow{\$} D_\lambda \ : \ (C_1, C_2, \mathsf{RE}(1^{\lambda'}, C_1, x), z\}_\lambda \\ \overset{\mu}{\approx} \{(C_1, C_2, z) \xleftarrow{\$} D_\lambda \ : \ (C_1, C_2, \mathsf{RE}(1^{\lambda'}, C_2, x), z\}_\lambda$$

  Let $P_b = \mathsf{RE}(1^{\lambda'}, C_b, \cdot)$ and $s(\lambda) = |P_b|$; set the function X to be such that $X(s(\lambda)) = 2^\lambda$. Then it follows from the above indistinguishability that the modified distribution D'_s that samples (P_1, P_2, z), where P_b and z correspond to C_b and z sampled according to D, is an X-Ind sampler over $\mathbf{NC}^0$ circuits.

- By Theorem 2.5, using a subexponentially hard indistinguishability obfuscator for the circuit class **WEAK** and a subexponentially hard puncturable PRF computable in **WEAK**, we can construct an X-Ind piO scheme $pi\mathcal{O}$ for the circuit class $\mathbf{NC}^0$.

  By the definition of X-Ind piO, and the fact that D' is an X-Ind sampler, we have

$$\left\{(P_1, P_2, z) \xleftarrow{\$} D'_s \ : \ (P_1, P_2, pi\mathcal{O}(1^s, P_1), z)\right\}_s \approx \left\{(P_1, P_2, z) \xleftarrow{\$} D'_s \ : \ (P_1, P_2, pi\mathcal{O}(1^s, P_2), z)\right\}_s ,$$

which implies that

$$\left\{(C_1, C_2, z) \xleftarrow{\$} D_\lambda \ : \ (C_1, C_2, pi\mathcal{O}(1^s, \mathsf{RE}(1^{\lambda'}, C_1, \cdot)), z)\right\}_\lambda \\ \overset{c}{\approx} \left\{(C_1, C_2, z) \xleftarrow{\$} D_\lambda \ : \ (C_1, C_2, pi\mathcal{O}(1^s, \mathsf{RE}(1^{\lambda'}, C_2, \cdot)), z)\right\}_\lambda$$

Finally, note that sub-exponentially indistinguishable PRF in $\mathbf{NC}^1$ implies a sub-exponentially hard PRG in $\oplus\mathbf{L}/poly$. This finishes the proof. □

The above theorem directly yields the following corollary:

Corollary 4.2. *Assume the existence of a sub-exponentially hard indistinguishability obfuscator $i\mathcal{O}_{\mathbf{NC}^1}$ for the class $\mathbf{NC}^1$, subexponential indistinguishable puncturable PRF in $\mathbf{NC}^1$. Then, there is an indistinguishability obfuscator for $\mathbf{P}/poly$.*

We note that, so far, there are two constructions of puncturable PRFs in $\mathbf{NC}^1$ [BLMR13], based on the Learning With Error (LWE) problem or the decision linear assumptions on multilinear maps. Assuming that these underlying assumptions are sub-exponential hard, we get sub-exponentially indistinguishable puncturable PRF in $\mathbf{NC}^1$.

4.1 Bootstrapping Worst-case-input piO from $\mathbf{NC}^0$ to $\mathbf{P}/poly$

Using similar idea as above for bootstrapping IO, we show how to bootstrap worst-case-input piO for circuits in $\mathbf{NC}^0$ to worst-case-input piO for $\mathbf{P}/poly$.

Theorem 4.3. *Assume the existence of a worst-case-input piO scheme $pi\mathcal{O}_{\mathbf{NC}^0}$ for circuits in $\mathbf{NC}^0$, and a pseudo-random generator in $\oplus\mathbf{L}/poly$. Then, there is a worst-case-input piO scheme $pi\mathcal{O}_{\mathbf{P}/poly}$ for $\mathbf{P}/poly$.*

Proof. We first present the construction of piO obfuscator $pi\mathcal{O}_{\mathbf{P}/poly}$ for $\mathbf{P}/poly$, and then analyze its security. Assuming a pseudo-random generator in $\oplus\mathbf{L}/poly$, there is a randomized encoding scheme $(\mathsf{RE}, \mathsf{Decode})$ for $\mathbf{P}/poly$ in $\mathbf{NC}^0$.

Construction of $pi\mathcal{O}_{\mathbf{P}/poly}$: On input 1^λ and a probabilistic circuit $C(\cdot\ ;\ \cdot)$ of size at most λ, our piO obfuscator proceeds as follows:

1. View C as a deterministic circuit with input x and r. Compute the randomized encoding circuit $\hat{C}(x, r\ ;\ r') = \mathsf{RE}(1^\lambda, C, (x, r)\ ;\ r')$; let $\tilde{C}$ be the randomized circuit that views the second input r of $\hat{C}$ as a part of the random input, that is, $\tilde{C}(x\ ;\ r, r') = \hat{C}(x, r\ ;\ r')$.

   We have that the size and depth of $\tilde{C}$ is the same as $\hat{C}$, which are respectively $s(\lambda, |C|)$ and c for a fixed polynomial s and constant c associated with the randomized encoding scheme.

2. Obfuscate the circuit $\tilde{C}$ to obtain $\Lambda = pi\mathcal{O}_{\mathbf{NC}^0}(c, 1^s, \tilde{C})$, and output circuit $C' = \mathsf{Decode} \circ \Lambda$, that is, $C'(x) = \mathsf{Decode}(\Lambda(x))$ for all input x.

Correctness: Suppose for contradiction that there is a distinguisher $\mathcal{D}$ that violates the correctness of $pi\mathcal{O}_{\mathbf{P}/poly}$, that is, for an infinite sequence of $\lambda \in \mathbb{N}$, there exists a circuit C of size at most λ, and auxiliary input $z \in \{0,1\}^{\mathrm{poly}(\lambda)}$, such that, the advantage of $\mathcal{D}$ in distinguishing experiments $\mathsf{Exp}^1_{\mathcal{D}}(1^\lambda, C, z)$ and $\mathsf{Exp}^2_{\mathcal{D}}(1^\lambda, C, z)$ is an inverse polynomial $1/p(\lambda)$. (Recall that in Exp^1, $\mathcal{D}$ has access to oracle $C(\cdot; r)$ with independently sampled random coins r, and in Exp^2, it has access to oracle $\Lambda(\cdot) \xleftarrow{\$} pi\mathcal{O}_{\mathbf{P}/poly}(1^\lambda, C)$). We derive a contradiction, by building a distinguishers $\mathcal{D}'$ that violates the correctness of $pi\mathcal{O}_{\mathbf{NC}^0}(c, \cdot, \cdot)$ with respect to security parameter s and circuit $\tilde{C}$.

The distinguisher $\mathcal{D}'$ on input $(1^s, \tilde{C}, z)$, internally runs $\mathcal{D}(1^\lambda, C, z)$ and externally participates in either $\mathsf{Exp}^1_{\mathcal{D}}(1^s, \tilde{C}, z)$ or $\mathsf{Exp}^2_{\mathcal{D}}(1^s, \tilde{C}, z)$ w.r.t. $pi\mathcal{O}_{\mathbf{NC}^1}$ as follows: In each iteration i, it relays the input x_i chosen by $\mathcal{D}$ to its oracle; and upon receiving an output y_i, it feeds $\mathcal{D}$ the value $\mathsf{Decode}(y_i)$; finally, $\mathcal{D}'$ outputs the bit returned by $\mathcal{D}$. By construction, for every $i \in \{1,2\}$, the view of $\mathcal{D}$ emulated by $\mathcal{D}'$ when participating in experiment $\mathsf{Exp}^i_{\mathcal{D}'}(1^s, \tilde{C}, z)$ (w.r.t. $pi\mathcal{O}_{\mathbf{NC}^0}$) is identical to the view of $\mathcal{D}$ in $\mathsf{Exp}^i_{\mathcal{D}}(1^\lambda, C, z)$ (w.r.t. $pi\mathcal{O}_{\mathbf{P}/poly}$). Thus, by our hypothesis, $\mathcal{D}'$ achieves an advantage $1/p(\lambda) = 1/q(s)$, which violates the correctness of $pi\mathcal{O}_{\mathbf{NC}^0}$ and gives a contradiction.

Security: To show the security of $pi\mathcal{O}_{\mathbf{P}/poly}$, consider a worst-case-input indistinguishable sampler D. We want to show the following indistinguishability:

$$\left\{(C_1, C_2, z) \xleftarrow{\$} D_\lambda \ : \ (C_1, C_2, pi\mathcal{O}_{\mathbf{P}/poly}(1^\lambda, C_1), z)\right\}_\lambda \approx \left\{(C_1, C_2, z) \xleftarrow{\$} D_\lambda \ : \ (C_1, C_2, pi\mathcal{O}_{\mathbf{P}/poly}(1^\lambda, C_2), z)\right\}_\lambda \quad (4)$$

Towards this, we will reduce the above indistinguishability to the indistinguishability of $pi\mathcal{O}_{\mathbf{NC}^0}$ for $\mathbf{NC}^0$ w.r.t. a worst-case-input indistinguishability sampler D' over circuits in $\mathbf{NC}^0$ defined as follows.

$$(\tilde{C}_1, \tilde{C}_2, z) \xleftarrow{\$} D'_s \ : \ (C_1, C_2, z) \xleftarrow{\$} D_\lambda, \ \tilde{C}_b(x \ ; \ r, r') = \mathsf{RE}(1^\lambda, C, (x, r) \ ; \ r')$$

Since the randomized encoding algorithm RE is in $\mathbf{NC}^0$, so are $\tilde{C}_b$. Furthermore, we show that D' is also a worst-case-input indistinguishable sampler. This is because that D is a worst-case-input indistinguishable sampler, meaning for every adversary $(\mathcal{A}_1, \mathcal{A}_2)$, where the former is unbounded and the latter is PPT, with overwhelming probability over the choice of (C_1, C_2, z), and the choice of $(x, st) \xleftarrow{\$} \mathcal{A}_1(C_1, C_2, z)$, the advantage of $\mathcal{A}_2(st, C_1, C_2, C_b(x), z)$ in guessing b is negligible. It follows from the fact that $\tilde{C}_b$ can be constructed from C_b and the security of the randomized encoding scheme RE that for any $(\mathcal{A}'_1, \mathcal{A}'_2)$, over the randomness of D' and $(x, st) \xleftarrow{\$} \mathcal{A}'_1(\tilde{C}_1, \tilde{C}_2, z)$, no second stage adversary $\mathcal{A}'_2$ can tell apart $(st, \tilde{C}_1, \tilde{C}_2, \tilde{C}_b(x), z)$ for $b = 1$ or 2, as they are indistinguishable to $(st, \tilde{C}_1, \tilde{C}_2, \mathsf{Sim}(1^s, C_b(x; r)), z)$ where Sim is the simulator for the randomized encoding.

Then it follows directly from the fact that $pi\mathcal{O}_{\mathbf{NC}^0}$ is a worst-case-input piO and that $\tilde{C}_b$ are in $\mathbf{NC}^0$ that,

$$\left\{(\tilde{C}_1, \tilde{C}_2, z) \xleftarrow{\$} D'_s \ : \ (\tilde{C}_1, \tilde{C}_2, pi\mathcal{O}_{\mathbf{NC}^0}(1^s, \tilde{C}_1), z)\right\}_s \approx \left\{(\tilde{C}_1, \tilde{C}_2, z) \xleftarrow{\$} D'_s \ : \ (\tilde{C}_1, \tilde{C}_2, pi\mathcal{O}_{\mathbf{NC}^0}(1^s, \tilde{C}_2), z)\right\}_s$$

By construction of $pi\mathcal{O}_{\mathbf{P}/poly}$, this implies Equation (4) and concludes the security of $pi\mathcal{O}_{\mathbf{P}/poly}$. □

References

[ABF+13] 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.

[ABG+13] 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/.

[AIK04] Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. Cryptography in NC^0. In *FOCS*, pages 166–175. IEEE Computer Society, 2004.

[AIK06] Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. Computationally private randomizing polynomials and their applications. *Computational Complexity*, 15(2):115–162, 2006.

[App13] Benny Applebaum. Bootstrapping obfuscators via fast pseudorandom functions. *IACR Cryptology ePrint Archive*, 2013:699, 2013.

[BCC+14] 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.

[BCP14] Elette Boyle, Kai-Min Chung, and Rafael Pass. On extractability obfuscation. In *TCC*, pages 52–73, 2014.

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

[BGI14] 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.

[BGK+14] 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.

[BGT14] Nir Bitansky, Sanjam Garg, and Sidharth Telang. Succinct randomized encodings and their applications. Cryptology ePrint Archive, Report 2014/771, 2014. http://eprint.iacr.org/.

[BGV12] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (leveled) fully homomorphic encryption without bootstrapping. In *ITCS*, pages 309–325, 2012.

[BHY09] 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.

[BLMR13] Dan Boneh, Kevin Lewi, Hart William Montgomery, and Ananth Raghunathan. Key homomorphic prfs and their applications. In Ran Canetti and Juan A. Garay, editors, *CRYPTO (1)*, volume 8042 of *Lecture Notes in Computer Science*, pages 410–428. Springer, 2013.

[BR13] 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.

[Bra12] Zvika Brakerski. Fully homomorphic encryption without modulus switching from classical gapsvp. In *CRYPTO*, pages 868–886, 2012.

[BST14] 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.

[BV11] 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.

[BW13] 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.

[DJ01] 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.

[Gen09] Craig Gentry. Fully homomorphic encryption using ideal lattices. In *STOC*, pages 169–178, 2009.

[GGH+13] 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.

[GGHW14] Sanjam Garg, Craig Gentry, Shai Halevi, and Daniel Wichs. On the implausibility of differing-inputs obfuscation and extractable witness encryption with auxiliary input. In *Advances in Cryptology - CRYPTO 2014 - 34th Annual Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2014, Proceedings, Part I*, pages 518–535, 2014.

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

[GHV10] Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. -hop homomorphic encryption and rerandomizable yao circuits. In Tal Rabin, editor, *CRYPTO*, volume 6223 of *Lecture Notes in Computer Science*, pages 155–172. Springer, 2010.

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

[GM84] Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *J. Comput. Syst. Sci.*, 28(2):270–299, 1984.

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

[HLOV11] 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.

[IK00] 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.

[KN08] Gillat Kol and Moni Naor. Games for exchanging information. In *STOC*, pages 423–432, 2008.

[KPTZ13] 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.

[LP14] Huijia Lin and Rafael Pass. Succinct garbling schemes and applications. Cryptology ePrint Archive, Report 2014/766, 2014. http://eprint.iacr.org/.

[Pai99] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *EUROCRYPT*, pages 223–238, 1999.

[PR07] Manoj Prabhakaran and Mike Rosulek. Rerandomizable RCCA encryption. In *CRYPTO*, pages 517–534, 2007.

[PVW08] Chris Peikert, Vinod Vaikuntanathan, and Brent Waters. A framework for efficient and composable oblivious transfer. In *CRYPTO*, pages 554–571, 2008.

[PW11] Chris Peikert and Brent Waters. Lossy trapdoor functions and their applications. *SIAM J. Comput.*, 40(6):1803–1844, 2011.

[SW14] 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.

A Preliminaries

We here define cryptographic primitives used in this paper. For simplicity, we only provide definitions of their security w.r.t. PPT adversaries and, conventional, negligible advantage. In the paper, we sometimes need sub-exponential security (or μ-security for a small function μ) of these primitives, which are defined in the same way w.r.t. *still PPT adversaries*, but sub-exponentially small advantage (or advantage bounded by μ). For example, a sub-exponentially indistinguishable (or μ-indistinguishable) PRF is one whose output is sub-exponentially indistinguishable (or μ-indistinguishable) from random, and a sub-exponentially indistinguishable IO is one, such that, obfuscation of functionally equivalent circuits are sub-exponentially indistinguishable. We omit the formal definitions of sub-exponential security for individual primitives.

A.1 Puncturable Pseudo-Random Functions

We recall the definition of puncturable pseudo-random functions (PRF) from [SW14]. Since in this work, we only uses puncturing at one point, the definition below is restricted to puncturing only at one point instead of at a polynomially many points.

Definition A.1 (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 [BW13, BGI14, KPTZ13], the GGM tree-based construction of PRFs [GGM86] from pseudorandom generators (PRGs) yields puncturable PRFs. Furthermore, it is easy to see that 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.

A.2 Randomized Encoding

Below we recall the definition of a *randomized encoding* for efficiently computatble functions from [AIK04, AIK06]. We focus on the computational case.

Definition A.2 ((Computational) Randomized Encoding)**.** *Let* $f : \{0,1\}^* \to \{0,1\}^*$ *be a polynomial time computable function, and* $l(n)$ *an output length function such that* $|f(x)| = l(|x|)$ *for every* $x \in \{0,1\}^*$*. We say that* $\hat{f} : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^*$ *is a randomized encoding of* f*, if there exists a negligible function* μ*, such that, the following holds:*

Length Regularity: *There exists polynomially-bounded and efficiently computable functions* $m(n)$*,* $s(n)$*, such that for every* $x \in \{0,1\}^n$ *and* $r \in \{0,1\}^{m(n)}$*, we have* $|\hat{f}(x, r)| = s(n)$*.*

Efficient Evaluation: *There exists a polynomial-time evaluation algorithm that on input* $x \in \{0,1\}^*$ *and* $r \in \{0,1\}^{m(|x|)}$ *outputs* $\hat{f}(x,r)$.

Correctness: *There exists a polynomial-time algorithm* Decode*, called a decoder, such that, for every input* $x \in \{0,1\}^n$, $\Pr[\mathsf{Decode}(\hat{f}(x, U_{m(n)})) \neq f(x)] \leq \mu(n)$.

Privacy: *There exists a randomized polynomial-time algorithm* S*, called a simulator satisfying that for every non-uniform PPT adversary* $\mathcal{A}$*, there is a negligible function* μ' *such that for every input* $x \in \{0,1\}^n$ *and auxiliary input* $z \in \{0,1\}^{\mathrm{poly}(n)}$,

$$\left| \Pr[\mathcal{A}(S(1^n, f(x)), z) = 1] - \Pr[\mathcal{A}(\hat{f}(x, U_{m(n)}), z) = 1] \right| \leq \mu'(n)$$

.

Definition A.3 ((Uniform) Randomized Encoding Scheme from $\mathbf{P}/poly$ to $\mathbf{NC}^0$)**.** *We say that a pair of uniform polynoimial time algorithms* $(\mathsf{RE}, \mathsf{Decode})$ *is a randomized encoding scheme for* $\mathbf{P}/poly$ *in* $\mathbf{NC}^0$*, if the following holds:*

- *There is a polynomial* s *and constant* c*, such that, for every* $n \in \mathbb{N}$ *and deterministic circuit* $C(x)$ *of size at most* n*, the randomized function* $\mathsf{RE}(1^n, C, x; r)$ *is a randomized encoding of* C *w.r.t. decoder* Decode *as in Definition A.2. Moreover,* $\mathsf{RE}(1^n, C, x; r)$ *has size* $s(n)$ *and depth* c.

It was shown in [AIK06] that a randomized encoding scheme for $\mathbf{P}/poly$ in $\mathbf{NC}^0$ exsits based on the existence of pseudo-random generator (PRG) in $\oplus\mathbf{L}/poly$, which in turn can be based on, for instance, the intractability of factoring and the learning with error (LWE) problems.

A.3 Homomorphic Encryption – Definitions

We now define homomorphic encryption and its desired properties. Throughout this section (and this work) we use λ to indicate the security parameter. Below we provide our definition for bit encryption; the generalization to an arbitrary message space is immediate.

A homomorphic (public-key) encryption scheme $\mathsf{HE} = (\mathsf{HE.Keygen}, \mathsf{HE.Enc}, \mathsf{HE.Dec}, \mathsf{HE.Eval})$ is a quadruple of PPT algorithms as follows.

- **Key generation.** The algorithm $(\mathsf{pk}, \mathsf{evk}, \mathsf{sk}) \leftarrow \mathsf{HE.Keygen}(1^\lambda)$ takes a unary representation of the security parameter and outputs a public encryption key pk, a public evaluation key evk and a secret decryption key sk.
- **Encryption.** The algorithm $c \leftarrow \mathsf{HE.Enc}_{\mathsf{pk}}(\mu)$ takes the public key pk and a single bit message $\mu \in \{0,1\}$ and outputs a ciphertext c.
- **Decryption.** The algorithm $\mu^* \leftarrow \mathsf{HE.Dec}_{\mathsf{sk}}(c)$ takes the secret key sk and a ciphertext c and outputs a message $\mu^* \in \{0,1\}$.
- **Homomorphic evaluation.** The algorithm $c_f \leftarrow \mathsf{HE.Eval}_{\mathsf{evk}}(f, c_1, \ldots, c_\ell)$ takes the evaluation key evk, a function $f : \{0,1\}^\ell \to \{0,1\}$ and a set of ℓ ciphertexts $c_1, \ldots, c_\ell$, and outputs a ciphertext c_f.

  The representation of the function f is an important issue. Since the representation can vary between schemes, we leave this issue outside of this syntactic definition. In this work, we will focus on the circuits consisting of only NAND gates.

The only security notion we consider in this work is semantic security, namely security w.r.t. passive adversaries. We use its widely known formulation as IND-CPA security, defined as follows.

Definition A.4 (CPA security)**.** *A scheme* HE *is IND-CPA secure if for any polynomial time adversary* $\mathcal{A}$ *it holds that*

$$\mathrm{Adv}_{\mathrm{CPA}}[\mathcal{A}] := |\Pr[\mathcal{A}(\mathsf{pk}, \mathsf{evk}, \mathsf{HE.Enc}_{\mathsf{pk}}(0)) = 1] - \Pr[\mathcal{A}(\mathsf{pk}, \mathsf{evk}, \mathsf{HE.Enc}_{\mathsf{pk}}(1)) = 1]| = \mathsf{negl}(\lambda) \ ,$$

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

We move on to define the homomorphism property. Note that we do not define the "correctness" of the scheme as a separate property, but rather (some form of) correctness will follow from our homomorphism properties.

We start by defining $\mathcal{C}$-homomorphism, which is homomorphism with respect to a specified class $\mathcal{C}$ of functions. This notion is sometimes also referred to as "somewhat homomorphism".

Definition A.5 ($\mathcal{C}$-homomorphism)**.** *Let* $\mathcal{C} = \{\mathcal{C}_\lambda\}_{\lambda \in \mathbb{N}}$ *be a class of functions (together with their respective representations). A scheme* HE *is* $\mathcal{C}$*-homomorphic (or, homomorphic for the class* $\mathcal{C}$*) if for any sequence of functions* $f_\lambda \in \mathcal{C}_\lambda$ *and respective inputs* $\mu_1, \ldots, \mu_\ell \in \{0,1\}$ *(where* $\ell = \ell(\lambda)$*), it holds that*

$$\Pr\left[\mathsf{HE.Dec}_{\mathsf{sk}}(\mathsf{HE.Eval}_{\mathsf{evk}}(f, c_1, \ldots, c_\ell)) \neq f(\mu_1, \ldots, \mu_\ell)\right] = \mathsf{negl}(\lambda) \ ,$$

where $(\mathsf{pk}, \mathsf{evk}, \mathsf{sk}) \leftarrow \mathsf{HE.Keygen}(1^\lambda)$ *and* $c_i \leftarrow \mathsf{HE.Enc}_{\mathsf{pk}}(\mu_i)$.

We point out two important properties that the above definition *does not* require. First of all, it does not require that the ciphertexts c_i are decryptable themselves, only that they become decryptable after homomorphic evaluation.[5] Secondly, it does not require that the output of $\mathsf{HE.Eval}$ can undergo additional homomorphic evaluation.[6]

Next, we define full homomorphism and its relaxation. Before that, we first introduce the notion of *compactness*.

Definition A.6 (compactness)**.** *A homomorphic scheme* HE *is compact if there exists a polynomial* $s = s(\lambda)$ *such that the output length of* $\mathsf{HE.Eval}(\cdots)$ *is at most* s *bits long (regardless of* f *or the number of inputs).*

Note that a $\mathcal{C}$-homomorphic scheme is not necessarily compact. We now provide a minimal definition of a fully homomorphic encryption.

Definition A.7 (fully homomorphic encryption)**.** *A scheme* HE *is fully homomorphic if it is both compact and homomorphic for the class of all circuits consisting of only* NAND *gates.*

An important relaxation of fully homomorphic encryption is the following.

Definition A.8 (leveled fully homomorphic encryption)**.** *A leveled fully homomorphic encryption scheme is a homomorphic scheme where the* $\mathsf{HE.Keygen}$ *gets an additional input* 1^L *(now* $(\mathsf{pk}, \mathsf{evk}, \mathsf{sk}) \leftarrow \mathsf{HE.Keygen}(1^\lambda, 1^L)$*) and the resulting scheme is homomorphic for all depth-*L *binary arithmetic circuits. The bound* $s(\lambda)$ *on the ciphertext length must remain independent of* L.

[5] Jumping ahead, while this may seem strange at first, this notion of somewhat homomorphism is all that is really required in order to bootstrap into full homomorphism and it also makes our schemes easier to describe. Lastly, note that one can always perform a "blank" homomorphic operation and then decrypt, so functionality is not hurt.

[6] This is termed "1-hop homomorphism" in [GHV10].

In this work, we show that a leveled fully homomorphic encryption scheme can be constructed from a piO obfuscator and a vanilla CPA-secure public key encryption scheme. Similar to previous constructions of leveled fully homomorphic encryption schemes, the only parameter of our scheme that becomes dependent on L is the bit-length of the evaluation key evk.

B Missing Proofs from the Body

B.1 Proof of Proposition 2.3

Proof. Indeed, assume that there exists $D \in \mathbf{S}^{X\text{-}\mathsf{Ind}}$ such that $D \notin \mathbf{S}^{\mathsf{mw}\text{-}\mathsf{Ind}}$. In particular, this means that there exists a non-uniform memory-less worst-case-input adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, where $\mathcal{A}_1$ is computationally unbounded and $\mathcal{A}_2$ is computationally bounded, and a polynomial p, such that for infinitely many values λ, $\mathcal{A}$ achieves non-negligible advantage $1/p(\lambda)$ in Experiment worst-case-input-$\mathsf{IND}^D_{\mathcal{A}}(1^\lambda)$. As we are in the non-uniform setting, we can assume without loss of generality that $\mathcal{A}$ is deterministic. In this proof, we are going to construct a new non-uniform adversary $\mathcal{A}' = (\mathcal{A}'_1, \mathcal{A}'_2)$ which achieves advantage $\frac{1}{16p^2(\lambda)X(\lambda)}$ in Experiment static-input-$\mathsf{IND}^D_{\mathcal{A}'}(1^\lambda)$ for infinitely many λ, which contradicts the fact that $D \in \mathbf{S}^{X\text{-}\mathsf{Ind}}$.

Fix λ for which $\mathcal{A}$ achieves advantage $1/p(\lambda)$. As $D \in \mathbf{S}^{X\text{-}\mathsf{Ind}}$, let $\mathcal{X}_\lambda$ be the corresponding differing domain. The adversary $\mathcal{A}' = (\mathcal{A}'_1, \mathcal{A}'_2)$ operates as follows, for security parameter λ:

- The adversary $\mathcal{A}'_1$ outputs a random string x^* from the differing domain $\mathcal{X}_\lambda$, and passes empty state $st = \perp$ to $\mathcal{A}'_2$. (Note that the sampling may make $\mathcal{A}'_1$ not necessarily polynomial-time, but the resulting adversary can be *de-randomized* without loss of generality, and the sampling can be replaced by the optimal choice of x^*.)
- Given $(\perp, C_0, C_1, x^*, y^* = C_b(x^*), z)$, the adversary $\mathcal{A}'_2$ first tests the performance of $\mathcal{A}_2$ by running multiple independent copies of $\mathcal{A}_2$ on input $(C_0, C_1, z, x^*, C_{b^*_i}(x^*))$ for fresh independent bits $b^*_1, \ldots, b^*_k$ and freshly sampled $C_{b^*_i}(x^*)$, where $k = \mathrm{poly}(\lambda)$ is appropriately set to be large enough. Each one of these copies outputs a guess b'_i, and if the fraction of correct guesses $b'_i = b^*_i$ is at least $\frac{1}{2} + \frac{1}{4p(\lambda)}$, then $\mathcal{A}'_2$ runs $\mathcal{A}_2(\perp, C_0, C_1, x^*, y^*, z)$ and outputs the corresponding bit. Otherwise, $\mathcal{A}'_2$ outputs a random bit.

Note that $\mathcal{A}_1$ (which is computationally unbounded) is *never* run by the reduction. Since $\mathcal{A}_1$ is deterministic, for every C_0, C_1, z, there exists a unique $x = x_{C_0,C_1,z}$ which is output by $\mathcal{A}_1$ on input C_0, C_1, z. Also, we call a triple (C_0, C_1, z) *good* if $\mathcal{A}_2(C_0, C_1, x = x_{C_0,C_1,z}, C_b(x), z)$, for a random bit b, guesses the bit with probability at least $\frac{1}{2} + \frac{1}{2p(\lambda)}$. (Here, probability is over the computation of $C_b(x)$, as $\mathcal{A}_2$ is deterministic.) Note that the probability that D_λ samples a good triple is at least $\frac{1}{2p(\lambda)}$ by a Markov argument. Finally, note that for a good triple C_0, C_1, z, it must be that $x_{C_0,C_1,z} \in \mathcal{X}$, as otherwise there is no way that $\mathcal{A}_2$ could ever achieve a positive advantage over $\frac{1}{2}$.

Now, let us turn to the analysis of $\mathcal{A}' = (\mathcal{A}'_1, \mathcal{A}'_2)$ in Experiment static-input-$\mathsf{IND}^D_{\mathcal{A}'}(1^\lambda)$. We first note that $\mathcal{A}'_2$ – using $\mathcal{A}_2$ – is going to always guess the right bit with probability at least $\frac{1}{2} - \frac{1}{16p(\lambda)}$ on any given input. This is because if $\mathcal{A}_2(\perp, C_0, C_1, x^*, y^*, z)$ succeeds with probability less than $\frac{1}{2}$, than we are going to detect this (and take a random guess) except with probability $\frac{1}{8p(\lambda)}$ (by choosing $k = \Omega(p(\lambda)^2 \log p(\lambda))$ and using the Chernoff bound), and thus overall, the guess is going to be correct with probability at least $\frac{1}{2} - \frac{1}{16p(\lambda)}$.

Then, for a triple $(C_0, C_1, z) \xleftarrow{\$} D_\lambda$, consider the event that $x_{C_0,C_1,z} = x^*$, and (C_0, C_1, z) is good. Note that this occurs with probability at least $\frac{1}{2p(\lambda)X}$. Then, in this case, $\mathcal{A}_2$ guesses correctly with probability at least $\frac{1}{2} + \frac{1}{2p(\lambda)}$, and the probability that this is not *detected* by $\mathcal{A}'_2$ can be made as small as $\frac{1}{4p(\lambda)}$ by an appropriate choice of $k = \mathrm{poly}(\lambda)$, using the Chernoff bound. Thus when (C_0, C_1, z) is good, $\mathcal{A}'_2$ always guesses with probability at least $\frac{1}{2} + \frac{1}{4p(\lambda)}$

Putting things together, in Experiment static-input-$\mathsf{IND}^D_{\mathcal{A}'}(1^\lambda)$, with b' denoting the output of $\mathcal{A}'$, we have, using $X = X(\lambda)$,

$$\Pr[b = b'] \geq \frac{1}{2p(\lambda)X} \cdot \left(\frac{1}{2} + \frac{1}{4p(\lambda)}\right) + \left(1 - \frac{1}{2p(\lambda)X}\right)\left(\frac{1}{2} - \frac{1}{16p(\lambda)}\right) \geq \frac{1}{2} + \frac{1}{16p(\lambda)^2 X} ,$$

which concludes the proof. □

B.2 Proof of Theorem 2.5

Next we provide the complete proof of Theorem 2.5 showing that X-pIO can be constructed from sub-exponentially secure IO and OWFs. For completeness, we repeat the construction in the proof of the theorem below.

Proof. We first present the construction of X-$pi\mathcal{O}$ and then analyze its correctness and security. Recall that by our assumption, both $i\mathcal{O}$ and the puncturable PRF $(\mathrm{Key}, \mathrm{Puncture}, \mathsf{PRF})$ 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 $E^{(C,K)}$ as described in Figure 2. 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 $i\mathcal{O}$, $\Lambda \xleftarrow{\$} i\mathcal{O}(1^{\lambda''}, E^{(C,K)})$.
4. Output Λ.

Correctness of X-$pi\mathcal{O}$: It is easy to see that the output circuit Λ has size polynomial in λ and $|C|$. Next, we show that no distinguisher can tell apart whether it is receiving outputs of a probabilistic circuit C or output of the obfuscated circuit $\Lambda \xleftarrow{\$} X$-$pi\mathcal{O}(1^\lambda, C)$. Towards this, fix any distinguisher $\mathcal{D}$, $\lambda \in \mathbb{N}$ and C with $|C| \leq \lambda$. We show that the advantage of $\mathcal{D}$ in distinguishing experiments $\mathsf{Exp}^1_{\mathcal{D}}(1^\lambda, C, z)$ (where it receives outputs of C) and $\mathsf{Exp}^2_{\mathcal{D}}(1^\lambda, C, z)$ (where it receives outputs of Λ) is negligible. Consider the hybrid experiment $\widetilde{\mathsf{Exp}}_{\mathcal{D}}(1^\lambda, C, z)$: A random key K of the PRF is sampled; whenever $\mathcal{D}(1^\lambda, C, z)$ chooses an input x_i (that is different from all previous inputs), it is fed with the output $E^{(C,K)}(x_i)$. It follows from the security of PRF function that the view of $\mathcal{D}$ in $\mathsf{Exp}^1_{\mathcal{D}}$ and $\widetilde{\mathsf{Exp}}_{\mathcal{D}}$ are indistinguishable, since distinct PRF inputs yield outputs which are indistinguishable from independently sampled random coins. Furthermore, it follows from the

correctness of the indistinguishability obfuscator $i\mathcal{O}$ that the obfuscated circuit Λ computes the same functionality as $E^{(C,K)}$, thus the view of $\mathcal{D}$ in $\mathsf{Exp}^2_{\mathcal{D}}$ and $\widetilde{\mathsf{Exp}}_{\mathcal{D}}$ are identical. Therefore, we conclude that the advantage of $\mathcal{D}$ in $\mathsf{Exp}^1_{\mathcal{D}}$ and $\mathsf{Exp}^2_{\mathcal{D}}$ is negligible.

Security of X-$pi\mathcal{O}$: Fix a X-Ind sampler $D \in \mathbf{S}^{X\text{-}\mathsf{Ind}}$. We need to prove that for every non-uniform PPT machine $\mathcal{A}$, there exists a negligible function μ such that for all $\lambda \in \mathbb{N}$,

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

Now, assume instead towards a contradiction that there exists a polynomial p such that $\Delta(\lambda) \geq 1/p(\lambda)$ for infinitely many $\lambda \in \mathbb{N}$. Let us fix such $\lambda \in \mathbb{N}$, and in particular, let us fix $\mathcal{X} = \mathcal{X}_\lambda$ as in the Definition of X-Ind samplers. Also, let $<$ be some ordering of the strings, and enumerate the X elements of $\mathcal{X}$ as $x_1 < x_2 < \cdots < x_X$. Recall that $X \leq 2^\lambda$, as the circuits have size at most λ.

We now consider a series of hybrids H'_1, $\left\{H^0_i, H^1_i, \cdots, H^4_i\right\}_{0 \leq i < X}$, H^0_X, H'_2, where in each hybrid the adversary $\mathcal{A}$ receives the obfuscation of a different circuit: In H'_1 it receives the honestly generated obfuscation of C_1, and in H'_2 it receives that of C_2; in intermediate hybrids, it receives the obfuscation of "hybrid circuits" that evaluates part of its inputs using C_1 and the rest using C_2. Let $\mathsf{hyb}^j_i(1^\lambda)$ be the output of $\mathcal{A}$ in hybrid H^j_i and $\mathsf{hyb}'_i(1^\lambda)$ that in H'_i. The hybrids are formally defined below:

Hybrid H'_1, H'_2: In H'_i, sample a tuple $(C_1, C_2, z) \xleftarrow{\$} D_\lambda$ and a PRF key $K \xleftarrow{\$} \mathrm{Key}(1^{\lambda'})$; obfuscate the circuit $E^{(C_i,K)}$ (described in Figure 2) to obtain $\Lambda'_i \xleftarrow{\$} i\mathcal{O}(1^{\lambda''}, E^{(C_i,K)}_i)$. Finally, Output $\mathcal{A}(C_1, C_2, \Lambda'_i, z)$. By definition of H'_i, we have

$$\Delta = \big|\Pr[\mathsf{hyb}'_1(1^\lambda) = 1] - \Pr[\mathsf{hyb}'_2(1^\lambda) = 1]\big| \geq \frac{1}{p(\lambda)} \ .$$

Hybrid H^0_i: H^0_i proceeds identically to H'_1, H'_2, except that the circuit $E^{(C_1,C_2,K)}_i$ (as described in Figure 2) instead of $E^{(C_iK)}$ is obfuscated using $i\mathcal{O}$.

Recall that circuit $E^{(C_1,C_2K)}_i$ evaluates the first inputs $x < x_i$ using circuit C_2, and the rest $x \geq x_i$ using circuit C_1 (both are fed with pseudo-random coins $\mathsf{PRF}(K, x)$). By construction, $E^{(C_1,C_2,K)}_0$ computes the same function as $E^{(C_1,K)}$, and $E^{(C_1,C_2,K)}_X$ computes the same function as $E^{(C_2,K)}$. Thus by the fact that the indistinguishability obfuscator $i\mathcal{O}$ has $2^{-\lambda''^\epsilon}$-distinguishing gap for security parameter λ'' and that $\lambda'' \geq \lambda' = (\lambda \log^2 \lambda)^{1/\epsilon}$, we have that:

$$\big|\Pr[\mathsf{hyb}'_1(1^\lambda) = 1] - \Pr[\mathsf{hyb}^0_0(1^\lambda) = 1]\big| \leq \frac{1}{X 2^{\log^2 \lambda}}$$
$$\big|\Pr[\mathsf{hyb}'_2(1^\lambda) = 1] - \Pr[\mathsf{hyb}^0_X(1^\lambda) = 1]\big| \leq \frac{1}{X 2^{\log^2 \lambda}}$$

Hybrid H^1_i: H^1_i proceeds identically as H^0_i except that, H^1_i additionally generates a key K_{-i} that is punctured at input i (i.e., $K_{-i} = \mathrm{Puncture}(K, i)$) and replaces the circuit $E^{(C_1,C_2,K)}_i$ with $E^{(C_1,C_2,K,y^1_i)}_i$ for $y^1_i = C_1(x_i \ ; \ \mathsf{PRF}(K, x_i))$.

We note that for every $0 \leq i < X$, circuits $E_i^{(C_1,C_2,K)}$ and $E_i^{(C_1,C_2,K,y_i^1)}$ computes the same function. This follows from the fact that the punctured key K_{-i} computes the same pseudo-random coins for every input $x \neq i$, and for input $x = i$, $E_i^{(C_1,C_2,K,y_i^1)}$ outputs y_i^1 which is exactly the same output $E_i^{(C_1,C_2,K)}$ produces on input $x = i$. Thus as argued above, it follows from the indistinguishability of $i\mathcal{O}$ that, for every $0 \leq i < X$,

$$\left| \Pr[\mathsf{hyb}_i^0(1^\lambda) = 1] - \Pr[\mathsf{hyb}_i^1(1^\lambda) = 1] \right| \leq \frac{1}{X 2^{\log^2 \lambda}}$$

Hybrid H_i^2: H_i^2 proceeds identically as H_i^1 except that, it uses circuit $E_i^{(C_1,C_2,K,y_i^2)}$ with value $y_i^2 \xleftarrow{\$} C_1(x_i \ ; \ r^*)$, produced using truly random coins r^*.

It follows from the pseudo-randomness of the puncturable PRF that the pseudo-random coins $\mathsf{PRF}(K, x_i)$ is indistinguishable from truly random coins $r*$, even if punctured key K_{-i} is public. Furthermore, since the distinguishing gap of the PRF function is $2^{-\lambda'^\epsilon}$ for security parameter $\lambda' = (\lambda \log^2 \lambda)^{1/\epsilon}$, we have:

$$\left| \Pr[\mathsf{hyb}_i^1(1^\lambda) = 1] - \Pr[\mathsf{hyb}_i^2(1^\lambda) = 1] \right| \leq \frac{1}{X 2^{\log^2 \lambda}}$$

Hybrid H_i^3: H_i^3 proceeds identically as H_i^2 except that, it uses circuit $E_i^{(C_1,C_2,K,y_i^3)}$ with value $y_i^3 = C_2(x_i \ ; \ r^*)$, produced by evaluating C_2 instead of C_1 (still with truly random coins). Then, we define

$$\mathsf{Gap}(i, \lambda) := \left| \Pr[\mathsf{hyb}_i^2(1^\lambda) = 1] - \Pr[\mathsf{hyb}_i^3(1^\lambda) = 1] \right| .$$

Hybrid H_i^4: H_i^4 proceeds identically as H_i^3 except that, it uses circuit $E_i^{(C_1,C_2,K,y_i^4)}$ with value $y_i^4 = C_2(x_i \ ; \ \mathsf{PRF}(K, x_i))$, produced using pseudo-random coins (still evaluating C_2).

It follows from the same argument as in H_i^2 that by the pseudo-randomness of the puncturable PRF function, we have:

$$\left| \Pr[\mathsf{hyb}_i^3(1^\lambda) = 1] - \Pr[\mathsf{hyb}_i^4(1^\lambda) = 1] \right| \leq \frac{1}{X 2^{\log^2 \lambda}}$$

Hybrid H_{i+1}^0: H_{i+1}^0 proceeds according to the description of H_i^0 above but using $i + 1$ instead. Comparing with H_i^4, it differs in that the circuit $E_{i+1}^{(C_1,C_2,K,y)}$ is used instead of $E_i^{(C_1,C_2,K,y_i^4)}$ with $y_i^4 = C_2(x_i \ ; \ \mathsf{PRF}(K, x_i))$.

It follows from the same argument as in H_i^1 that by the indistinguishability of $i\mathcal{O}$, we have:

$$\left| \Pr[\mathsf{hyb}_i^4(1^\lambda) = 1] - \Pr[\mathsf{hyb}_{i+1}^0(1^\lambda) = 1] \right| \leq \frac{1}{X 2^{\log^2 \lambda}}$$

We can use the above inequality, repeatedly applying the triangle inequality, to derive the following lower bound

$$\begin{aligned}
\sum_{i=0}^{X-1} \mathsf{Gap}(i,\lambda) &= \sum_{i=0}^{X-1} \left| \Pr[\mathsf{hyb}_i^2(1^\lambda)=1] - \Pr[\mathsf{hyb}_i^3(1^\lambda)=1] \right| \\
&\geq \left(\sum_{i=0}^{X-1} \left| \Pr[\mathsf{hyb}_i^1(1^\lambda)=1] - \Pr[\mathsf{hyb}_i^4(1^\lambda)=1] \right| \right) - \frac{2}{2^{\log^2 \lambda}} \\
&\geq \left(\sum_{i=0}^{X-1} \left| \Pr[\mathsf{hyb}_i^0(1^\lambda)=1] - \Pr[\mathsf{hyb}_{i+1}^0(1^\lambda)=1] \right| \right) - \frac{4}{2^{\log^2 \lambda}} \\
&\geq \left| \Pr[\mathsf{hyb}_0^0(1^\lambda)=1] - \Pr[\mathsf{hyb}_X^0(1^\lambda)=1] \right| - \frac{4}{2^{\log^2 \lambda}} \\
&\geq \left| \Pr[\mathsf{hyb}_1'(1^\lambda)=1] - \Pr[\mathsf{hyb}_2'(1^\lambda)=1] \right| - \frac{4X+2}{X2^{\log^2 \lambda}} \geq \frac{1}{p(\lambda)} - \frac{4X+2}{X2^{\log^2 \lambda}} \geq \frac{1}{2p(\lambda)}
\end{aligned}$$

for sufficiently large λ.

Thus for every sufficiently large λ, there must exist one input $i = i_\lambda$ for which $\mathsf{Gap}(i_\lambda, \lambda) \geq \frac{1}{2Xp(\lambda)}$. We are now going to build a static adversary $(\mathcal{A}_1, \mathcal{A}_2)$ in Experiment static-input-$\mathsf{IND}_{\mathcal{A}}^D(1^\lambda)$ achieving distinguishing gap at least $\frac{1}{4Xp(\lambda)}$, and hence contradicting the fact that $D \in \mathbf{S}^{X\text{-}\mathsf{Ind}}$.

The non-uniform PPT machine $\mathcal{A}_1$ has $x_i = x_{i_\lambda}$ hard-wired in and this is the value which is output, together with state $st = \perp$. Then, on input $(\perp, C_1, C_2, z, x_i, y)$, $\mathcal{A}_2$ emulates the execution of $\mathcal{A}$ in H_i^2 or H_i^3 with the difference that it uses the supplied value y, and obfuscates $E^{(C_1,C_2,K,y)}$; and finally, $\mathcal{A}_2$ returns the output of $\mathcal{A}$. By construction of $\mathcal{A}_2$, for every $b \in \{1,2\}$, when (C_1, C_2, z) are sampled from D and y computed by evaluating $C_b(x_i \ ; \ r^*)$ using fresh random coins, $\mathcal{A}_2$ emulates the view of $\mathcal{A}$ in H_i^{b+1} perfectly. Thus, we have:

$$\begin{aligned}
\Big| \Pr[(C_1,C_2,z) \xleftarrow{\$} D_\lambda \ : \ \mathcal{A}_2(\perp, C_1, C_2, z, x_i, C_1(x_i)) = 1] \\
- \Pr[(C_1,C_2,z) \xleftarrow{\$} D_\lambda \ : \ \mathcal{A}_2(\perp, C_1, C_2, z, x_i, C_2(x_i)) = 1] \Big| = \mathsf{Gap}(i_\lambda, \lambda) \geq \frac{1}{2Xp(\lambda)}.
\end{aligned}$$

This in particular implies that the advantage over $\frac{1}{2}$ is $\frac{1}{2Xp(\lambda)}$, contradicting the fact that $D \in \mathbf{S}^{X\text{-}\mathsf{Ind}}$. □