Obfuscation from Polynomial Hardness: Beyond Decomposable Obfuscation

Yuan Kang[1] *, Chengyu Lin[1], Tal Malkin[1] **, and Mariana Raykova[2]***

[1] Columbia University yuan.j.kang@gmail.com, {chengyu, tal}@cs.columbia.edu
[2] Yale University mariana.raykova@yale.edu

Abstract. Every known construction of general indistinguishability obfuscation ($\mathsf{i}\mathcal{O}$) is either based on a family of exponentially many assumptions, or is based on a single assumption – e.g. functional encryption (FE) – using a reduction that incurs an exponential loss in security. This seems to be an inherent limitation if we insist on providing indistinguishability for any pair of functionally equivalent circuits.

Recently, Liu and Zhandry (TCC 2017) introduced the notion of decomposable $\mathsf{i}\mathcal{O}$ ($\mathsf{di}\mathcal{O}$), which provides indistinguishability for a restricted class of functionally equivalent circuit pairs, and, as the authors show, can be constructed from polynomially secure FE.

In this paper we propose a new notion of obfuscation, termed $\mathsf{radi}\mathcal{O}$ (repeated-subcircuit and decomposable obfuscation), which allows us to obfuscate a strictly larger class of circuit pairs using a polynomial reduction to FE. Our notion builds on the equivalence criterion of Liu and Zhandry, combining it with a new incomparable criterion to obtain a strictly larger class.

1 Introduction

Indistinguishability obfuscation ($\mathsf{i}\mathcal{O}$) provides a way to obfuscate a circuit in a way that preserves its functionality, but such that the obfuscated versions $\mathsf{i}\mathcal{O}(C_0)$ and $\mathsf{i}\mathcal{O}(C_1)$ for any two functionally equivalent circuits C_0 and C_1 are computationally indistinguishable. In the last several years, following the first candidate

* Work done while supported by Air Force Office of Scientific Research (AFOSR) grant FA9550-12-1-0162. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of AFOSR.

** Lin and Malkin are supported by NSF grants CNS-1445424 and CCF1423306, the Leona M. & Harry B. Helmsley Charitable Trust, the Defense Advanced Research Project Agency (DARPA) and Army Research Office (ARO) under Contract W911NF-15-C-0236.

*** Supported by NSF grants CNS-1633282, 1562888, 1565208, and DARPA SafeWare W911NF-15-C-0236,W911NF-16-1-0389.
Any opinions, findings and conclusions or recommendations expressed are those of the authors and do not necessarily reflect the views of the the Defense Advanced Research Projects Agency, Army Research Office, the National Science Foundation, or the U.S. Government.

construction by Garg, Gentry, Halevi, Raykova, Sahai, and Waters [GGH+13], $i\mathcal{O}$ has become a central cryptographic primitive, with many results demonstrating its extreme power and wide applicability in cryptography and beyond (cf. [SW14,BZ14,BPR15,BPW16] and many more).

Constructions of general $i\mathcal{O}$ from various security assumptions can be divided into two categories: constructions that rely on families of assumptions of exponential size, one per pair of functionally equivalent circuits [GGH+13,BGK+14,PST14], and constructions that incur exponential security loss in their proof reduction and thus require their underlying assumptions to provide sub-exponential hardness [GLSW15,BV15,AJ15,Lin16,LV16,LT17,AS17].

The most prominent example of the latter type of constructions is constructing $i\mathcal{O}$ from functional encryption (FE). The works of Bitansky and Vaikunthanatan [BV15] and Ananth and Jain [AJ15] provided the first reductions from $i\mathcal{O}$ to FE. While following papers have improved the requirements of compactness [AJS15,Lin16,LV16,AS17] and the public key properties for the starting functional encryption [BNPW16,KNT17], all known constructions of general $i\mathcal{O}$ still require subexponential security for the starting FE scheme.

Given this state of affairs, an obvious goal is to achieve a construction of $i\mathcal{O}$ from underlying primitives – such as FE – with *polynomial* security loss. However, as discussed in several previous works [GGSW13,GPSZ17,LZ17], this goal is likely unattainable for general $i\mathcal{O}$. The argument can be informally summarized as follows (see [LZ17] for a more comprehensive exposition and discussion).

Sub-Exponential Barrier for General Obfuscation. A security reduction proving indistinguishability obfuscation implicitly tests whether the two circuits C_0 and C_1 are functionally equivalent: if they are, the reduction must go through, but if they are not, an adversary with a hard-coded input where the circuits differ can easily distinguish between the obfuscated circuits, and so the reduction will fail. Thus, an *efficient* reduction would seemingly yield an efficient verification of circuit equivalence, which in turn would imply that the polynomial hierarchy collapses. We conclude that an exponential security loss seems unavoidable for general $i\mathcal{O}$. Note, however, that this barrier does not hold if the given reduction only works for pairs of circuits in some class where equivalence is efficiently verifiable (namely the language of circuit pairs is in NP).

With this barrier for general circuits in mind, and following the recent work of Liu and Zhandry [LZ17] (discussed below), in this paper we focus on the following goal:

> *Reduce* $i\mathcal{O}$ *to* FE *incurring only* polynomial security loss*, for as large a class of circuits as possible.*

This question is interesting not only as a goal on its own (namely, poly-secure obfuscation for a restricted class), but also as a tool for other potential applications. Indeed, when considering various applications that use $i\mathcal{O}$ as a building block, the security proof for the constructed primitive relies on the security proof of the $i\mathcal{O}$. If we instantiate $i\mathcal{O}$ using general constructions from FE, the resulting schemes inherit the subexponential hardness requirement for the underlying

FE. However, this exponential security loss may not be inherent in the application, even if it is inherent for general i$\mathcal{O}$. This has been demonstrated by several works, which provide constructions of a primitive directly from FE, with polynomial security loss. Such applications include universal samplers and trapdoor permutations [GPSZ17], multi-key functional encryption [GS16], and proving the hardness for the complexity class PPAD [GPS16]. Roughly speaking, these works looked at the generic composition of FE-to-i$\mathcal{O}$ and i$\mathcal{O}$-to-application, and combined, improved, and optimized it for the specific application, in order to achieve polynomial security loss.

In a recent work, Liu and Zhandry [LZ17] introduce a new notion of obfuscation called *decomposable obfuscation* (d$\mathcal{O}$), which aims to abstract and unify the proof techniques from these works, and address the same goal as we do here (i$\mathcal{O}$ from FE with polynomial loss, for a restricted class of circuits). The security definition of d$\mathcal{O}$ requires further restrictions in addition to functional equivalence of the two circuits, enabling the authors to prove d$\mathcal{O}$ security from FE incurring only polynomial security loss. The authors also show that the d$\mathcal{O}$ notion can replace the use of i$\mathcal{O}$ in the above applications, providing a direct polynomial time reduction to the FE security (rather than the application specific tailoring in previous works). However, there are other applications of i$\mathcal{O}$ where a polynomial reduction to FE is not known, but where the general barrier we discussed does not necessarily apply. Liu and Zhandry leave open the problem of obfuscating larger classes of circuit pairs with only polynomially-hard primitives, which may open the way to polynomial security proofs for new applications of i$\mathcal{O}$.

1.1 Our Results

We present a new obfuscation construction from functional encryption that provides indistinguishability assuming only *polynomial* security for the underlying FE, for a *strictly larger* class than the one handled by decomposable obfuscation [LZ17].

We note that, similarly to [LZ17], if two circuits are functionally equivalent but do not belong to our class, our construction is still a secure i$\mathcal{O}$ if the underlying FE has subexponential security.

Below, we first provide a (very) rough description of the relevant aspects of previous constructions of obfuscation from FE, and in particular the restriction imposed by d$\mathcal{O}$ on two functionally equivalent circuits. We then outline our restriction (which is weaker, thus allowing more circuit pairs), and discuss its potential implications and open problems. A more detailed description is given in our technical overview in Section 1.3, with a formal summary after the introduction, leaving the detailed descriptions and proofs in the appendix.

For a circuit C on n-bit inputs, consider the depth-n binary tree, where a node at location $x_1 \dots x_i$ is thought of as corresponding to the partial circuit, or "fragment", resulting from hard-coding into C the values of the corresponding

The authors originally called this *exploding obfuscation*, and this is the term they use in the eprint version of their paper.

prefix. That is, the root node corresponds to C, its left child corresponds to C with the first variable set to 0, its right child corresponds to C with the first variable set to 1, and so on, with each leaf corresponding to a constant circuit (0 or 1) based on the evaluation of C on the n-bit input leading to that leaf. A "cover" of the tree is a set of nodes such that each leaf belongs to a unique subtree rooted at one of the cover nodes (for example, the root is a cover of size 1, and the set of leaves is a cover of size 2^n). With this terminology, it is easy to see that two circuits are equivalent if and only if their trees have identical leaves, which in turn happens if and only if their trees have any identical common cover.

From FE to i$\mathcal{O}$: Previous Work. In the original constructions of i$\mathcal{O}$ from FE [BV15,AJ15], the obfuscation of a circuit included a FE ciphertext, together with FE decryption keys for specific functions, allowing to evaluate the circuit in roughly the following way. The ciphertext corresponds to the root of the tree (the obfuscated circuit), and each evaluation of the FE decryption allows to obtain ciphertexts for each of the two children of that node. The evaluator uses one of them depending on the input, and continues to apply FE decryption along a path in the tree, until finally obtaining the value at the leaf, which is the output. The proof of security utilizes the fact the leaves are identical in C_0 and C_1, and so the proof hybrids can go through the entire tree starting from C_0 to the leaves, then go back up from the leaves to the root using C_1, showing indistinguishability of obfuscation of both circuits. This proof has exponentially many hybrids.

Decomposable Obfuscation Limitation. The d$\mathcal{O}$ notion puts an additional restriction that the pair of circuits has an identical common cover of *of polynomial size*. If we only require indistinguishability for circuit pairs satisfying this restriction, then the proof of security needs only develop the tree up to the common cover, at which point C_0 and C_1, partially evaluated up to that point, are identical circuits. This results in polynomially many hybrids, and thus can be achieved from FE with a polynomial reduction.

Our Work: Loosening the Limitation. Our result expands that of d$\mathcal{O}$, by taking advantage of structural similarities within different circuits on the tree, so we can consider common covers of *exponential* size, subject to some conditions.

We start by defining a notion we call *repeated-subcircuit exploding* i$\mathcal{O}$, denoted rescuei$\mathcal{O}$, which is incomparable to that of d$\mathcal{O}$. Specifically, we require that the two circuits have a common cover such that each of the trees, from root to the cover, satisfies: (1) there are only polynomially many different circuits assigned to all nodes on or above the cover (for an exponential cover, this implies that many of the nodes are assigned identical circuits); and (2) there is some "common structure" condition between the two trees. We will formalize the common structure later, but intuitively it captures the fact that the patterns of the identical circuits in each level of the tree, and the relations between parent and child on the tree, are the same in both trees.

While the first condition strictly generalizes the d$\mathcal{O}$ condition of a polynomial size common cover, the second condition adds an additional restriction that makes the rescuei$\mathcal{O}$ class incomparable to d$\mathcal{O}$.

To overcome the additional structural limitation we apply $\mathsf{d}\mathcal{O}$ on top of $\mathsf{rescuei}\mathcal{O}$. $\mathsf{d}\mathcal{O}$ is independent of partial evaluations before the tree cover. So if any such fragments violate the structural requirement of $\mathsf{rescuei}\mathcal{O}$, we can ignore them. On the other hand, $\mathsf{rescuei}\mathcal{O}$ lets us create a tree cover that includes a potentially-exponential number of input prefixes. With this composition, we can achieve obfuscation for a class *strictly larger* than the union of both individual notions. This is our final $\mathsf{radi}\mathcal{O}$ construction.

1.2 Potential Implications and Open Problems

We summarize the more detailed discussion in [LZ17], who posed interesting open problems, and discuss how our results fit in this context.

As explained above, the subexponential barrier for obfuscation applies to any class of equivalent circuit pairs for which verifying whether a pair of circuits is in the class is hard, even given a witness (which the reduction may get). That is, the barrier holds for any such class that is believed not to be in NP (for example, the general class of all equivalent circuit pairs, which is not in NP unless the polynomial hierarchy collapses). This suggests a way to bypass the barrier, by considering a subclass of equivalent circuit pairs which is in NP.

The notion of $\mathsf{d}\mathcal{O}$, proposed by Liu and Zhandry, follows this path. In fact, in their case, testing if two circuits are in the class is not only in NP, but even in P, as the authors show. This avoids the barrier and can replace $\mathsf{i}\mathcal{O}$ in several applications (hence getting polynomial security for those applications). However, the fact the class is in P hinders its applicability to other important applications of $\mathsf{i}\mathcal{O}$, such as getting public key encryption from private key encryption, deniable encryption, and NIZK. For those applications, the known proofs rely on indistinguishability of obfuscated pairs of circuits in a class that cannot possibly be efficiently tested without a witness, and so $\mathsf{d}\mathcal{O}$ could not be used in place of general $\mathsf{i}\mathcal{O}$. On the other hand, all these applications can be tweaked so that the pairs of circuits in the proof do in fact have a witness proving their equivalence.

One of the open problems proposed by Liu and Zhandry is thus to build $\mathsf{i}\mathcal{O}$ for a class of circuit pairs for which equivalence is efficiently verifiable (with a witness) but *not* efficiently testable (without a witness), and which can be based on polynomial hardness of a small number of assumptions (such as FE).

We show that our intermediate class of circuits, $\mathsf{rescuei}\mathcal{O}$, is efficiently testable, thus suffering from the same limitation as $\mathsf{d}\mathcal{O}$ (and not solving the open problem). However, for our main, combined class, $\mathsf{radi}\mathcal{O}$, while verifying with a witness is easy, we do not know how to test whether two circuits are in the class, and conjecture that this may be hard. Further exploring this and the implications for other potential applications of $\mathsf{i}\mathcal{O}$ with polynomial security, remains an interiguing open problem.

1.3 Technical Overview

Decomposable Obfuscation. As mentioned before, the construction of $\mathsf{d}\mathcal{O}$ for n-bit inputs uses a binary tree of depth n as an underlying structure. Each

internal node is assigned a circuit that is the partial evaluation of $C(x_{\leq}, \cdot)$, where $x_{\leq}$ is an input prefix of length i bits equal to the index of the node in its level. Thus the leaves of the tree contain the circuit evaluations on each possible input.

The obfuscated $\mathsf{d}\mathcal{O}$ object consists of n pairs of FE decryption keys, each pair corresponding to the two possible values of each input bit, and a starting FE ciphertext pair corresponding to the original circuit, with no input, each one decryptable by a different decryption key of the first pair. The $\mathsf{d}\mathcal{O}$ evaluation works by starting from the given ciphertext pair, and successively generating new pairs of FE ciphertexts that correspond to partial evaluations of the obfuscated function on each input prefix, using FE decryption, until revealing the output at the leaf. In more detail, each FE ciphertext is an encryption of two data fields: the partial circuit evaluation on a prefix of the input bits, and a string of pseudorandom values. Depending on the value of the next bit of the input, the evaluator chooses the decryption key indexed by the input bit, to decrypt the corresponding ciphertext, which is encrypted with the matching FE encryption key. The decryption algorithm applies the next bit to the partial circuit, and uses a PRG to expand the pseudorandom values into the pseudorandom values for plaintexts of the next FE ciphertext pairs, as well as the random coins for their encryption algorithms.

But this version of the obfuscation appears to depend on the circuit itself. To show that the obfuscation of two equivalent circuits is indistinguishable, Liu and Zhandry use hybrids that decompose circuits using the notion of a tree cover. A tree cover is a subset of input prefixes so that all full inputs are the extensions of exactly one element in the tree cover. The proof hybrids obfuscate intermediate circuit representations, called circuit assignments, which are a set of fragments, representing partial evaluations of the starting circuit, generated and indexed by a tree cover. For any input prefix that is an extension of a member of the tree cover, the generated partial circuit will still be hidden in an FE ciphertext pair, which is again generated by FE decryption, like the aforementioned default case, *ie.* plug in the input bit to generate the new fragment, and use the PRG to expand the pseudorandom seed to generate a new pseudorandom seed, and randomness for the FE encryption algorithm. The new feature is that if the input prefix is a (possibly improper) prefix of a member of the tree cover, then its FE ciphertext pair is precomputed, but hidden in an sk ciphertext, which is also precomputed during obfuscation time, and the sk ciphertext is stored in a random index in the function of the FE decryption key that would reveal the FE ciphertext. To uncover this pair, the FE ciphertext corresponding to the immediate prefix contains not a fragment and a random string, but the index pointing to the position, and the sk decryption key. Since this prefix would also be a prefix of an element in the tree cover, the index and key are also assigned during obfuscation time. During evaluation time, when the FE decryption key detects that the plaintext actually contains an index and an sk key, it will use the index to find the sk ciphertext stored in the decryption function, and decrypt it using the sk key. The indexes are wholly independent of the actual circuit, which means that the obfuscation only depends on the circuit assignment. Thus,

if we have a common tree cover between two circuits, the obfuscation of their respective circuit assignments is statistically equivalent. So we need to prove that the original obfuscation of a circuit, i.e, without the index-key ciphertexts, is indistinguishable from that of the final circuit assignment.

To do so, we move between hybrids of circuits of "adjacent" circuit assignments. That means that the two circuit assignments are identical, except that in one hybrid, one input prefix is on the tree cover, and in the other hybrid, it is the parent of two nodes on the tree cover. Proving indistinguishability between adjacent circuit assignments relies on sk security to hide the existence of the stored FE ciphertext pairs, which do not exist in the default case; FE security to hide if an FE ciphertext generated the next ciphertext pair, as in the default case, or revealed it, if it was stored, since the output is the same in both cases, an pair of FE ciphertexts that encrypt the next fragment and a random string; and PRG security to hide if the pseudorandom parts were generated from the previous ciphertext, as in the default case, or were freshly generated, and stored in the corresponding FE decryption key. If the common tree cover is only polynomial size, then we only have a polynomial number of hybrid steps from the default obfuscation to the obfuscation of the circuit assignment, e.g., by following a depth first search order until reaching the tree cover. In this special case, we therefore have polynomial security loss.

Our Results. We present a new obfuscation construction from functional encryption that provides indistinguishability assuming only subpolynomial security for the underlying FE for a larger class of circuits than the one handled by decomposabe obfuscation. In particular, we no longer need to require that any two circuits that have indistinguishability obfuscation have a common cover of polynomial size. Instead we require that the number of fragments at and before the common cover is polynomial, which still allows for a exponential size of the cover, and in addition to that we need a common subcircuit, also called a fragment, structure which we define precisely after the introduction. Our construction proceeds in two steps.

Repeated subcircuit obfuscation ($\mathsf{rescuei}\mathcal{O}$). The goal for this obfuscation construction is to relax the $\mathsf{d}\mathcal{O}$ requirement for a polynomial size common tree cover to a polynomial number of unique fragments on and before that cover. Our construction follows closely the $\mathsf{d}\mathcal{O}$ construction but we introduce new techniques to obtain polynomial-hybrid security reductions between circuits that have this property. In particular, instead of storing ciphertexts for every input prefix at or before the tree cover, we only store a ciphertext for unique fragments.

But by reusing FE ciphertexts for the same fragments, the obfuscation, or more specifically, its evaluation, could leak information about the obfuscated circuit. For example if for one circuit, two input prefixes produce different fragments, while for the second circuit, the two input prefixes produce the the same fragment, then a distinguisher can try evaluating up to those two prefixes. If the resulting FE ciphertexts are different, then the original circuit is the first one; otherwise the original circuit is the second one.

To avoid such problems, we impose a new restriction on the pairs of circuits that we can obfuscate. Unlike the requirements for $\mathsf{d}\mathcal{O}$, these new requirements apply to the fragments before the tree cover. In short, we require a bijection between fragments of the two circuits so that for any input prefix, we can apply the bijection on the resulting fragment of the first circuit, and get the fragment of the second circuit for the same prefix.

Hybrid $\mathsf{rescuei}\mathcal{O} + \mathsf{d}\mathcal{O}$ solution. As we mentioned above, the evaluation of the $\mathsf{rescuei}\mathcal{O}$ obfuscation reveals the structure of common fragments in a tree cover. This means that we can transition between obfuscations of two different circuits only if they have the same common fragment structure among the nodes above the common tree cover. This requirement could be potentially more restrictive than what is required by $\mathsf{d}\mathcal{O}$ in the sense that there are circuits that have a polynomial size common tree cover without having the common fragment structure.

We remove this limitation in a construction that combines $\mathsf{rescuei}\mathcal{O}$ and $\mathsf{d}\mathcal{O}$. Our goal is to be able to use the $\mathsf{d}\mathcal{O}$ proof techniques to replace the obfuscated circuit, with its fragment representation, by a polynomial size cover, which removes any information about the circuit structure above the $\mathsf{d}\mathcal{O}$ tree cover. Then, we can use the techniques of $\mathsf{rescuei}\mathcal{O}$ to move to a more fine-grained fragment representation that only requires a polynomial number of unique circuits but allows a potentially exponential number of input prefixes. We achieve this by composing the two obfuscation techniques, first applying $\mathsf{rescuei}\mathcal{O}$ and then obfuscating the resulting circuit using $\mathsf{d}\mathcal{O}$. With this construction and the above proof approach we only need to require the common fragment structure across circuits for nodes *between* the $\mathsf{d}\mathcal{O}$ and $\mathsf{rescuei}\mathcal{O}$ covers. This is enabled by the fact that the partially evaluated circuits residing in the $\mathsf{d}\mathcal{O}$ tree cover are partial evaluations of the $\mathsf{rescuei}\mathcal{O}$ obfuscation, which do not contain FE ciphertexts used in nodes above the $\mathsf{d}\mathcal{O}$ tree cover. This means that we do not need to worry about the fragment structure in that part of the tree in the hybrid while switching to obfuscation of the second circuit. Note that we do not avoid a second restriction, which is that the number of unique circuits *before* the tree cover must be polynomial. Nevertheless, our result produces a strictly larger class of obfuscatable circuit pairs than $\mathsf{d}\mathcal{O}$.

2 Common Definitions

The work of Liu and Zhandry [LZ17] defines the notion of decomposable obfuscation, $\mathsf{d}\mathcal{O}$. In this section, we describe its properties and limitations.

It is based on a locally decomposable obfuscator, $\mathsf{ld}\mathcal{O}$ that takes as input a circuit assignment, which does not necessarily include the circuit itself, but a set of partially-evaluated versions of the circuit, also known as fragments. These fragments are exactly enough to calculate the output of the circuit for all possible inputs. In particular, the fragment with a matching input prefix is chosen, and the remaining input bits are plugged in to calculate the final output.

To define $\mathsf{Id}\mathcal{O}$ and $\mathsf{d}\mathcal{O}$, we first need to define circuit assignments, which depend on tree covers, and fragments. We define a fragment as follows:

Definition 1 *Let C be a circuit which takes as input $\{0,1\}^n$. For any string, $x_{\leq} \in \{0,1\}^{i^*}$, where $i^* \leq n$, we define a fragment of C to be a circuit $C(x_{\leq}, \cdot)$, where $\forall i \leq i^*$, the wires for input bit i are hardcoded with the bit value $x_{\leq}[i]$, and any circuit gates that have a constant input wire are simplified. If the fragment is evaluated on inputs $x_{\geq} \in \{0,1\}^{n-i^*}$, then it outputs $C(x_{\leq}||x_{\geq})$.*

We define a tree cover, which is a set of bit strings, so that every input string is either in the tree cover, or has a unique prefix in it. For two strings, $x_{\leq}, x$, let $x_{\leq} \sqsubseteq x$ denote that $x_{\leq}$ is a (possibly improper) prefix of x. Then a tree cover is defined as follows:

Definition 2 *A tree cover, TC, is a subset of $\bigcup_{i^*=0}^{n} \{0,1\}^{i^*}$, so that $\forall x \in \{0,1\}^n$, there exists exactly one $x_{\leq} \in \mathsf{TC}$, so that $x_{\leq} \sqsubseteq x$.*

A circuit and a tree cover allow us to derive a circuit assignment, as follows:

Definition 3 *For a circuit, C, and a tree cover, TC, a circuit assignment is* $\mathsf{Assignment}(C, \mathsf{TC})$*:* $\mathsf{Assignment}(C, \mathsf{TC}) = \{(x_{\leq}, C(x_{\leq}, \cdot)) : x_{\leq} \in \mathsf{TC}\}$

Now we define $\mathsf{Id}\mathcal{O}$ on a circuit assignment. Strictly speaking, its security is not between two functionally equivalent circuits, but between two adjacent, or locally decomposing equivalent, circuit assignments of the same circuit:

Definition 4 *A locally decomposable $\mathsf{i}\mathcal{O}$ algorithm, $\mathsf{Id}\mathcal{O}$ is an obfuscation algorithm that takes in a security parameter, a circuit assignment of C, a maximum tree cover size, l, and a maximum fragment size, s, and outputs a new circuit.*

For any $x_{\leq} \in \mathsf{TC}$, such that $|x_{\leq}| < n$, let $\mathsf{TC}_{x_{\leq}} = (\mathsf{TC} \setminus \{x_{\leq}\}) \cup \{x_{\leq}||b : b \in \{0,1\}\}$. $\mathsf{Assignment}(C, \mathsf{TC}_{x_{\leq}})$ *is a locally decomposing equivalent circuit assignment of* $\mathsf{Assignment}(C, \mathsf{TC})$*, and vice-versa. Then if $|\mathsf{TC}| \leq l$, and $|\mathsf{TC}_{x_{\leq}}| \leq l$, and $\forall (C^*, x_{\leq}) \in \mathsf{Assignment}(C, \mathsf{TC}) \cup \mathsf{Assignment}(C, \mathsf{TC}_{x_{\leq}})$, if $|C^*| \leq s$, then their local obfuscations are computationally equivalent:*

$$\mathsf{Id}\mathcal{O}(1^\lambda, \mathsf{Assignment}(C, \mathsf{TC}), l, s) \overset{c}{\approx} \mathsf{Id}\mathcal{O}(1^\lambda, \mathsf{Assignment}(C, \mathsf{TC}_{x_{\leq}}), l, s)$$

In general, obfuscation schemes only take in circuits, not circuit assignments. Thus we define the top-level obfuscator, $\mathsf{d}\mathcal{O}$, on the root circuit assignment, $\{(C, \epsilon)\}$.

Definition 5 *For a $\mathsf{Id}\mathcal{O}$ scheme, a decomposable obfuscator, or $\mathsf{d}\mathcal{O}$, is an obfuscation scheme defined as $\mathsf{d}\mathcal{O}(1^\lambda, C, l, s) = \mathsf{Id}\mathcal{O}(1^\lambda, \{(C, \epsilon)\}, l, s)$.*

In special cases, this definition of $\mathsf{d}\mathcal{O}$ allows us to discuss indistinguishability between two circuits, rather than circuit assignments of a single circuit. We can transition between the obfuscation of two circuits by way of an identical circuit assignment. If the circuit assignment has polynomial size, Liu *et al.* prove that only subpolynomial security for $\mathsf{Id}\mathcal{O}$ suffices for $\mathsf{d}\mathcal{O}$ security [LZ17].

First we claim indistinguishability between the root and the common circuit assignment.

Lemma 1 *For any polynomial $l \geq 1$, tree cover, $\mathsf{TC} \leq l$, and circuit size, s, even if $\mathsf{ld}\mathcal{O}$ is only subpolynomially secure, then:*

$$\mathsf{d}\mathcal{O}(1^\lambda, C, l, s) = \mathsf{ld}\mathcal{O}(1^\lambda, \{(C, \epsilon)\}, l, s) \overset{c}{\approx} \mathsf{ld}\mathcal{O}(1^\lambda, \mathsf{Assignment}(C, \mathsf{TC}), l, s)$$

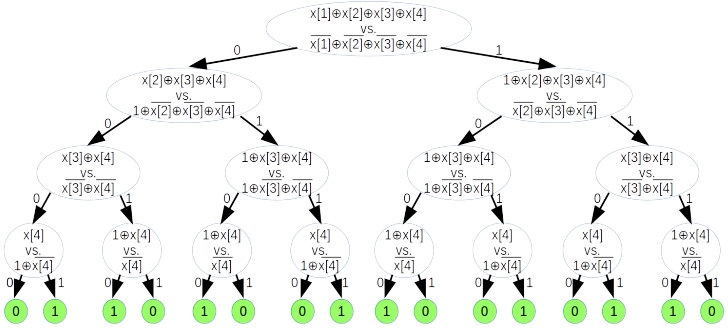
So we can claim indistinguishability between two different circuits:

Lemma 2 *For any polynomial, $l > 1$, and two circuits, C_0, C_1, if there exists a tree cover, $\mathsf{TC} \leq l$, and circuit size, s, so that $\mathsf{Assignment}(C, \mathsf{TC}) = \mathsf{Assignment}(C', \mathsf{TC})$, then the $\mathsf{d}\mathcal{O}$ obfuscations of the two circuits is computationally indistinguishable, ie. $\mathsf{d}\mathcal{O}(1^\lambda, C_0, l, s) \overset{c}{\approx} \mathsf{d}\mathcal{O}(1^\lambda, C_1, l, s)$.*

3 $\mathsf{d}\mathcal{O}$ Limitations

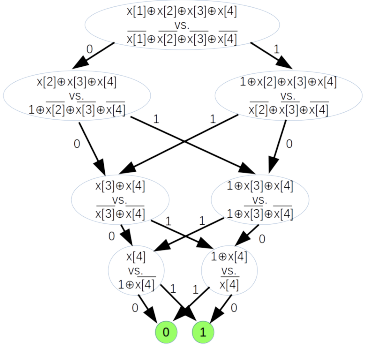
The limitation of $\mathsf{d}\mathcal{O}$ is that two circuits are only equivalent if they have an identical polynomia-size tree cover. We illustrate the cost of this limitation with the following example, where the common tree cover is exponentially large, even though there is only a very small number of unique circuits. Let n be even. The circuits $C_0(x) = \bigoplus_{i=1}^{n} x[i]$ and $C_1(x) = \bigoplus_{i=1}^{n} \overline{x[i]}$ are functionally equivalent. But no fragments will be identical until all input bits have been plugged in. So the common tree cover must have exponential size, as illustrated in Figure 1.

Fig. 1. Case where $\mathsf{d}\mathcal{O}$ requires an exponentially-sized tree cover for two circuits.



But if we look at what the fragments are for each input prefix, we notice that the fragments repeat, and in fact, at every level, C_0 and C_1 each have only at most two different kinds of fragments. Moreover, for every fragment of C_0, every input prefix that generates it generates exactly one fragment of C_1. We can consolidate these fragment pairs, as shown in Figure 2, and our obfuscation will take advantage of this property.

Fig. 2. Case where the number of fragments is polynomial.



4 Obfuscating a Circuit Assignment with Repeated Fragments

The example in the previous section shows that using $\mathsf{d}\mathcal{O}$, and looking for identical fragments between the two obfuscated circuits could lead to an exponential-size tree cover, even for relatively simple circuits. But it also points out a useful circuit property: a small number of repeated fragments. We introduce a technique that adapts the proof approach from $\mathsf{d}\mathcal{O}$ to take advantage of this property. This technique follows the ideas of $\mathsf{ld}\mathcal{O}$, but we introduce a different way to construct the hybrid sequence in the proof of indistinguishability, which leverages the circuit structure with repeated fragments. We thus call our algorithm REpeated SubCircUit Exploding $\mathsf{i}\mathcal{O}$, or $\mathsf{rescuei}\mathcal{O}$ for short.

Our new proof techniques allow the obfuscation size to depend only on the total number of unique fragments rather than all input prefixes in and before a tree cover, which could be exponential. We also show how we can argue indistinguishability across hybrids where we partially evaluate a fragment that may correspond to many input prefixes simultaneously. The novelty of our approach allows us to transition even to tree cover assignments with exponential number of input prefixes, as long as the unique fragments at and before tree cover are a polynomial number, while incurring only *polynomial security loss*. This expands the capabilities of the $\mathsf{ld}\mathcal{O}$ techniques that are the main tool for arguing indistinguishability for $\mathsf{d}\mathcal{O}$.

Below, we summarize the construction of $\mathsf{rescuei}\mathcal{O}$, and define the concepts and requirements necessary for its security. Due to space constraints, we will give a complete description of the construction and proof in Appendices C and C.1.

$\mathsf{rescuei}\mathcal{O}$ produces an obfuscation that is evaluated similarly to $\mathsf{d}\mathcal{O}$: the obfuscation contains a set of FE decryption keys and a starting pair of FE ciphertexts. For each input bit, the evaluator chooses the ciphertext for the input bit value, and decrypts it using the corresponding decryption key. And like in $\mathsf{d}\mathcal{O}$, the default case only obfuscates the original circuit. But already in this default case, we modified the function that the FE decryption key evaluates . Again, the plaintext contains a partial circuit and a random seed, but the random seed does not

necessarily vary by input prefix, because we do not want visibly different ciphertexts for the same fragment at different input prefixes. Instead, each input prefix length has a single seed that is shared amongst all evaluations at that length. To make it possible to have a per-level seed, and have the same pseuorandom output for the same fragment at different input prefixes, we use a PRF instead of a PRG: the PRF is evaluated on a constant to produce the key for the next level, and on the next fragment and the input bit to produce the random coins for the next FE encryptions. Inductively, by evaluating a PRF on a constant, each level will have exactly one PRF key. As a result the random coins for FE encryption will only depend on the fragment and the last input bit used to generate the fragment. That means that for two partial evaluations of the obfuscation, if the last bit is the same, and they result in the same underlying fragment of the original circuit, then the resulting FE ciphertext is identical.

This property lets us obfuscate circuit assignments by storing FE ciphertexts similar to $\mathsf{d}\mathcal{O}$. First, we need to define what ciphertexts we store, and what we generate. Instead of an arbitrary tree cover, our analogue considers all prefixes of a certain length:

Definition 6 *$\forall i^* \in n$, a level assignment, of circuit C for prefix length i^* is a set of **unique fragments**,* $\mathsf{LevelAssignment}(C, i^*)$*:*

$$\mathsf{LevelAssignment}(C, i^*) = \{C(x_{\leq}, \cdot) : x_{\leq} \in \{0,1\}^{i^*}\}$$

Note, that for the purposes of our algorithm, we do not keep track of which input prefixes give us which fragment, as the relation may be many-to-one. For the rest of the paper, we consider the circuits, C, so that when $\mathsf{LevelAssignment}(C, i^*)$ has polynomial size, it is known (which is true if all the previous level assignments are polynomial-size and known).

In $\mathsf{d}\mathcal{O}$, the decryption functions store ciphertexts up to the tree cover; in $\mathsf{rescuei}\mathcal{O}$, the decryption functions store ciphertexts for inputs whose lengths are up to and including i^* bits. The stored data of the two obfuscation schemes are nearly identical. They are both FE ciphertext pairs encrypted by an sk key. The main difference is how many ciphertexts are stored. In $\mathsf{d}\mathcal{O}$, there is a fresh ciphertext for every input prefix. On the other hand, $\mathsf{rescuei}\mathcal{O}$ contains only two ciphertexts for each unique fragment, one for each input bit. This is mirrored by the default evaluation of the obfuscation of the original circuit, in which only two ciphertexts could ever be generated for a fragment. In fact, for input prefixes whose lengths are strictly less than i^*, the plaintexts used to generate the FE ciphertext pairs are identical between the two schemes: they contain a pointer to the successor ciphertext, and the sk decryption key to uncover the FE ciphertext. For input prefixes of length i^*, the syntax plaintexts of the two schemes are similar: they contain a fragment and random coins. But for $\mathsf{rescuei}\mathcal{O}$, the random coins are identical for the whole level, which ensures that for the following levels, the random coins continue to be identical, like in the aforementioned default evaluation mode.

However, by generating or storing ciphertexts for unique fragments, the structure of the pointers is now dependent on how the original circuit generates those

fragments. That means that the obfuscation is dependent not only on the fragments in the level assignment, but also on the fragments before it. Therefore, it is not enough that two circuits have an identical level assignment. They require a similar structure for the prior fragments, too. The requirement for two circuits to be indistinguishable under $\mathsf{rescuei}\mathcal{O}$ is therefore defined as consistency, as follows:

Definition 7 (Consistent Circuits) *Two circuits, C_0, C_1, are consistent for i^*, if there exists a bijection τ and a polynomial values l, s, so that*

1. $\forall x$, *where* $|x| \leq i^*$: $\tau(C_0(x,\cdot)) = C_1(x,\cdot)$
2. $\forall x \in \{0,1\}^{i^*} : C_0(x,\cdot) = C_1(x,\cdot)$,
3. $|\bigcup_{i=0}^{i^*} \mathsf{LevelAssignment}(C_0, i)| \leq l$ *and* $|\bigcup_{i=0}^{i^*} \mathsf{LevelAssignment}(C_1, i)| \leq l$
4. $\forall C^* \in \bigcup_{i=0}^{i^*} \mathsf{LevelAssignment}(C_0, i) \cup \bigcup_{i=0}^{i^*} \mathsf{LevelAssignment}(C_1, i)$, $|C^*| \leq s$.

The mapping requirement is the main property that would make $\mathsf{rescuei}\mathcal{O}$ inapplicable to certain cases that $\mathsf{d}\mathcal{O}$ can obfuscate. Therefore, so far, even though $\mathsf{rescuei}\mathcal{O}$ can obfuscate some circuit pairs that $\mathsf{d}\mathcal{O}$ cannot, the two schemes are incomparable. Also note that even though the mapping property is defined over a potentially-exponential number of input prefixes, it can be efficiently checked, given the τ. In fact, we can show in Section 6 that if the two circuits are consistent, we can efficiently check it with an iterative algorithm.

5 radi$\mathcal{O}$: Combining d$\mathcal{O}$ and rescuei$\mathcal{O}$

We can provide indistinguishability obfuscation for the class of consistent circuits that satisfy Definition 7. However, this requirement is restrictive, and could exclude circuits that could be obfuscated in polynomial security loss with $\mathsf{d}\mathcal{O}$.

For a trivial example, assume some common subcircuit, C^* that takes $n-3$ bits as input, and the two functionally equivalent circuits:

$$C_0(x) = x[1] \oplus x[2] \oplus x[3] \oplus C^*(x[4], \ldots, x[n])$$

$$C_1(x) = \mathsf{Select}(x[1], \mathsf{Select}(x[2], x[3], \overline{x[3]}), \mathsf{Select}(x[2], \overline{\overline{x[3]}}, \overline{\overline{x[3]}})) \oplus C^*(x[4], \ldots, x[n])$$

Note that the selection functions in C_1 form a more complicated circuit that also performs XOR.

Because the only difference between the two circuits is in the first, constant number of bits, they can be obfuscated by $\mathsf{d}\mathcal{O}$. On the other hand, it cannot be obfuscated using $\mathsf{rescuei}\mathcal{O}$, because in C_0, the first two bits produce 2 fragments, while in C_1, the first two bits already produce 4, making a mapping impossible.

We show how we can overcome the restrictiveness of Definition 7 to obtain an obfuscation construction that relies on a subpolynomial security assumption for FE and can obfuscate an extended class of circuits that is a proper superset of the class handled by $\mathsf{d}\mathcal{O}$. Our idea is to relax the bijection requirement from Definition 7 by applying $\mathsf{ld}\mathcal{O}$ on the output of $\mathsf{rescuei}\mathcal{O}$, to hide inconsistencies before the tree cover, TC, used to form the $\mathsf{ld}\mathcal{O}$ circuit assignment.

In this way, we only require that the input prefixes of length at least i^* produce identical fragments, not for the whole tree cover TC. Furthermore there is no structural restriction on the input prefixes before TC. The tree cover TC and length restriction i^* will split the space of fragments into three regions, which we will formalize in 5.1. In short, they are: all the fragments that are only generated by input prefixes that are proper prefixes of those in TC, all the fragments that have a prefix in each original tree cover (equivalently, they have a prefix in the combined tree cover), and all of the remaining fragments. We can remove any remaining dependencies on fragments in the first group. For two circuits, the second group will be identical if we require that all the fragments formed by the new tree cover are identical. The third group is the only one for which the mapping applies.

Due to the security of $\mathsf{rescuei}\mathcal{O}$, we can increase i^*, until the fragments at level i^* are identical for the two circuits. Then, due to the security of $\mathsf{Id}\mathcal{O}$, we can decompose the circuit assignment to TC, until it meets the new requirements. Then we are in a state, where the obfuscations of the two circuits are indistinguishable, based on arguments we will make for each of the three regions. Actually, the region of fragments beyond i^* do not need to be hidden, as they are identical.

5.1 Properties of Fragment Partitions

More precisely, we define the following three regions, which may only overlap at their boundaries:

1. Before TC:

$$\mathsf{PreTC}_{C,\mathsf{TC}} = \{C^* : \nexists x \in \{0,1\}^*, x_{\leq} \in \mathsf{TC} \text{ so that } C^* = C(x,\cdot) \wedge x_{\leq} \sqsubseteq x\}$$

2. After TC and i^*: Define the set of input prefixes:

$$\mathsf{AfterInput}(\mathsf{TC}, i^*) = \{x : |x| \geq i^* \wedge \exists x_{\leq} \in \mathsf{TC} \text{ so that } x_{\leq} \sqsubseteq x\}$$

   We further define the boundary of $\mathsf{AfterInput}(\mathsf{TC}, i^*)$:

$$\begin{aligned}\mathsf{MinAfterInput}(\mathsf{TC}, i^*) = &\{x \in \mathsf{TC} : |x| \geq i^*\} \cup \\ &\{x \in \{0,1\}^{i^*} : \exists x_{\leq} \in \mathsf{TC} \text{ so that } |x_{\leq}| \leq |x| \wedge x_{\leq} \sqsubseteq x\}\end{aligned}$$

   We can see that any input prefix that has a prefix in $\mathsf{MinAfterInput}(\mathsf{TC}, i^*)$ has a prefix in TC, and has at least i^* bits. The only difference between the two subsets is which tree cover element comes first. Conversely, for every input prefix $x \in \mathsf{AfterInput}(\mathsf{TC}, i^*)$, either the prefix, $x_{\leq} \in \mathsf{TC}$ has at least i^* bits, in which case that prefix would be in the first subset, or it is the prefix of strings with i^* bits. In the second case, since $|x| \geq i^*$, x must have a prefix of length i^*, which also has $x_{\leq}$ as a prefix.
   This region also generates circuits, but we do not define them, as no algorithm will use them explicitly.

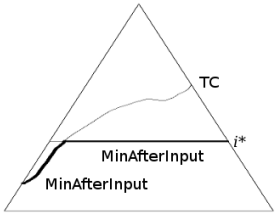
3. Between TC and i^*: Define the set of input prefixes:

$$\mathsf{InterInput}(\mathsf{TC}, i^*) = \{x : |x| \leq i^* \wedge \exists x_{\leq} \in \mathsf{TC} \text{ so that } x_{\leq} \sqsubseteq x\}$$

Then the set of circuits is $\mathsf{InterCirc}(C, \mathsf{TC}, i^*) = \{C(x_{\leq}, \cdot) : x_{\leq} \in \mathsf{InterInput}(\mathsf{TC}, i^*)\}$.

The input prefixes corresponding to the first and third regions are shown in Figure 3.

Fig. 3. The boundaries of the input prefix regions



The composed obfuscation requires that two circuits, C_0, C_1, be combined-cover consistent, which is defined as follows:

Definition 8 *Two circuits, C_0, C_1, are combined-cover consistent for tree cover TC and length i^*, if there exist a polynomial size l and a bijection τ: $\mathsf{InterCirc}(C_0, \mathsf{TC}, i^*) \rightarrow \mathsf{InterCirc}(C_1, \mathsf{TC}, i^*)$, so that:*

1. $\forall x \in \mathsf{TC} \cup \mathsf{InterInput}(\mathsf{TC}, i^*) : \tau(C_0(x, \cdot)) = C_1(x, \cdot)$,
2. $\forall x \in \mathsf{MinAfterInput}(\mathsf{TC}, i^*) : C_0(x, \cdot) = C_1(x, \cdot)$,
3. $|\bigcup_{i=0}^{i^*} \mathsf{LevelAssignment}(C_0, i)| \leq l$ *and* $|\bigcup_{i=0}^{i^*} \mathsf{LevelAssignment}(C_1, i)| \leq l$

In Appendix E, we give a complete proof of the security of the composition. In short, by applying $\mathsf{d}\mathcal{O}$, the final obfuscation only depends on the partial circuits at the $\mathsf{d}\mathcal{O}$ tree cover, which hides any violations of the consistency property before it, thus reducing the number of input prefixes for which the consistency property must hold. This relaxation indeed allows for the obfuscation of circuit pairs that cannot be obfuscated by $\mathsf{d}\mathcal{O}$ and $\mathsf{rescuei}\mathcal{O}$ alone, with an example shown in Appendix F.

6 Testing Consistency

Recall the Definition 7 of consistent circuits in $\mathsf{rescuei}\mathcal{O}$. In this case, testing consistency means that, given two circuits, we have to find an i^* and a bijection τ such that,

– Two circuits are identical after partial evaluating with the same input prefix of length i^*;

– Informally speaking, τ is a bijection between two sets of unique partial evaluated circuits and matches them with the same input prefix.

We present an efficient iterative algorithm to solve this problem.

Theorem 1 *There exists a deterministic algorithm that decides in polynomial time whether these two give circuits C_0 and C_1 are consistent for some polynomials l and s.*

Proof. Initially, we have $\mathsf{LevelAssignment}(C_b, 0) = \{C_b\}$ for $b \in \{0, 1\}$, and τ maps C_0 to C_1.

We iteratively generate $\mathsf{LevelAssignment}(C_b, i)$ for $i = 1, 2, \ldots, n$ and check the existence of a good mapping τ:

- First generate $\mathsf{LevelAssignment}(C_b, i)$ using $C^* \in \mathsf{LevelAssignment}(C_b, i-1)\}$. Return not consistent if $|\mathsf{LevelAssignment}(C_0, i)| \neq |\mathsf{LevelAssignment}(C_1, i)|$ or the number of circuits exceeds the limit l or any of those circuits exceeds the size limit s.
- Then we scan $\mathsf{LevelAssignment}(C_0, i)$ to construct the bijection τ in the following way:
  For each $C_0^* \in \mathsf{LevelAssignment}(C_0, i-1)$, assign $\tau(C_0^*(b, 0)) = \tau(C_0^*)(b, 0)$. Return not consistent if there's any conflict or τ is not bijective.
- Return consistent when $C_0^* = \tau(C_0^*)$ for every $C_0^* \in \mathsf{LevelAssignment}(C_0, i)$. Moreover, this i is our desired i^*.

The above algorithm takes time $O(nl^2s)$.

Then it's sufficient to show that the constructed bijection τ satisfies that $\tau(C_0(x, \cdot)) = C_1(x, \cdot)$ for every $|x| \leq i$. We can show it by induction on i. Clearly it holds when $i = 0$, and the construction above ensures that for every $C_0^* \in \mathsf{LevelAssignment}(C_0, i-1)$, $\tau(C_0^*(b, 0)) = \tau(C_0^*)(b, 0)$. Which concludes the result.

In the composition of $\mathsf{d}\mathcal{O}$ and $\mathsf{rescuei}\mathcal{O}$, we also defined the combined-cover consistency in Definition 8. But unfortunately we didn't come up with a polynomial time algorithm to check whether two circuits are combined-cover consistent. Like testing consistency in $\mathsf{d}\mathcal{O}$, the above algorithm takes advantages of the existence of a minimum tree cover that satisfies certain properties. But in our definition of combined-cover consistency, even if we know the length i^*, there could be multiple (even exponentially many) minimal tree covers that satisfy those properties. And it's not easy to certify that all those minimal tree covers exceed the size limit and hence two given circuits are not consistent.

We tried a modified version of the algorithm for testing consisntency in $\mathsf{d}\mathcal{O}$: it keeps decomposing the tree covers (originally it contains only the root) until all pairs of corresponding partial evaluated circuits are consistent in the $\mathsf{rescuei}\mathcal{O}$ definition, *ie.* we found a good bijection τ and all pairs of corresponding partial evaluated circuits after certain level i^* are identical. But merging those τ's doesn't immediately give us our desired bijection τ in the definition of combined-cover consistence. We are curious about whether we can tweak this algorithm and make it work.

References

[AJ15] Prabhanjan Ananth and Abhishek Jain. Indistinguishability obfuscation from compact functional encryption. In *Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part I*, pages 308–326, 2015.

[AJS15] Prabhanjan Ananth, Abhishek Jain, and Amit Sahai. Indistinguishability obfuscation from functional encryption for simple functions. Cryptology ePrint Archive, Report 2015/730, 2015. `http://eprint.iacr.org/2015/730`.

[AS17] Prabhanjan Ananth and Amit Sahai. Projective arithmetic functional encryption and indistinguishability obfuscation from degree-5 multilinear maps. In *Advances in Cryptology - EUROCRYPT 2017 - 36th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Paris, France, April 30 - May 4, 2017, Proceedings, Part I*, pages 152–181, 2017.

[BGI+01] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil Vadhan, and Ke Yang. On the (im) possibility of obfuscating programs. 2001.

[BGK+14] Boaz Barak, Sanjam Garg, Yael Tauman Kalai, Omer Paneth, and Amit Sahai. Protecting obfuscation against algebraic attacks. In *Advances in Cryptology - EUROCRYPT 2014 - 33rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Copenhagen, Denmark, May 11-15, 2014. Proceedings*, pages 221–238, 2014.

[BNPW16] Nir Bitansky, Ryo Nishimaki, Alain Passelègue, and Daniel Wichs. From cryptomania to obfustopia through secret-key functional encryption. In *Theory of Cryptography - 14th International Conference, TCC 2016-B, Beijing, China, October 31 - November 3, 2016, Proceedings, Part II*, pages 391–418, 2016.

[BPR15] Nir Bitansky, Omer Paneth, and Alon Rosen. On the cryptographic hardness of finding a nash equilibrium. In *Foundations of Computer Science (FOCS), 2015 IEEE 56th Annual Symposium on*, pages 1480–1498. IEEE, 2015.

[BPW16] Nir Bitansky, Omer Paneth, and Daniel Wichs. *Perfect Structure on the Edge of Chaos*, pages 474–502. Springer Berlin Heidelberg, Berlin, Heidelberg, 2016.

[BV15] Nir Bitansky and Vinod Vaikuntanathan. Indistinguishability obfuscation from functional encryption. In *IEEE 56th Annual Symposium on Foundations of Computer Science, FOCS 2015, Berkeley, CA, USA, 17-20 October, 2015*, pages 171–190, 2015.

[BZ14] Dan Boneh and Mark Zhandry. *Multiparty Key Exchange, Efficient Traitor Tracing, and More from Indistinguishability Obfuscation*, pages 480–499. Springer Berlin Heidelberg, Berlin, Heidelberg, 2014.

[GGH+13] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *FOCS*, 2013.

[GGM84] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct randolli functions. In *Foundations of Computer Science, 1984. 25th Annual Symposium on*, pages 464–479. IEEE, 1984.

[GGSW13] Sanjam Garg, Craig Gentry, Amit Sahai, and Brent Waters. Witness encryption and its applications. In *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013*, pages 467–476, 2013.

[GLSW15] Craig Gentry, Allison Bishop Lewko, Amit Sahai, and Brent Waters. Indistinguishability obfuscation from the multilinear subgroup elimination assumption. In *IEEE 56th Annual Symposium on Foundations of Computer Science, FOCS 2015, Berkeley, CA, USA, 17-20 October, 2015*, pages 151–170, 2015.

[GPS16] Sanjam Garg, Omkant Pandey, and Akshayaram Srinivasan. Revisiting the cryptographic hardness of finding a nash equilibrium. In *Advances in Cryptology - CRYPTO 2016 - 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part II*, pages 579–604, 2016.

[GPSZ17] Sanjam Garg, Omkant Pandey, Akshayaram Srinivasan, and Mark Zhandry. Breaking the sub-exponential barrier in obfustopia. In *Advances in Cryptology - EUROCRYPT 2017 - 36th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Paris, France, April 30 - May 4, 2017, Proceedings, Part III*, pages 156–181, 2017.

[GS16] Sanjam Garg and Akshayaram Srinivasan. Single-key to multi-key functional encryption with polynomial loss. In *Theory of Cryptography - 14th International Conference, TCC 2016-B, Beijing, China, October 31 - November 3, 2016, Proceedings, Part II*, pages 419–442, 2016.

[KNT17] Fuyuki Kitagawa, Ryo Nishimaki, and Keisuke Tanaka. Indistinguishability obfuscation for all circuits from secret-key functional encryption. Cryptology ePrint Archive, Report 2017/361, 2017. `http://eprint.iacr.org/2017/361`.

[Lin16] Huijia Lin. Indistinguishability obfuscation from constant-degree graded encoding schemes. In *Advances in Cryptology - EUROCRYPT 2016 - 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part I*, pages 28–57, 2016.

[LT17] Huijia Lin and Stefano Tessaro. Indistinguishability obfuscation from trilinear maps and block-wise local prgs. In *Advances in Cryptology - CRYPTO 2017 - 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20-24, 2017, Proceedings, Part I*, pages 630–660, 2017.

[LV16] Huijia Lin and Vinod Vaikuntanathan. Indistinguishability obfuscation from ddh-like assumptions on constant-degree graded encodings. In *IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS 2016, 9-11 October 2016, Hyatt Regency, New Brunswick, New Jersey, USA*, pages 11–20, 2016.

[LZ17] Qipeng Liu and Mark Zhandry. Decomposable obfuscation: A framework for building applications of obfuscation from polynomial hardness. In *Theory of Cryptography - 15th International Conference, TCC 2017, Baltimore, MD, USA, November 12-15, 2017, Proceedings, Part I*, pages 138–169, 2017. See also `http://eprint.iacr.org/2017/209`.

[PST14] Rafael Pass, Karn Seth, and Sidharth Telang. Indistinguishability obfuscation from semantically-secure multilinear encodings. In *Advances in Cryptology - CRYPTO 2014 - 34th Annual Cryptology Conference*, pages 500–517, 2014.

[SW14] Amit Sahai and Brent Waters. How to use indistinguishability obfuscation: Deniable encryption, and more. In *STOC*, 2014.

A Definitions

Notation. For two non-negative integers, u, v, let $[u, v]$ denote a sorted list of all integers greater than or equal to u, and less than or equal to v. If $u > v$, then the list is empty. We represent the special case where $u = 1$ by $[v]$.

For a bit string, x, and some index, $i \in [|x|]$, let $x[i]$ denote the bit of x in position i.

We denote two statistically equivalent distributions, $\mathcal{D}_0$ and $\mathcal{D}_1$, by:

$$\mathcal{D}_0 \overset{s}{\equiv} \mathcal{D}_1$$

If they are indistinguishable, we denote their relationship by:

$$\mathcal{D}_0 \overset{c}{\approx} \mathcal{D}_1$$

A.1 General Obfuscation

We will be discussing multiple obfuscation schemes, but all of them take at least a circuit as input, and output a circuit, or the parameters to calculate a circuit. The scheme must be efficient, and the output must be functionally equivalent to the input circuit. The following is a generalization of common properties of obfuscators, as described in [BGI+01] and [LZ17]:

Definition 9 *An obfuscation algorithm, $\mathcal{O}$, takes as input a security parameter, 1^λ, any circuit, C, and auxiliary input, z, and outputs a circuit, $\tilde{C}$. The algorithm $\mathcal{O}$ should satisfy the following properties:*

1. *Efficient: $\mathcal{O}$ is a PPT algorithm over λ, C, and z.*
2. *Correct:* $\Pr[\forall x, \tilde{C}(x) = C(x) | \tilde{C} \leftarrow \mathcal{O}(1^\lambda, C, z)] = 1$

Indistinguishability obfuscation, or $\mathsf{i}\mathcal{O}$, is a general-purpose obfuscation algorithm that makes any two equal-sized, functionally equivalent circuits indistinguishable:

Definition 10 *An obfuscation algorithm, $\mathsf{i}\mathcal{O}$, is an indistinguishability obfuscator if for any two circuits, C_0, and C_1, where $\forall x$, $C_0(x) = C_1(x)$, and $|C_0| = |C_1|$:*

$$\mathsf{i}\mathcal{O}(1^\lambda, C_0) \overset{c}{\approx} \mathsf{i}\mathcal{O}(1^\lambda, C_1)$$

A.2 Functional Encryption

$\mathsf{i}\mathcal{O}$, $\mathsf{d}\mathcal{O}$, and our schemess all have constructions based on functional encryption, or FE. We define its algorithms, and its security game, in the way that it is used for construction $\mathsf{d}\mathcal{O}$ [LZ17]:

Definition 11 *An* FE *scheme consists of the following parts:*

1. $\mathsf{FE.Gen}(1^\lambda)(1^\lambda)$*: Outputs a master secret key, and a master public key* $(\mathsf{msk}, \mathsf{mpk})$.
2. $\mathsf{FE.Enc}(\mathsf{mpk}, m, r)$*: A deterministic algorithm that takes in a master public key, a plaintext, and randomness coins, and outputs a ciphertext,* c. $\mathsf{FE.Enc}(\mathsf{mpk}, m)$ *is the randomized version, where* r *is chosen uniformly at random.*
3. $\mathsf{FE.KeyGen}(\mathsf{msk}, f)$*: Takes in a master secret key and a function, and outputs a function key,* fsk.
4. $\mathsf{FE.Dec}(\mathsf{fsk}, c)$*: Takes in a function key and a ciphertext, and outputs a string.*

An FE *scheme must be correct, which means that the decryption algorithm evaluates function key's function on the underlying plaintext, which means that for all security parameters,* λ*, plaintexts,* m*, and functions,* f*:*

$$\Pr[\mathsf{FE.Dec}(\mathsf{fsk}, c) = \ |(\mathsf{msk}, \mathsf{mpk}) \leftarrow \mathsf{FE.Gen}(1^\lambda)(1^\lambda), c \leftarrow \mathsf{FE.Enc}(\mathsf{mpk}, m), \mathsf{fsk} \leftarrow \mathsf{FE.KeyGen}(\mathsf{msk}, f)] = 1$$

An FE *scheme is single-key selectively secure, which depends on the following game,* $\mathsf{Game}_{\lambda,\mathcal{A},b}$*, for* $b \in \{0,1\}$*, for adversary,* $\mathcal{A}$*:*

1. $\mathcal{A}$ *submits two messages,* m_0*,* m_1*, where* $|m_0| = |m_1|$.
2. *The challenger calculates* $(\mathsf{msk}, \mathsf{mpk}) \leftarrow \mathsf{FE.Gen}(1^\lambda)(1^\lambda)$*, and returns* mpk *to* $\mathcal{A}$.
3. $\mathcal{A}$ *submits a function,* f*, such that where* $f(m_0) = f(m_1)$.
4. $\mathcal{A}$ *outputs* b'*, which is the output of the game.*

FE *is single-key selectively secure if for any PPT,* $\mathcal{A}$*:*

$$|\Pr[\mathsf{Game}_{\lambda,\mathcal{A},0} = 1] - \Pr[\mathsf{Game}_{\lambda,\mathcal{A},1} = 1]| = negl.$$

An FE *scheme is compact if the running time of* $\mathsf{FE.Enc}(\mathsf{mpk}, m)$ *is bounded by* $\mathsf{poly}(\lambda, |m|)$*. In particular, it is independent of the size of any functions,* f*, from which function keys,* fsk*, are derived.*

B Summary of $\mathsf{d}\mathcal{O}$ Construction

We summarize the $\mathsf{d}\mathcal{O}$ construction of Liu and Zhandry [LZ17]. The input of the $\mathsf{Id}\mathcal{O}$ algorithm includes a circuit assignment, and its output consists of a pair of FE ciphertexts, (c_0, c_1), and $n+1$ pairs of FE decryption keys, $\{\mathsf{fsk}_i^{(b)}\}_{i\in[1,n+1],b\in\{0,1\}}$, each corresponding to an input bit index and input bit

value. In order to generate the function keys, the obfuscation algorithm generates FE master public and private keys, $\{(\mathsf{msk}_i^{(b)}, \mathsf{mpk}_i^{(b)})\}_{i\in n, b\in\{0,1\}}$. For each $i \in [i+1]$, $b \in \{0,1\}$ the obfuscation scheme will generate $\mathsf{fsk}_i^{(b)}$ from $\mathsf{msk}_i^{(b)}$, to evaluate a function, $f_i^{(b)}$, which we will describe later in this section. The FE decryption keys will contain several hardcoded SKE ciphertexts that will contain FE encryptions of either fragments of the obfuscated circuit or indexes to other ciphertexts.

We describe the construction in increasing level of detail, starting from the default case of $\mathsf{d}\mathcal{O}$, which only gives $\mathsf{Id}\mathcal{O}$ the circuit assignment $\{(C, \epsilon)\}$, and add more features to support general tree covers. In particular, we will fill in what plaintext each FE ciphertext contains, and the function that each $\mathsf{fsk}_i^{(b)}$ evaluates.

In the default case, for $b \in \{0,1\}$, the algorithm generates a randomness key, K_b, and outputs the following initial FE ciphertext:

$$c_b = \mathsf{FE.Enc}(\mathsf{mpk}_1^{(b)}, (C, K_b, \bot, \bot))$$

Each FE decryption key $\mathsf{fsk}_i^{(b)}$ is generated for a function $f_i^{(b)}$, which works as follows. It takes input of the form $(C^i, K, \bot, \bot)$ where C^i is the fragment of C partially evaluated on the input prefix $x[1]\dots x[i-1]$. It evaluates $C^{i+1} = C^i(b, \cdot)$, and applies a PRG on K to obtain two new seeds K_0, K_1, as well as random coins, r_0 and r_1. For $i < n$ it outputs a pair of ciphertexts

$$\mathsf{FE.Enc}(\mathsf{mpk}_{i+1}^{(0)}, (C^{i+1}, K_0, \bot, \bot), r_0)$$
$$\mathsf{FE.Enc}(\mathsf{mpk}_{i+1}^{(1)}, (C^{i+1}, K_1, \bot, \bot), r_1)$$

Functions $f_{n+1}^{(b)}$ output C^n directly.

To evaluate the above obfuscation on n-bit input $x[1]\dots x[n]$, the evaluator applies a sequence of FE decryptions $(c_{i,0}, c_{i,1}) = \mathsf{FE.Dec}(\mathsf{fsk}_i^{(x[i])}, c_{i-1,x[i]})$ for $1 \le i \le n$ where $c_{0,x[1]} = c_{x[1]}$. The output of each such decryption is an encryption of a partial evaluation $C(x[1]\dots x[i], \cdot)$ for $i < n$ and the output of the final decryption is the value $C(x[1]\dots x[n])$ in the clear. Note that the use of the PRG in each decryption function to generate random values effectively creates a PRF [GGM84] that the evaluation algorithm applies on all input prefixes after the tree cover to generate coins for FE encryption. However, because each input prefix needs exactly one set of pseudorandom coins, this construction is sufficient, and explicitly using a PRF, say for each input prefix, is not necessary.

The above construction so far handles only obfuscating a single circuit description, however, $\mathsf{Id}\mathcal{O}$ should work on any polynomial-size circuit assignment, and it can only depend on that circuit assignment. This means that it no longer suffices to encrypt C in the initial ciphertexts but we should rather include the fragment circuits corresponding to the tree cover assignment. It additionally poses challenges how to select which of the fragment circuit should be used for the evaluation on each input and also how to hide circuit assignment information, which is needed for the $\mathsf{Id}\mathcal{O}$ security. To do that, the obfuscation algorithm creates an indexed set of SKE ciphertexts, $\{Z_{i,j}^{(b)}\}_{i\in[n], j\in[l], b\in\{0,1\}}$, which we describe

below. These SKE ciphertexts will be hardcoded into the decryption functions. These ciphertexts will contain, under a layer of FE encryption, either the fragments' circuits or index values necessary to reach the correct fragment for each input.

We can express a circuit assignment as $(C^*, x_{\leq}||b)$, where we define the prefix length, $i = |x_{\leq}||b|$. For such an element, the algorithm randomly generates K_0 and K_1, and sets

$$(c^{(b)}_{i+1,0}, c^{(b)}_{i+1,1}) = \\ (\mathsf{FE.Enc}(\mathsf{mpk}^{(0)}_{i+1}, (C^*, K_0, \bot, \bot)), \mathsf{FE.Enc}(\mathsf{mpk}^{(1)}_{i+1}, (C^*, K_1, \bot, \bot)))$$

$\mathsf{Id}\mathcal{O}$ will permute the above FE ciphertext pairs by assigning them unique indices, j_b, and then encrypt them with SKE as follows:

$$Z^{(b)}_{i,j_b} = \mathsf{SKE.Enc}(\mathsf{sk}^{(b)}_{i,j_b}, (c^{(b)}_{i+1,0}, c^{(b)}_{i+1,1}))$$

These $Z^{(b)}_{i,j_b}$ values are hardcoded in each of the FE decryption keys in the obfuscation. In the above description of the obfuscation evaluation algorithm we were doing a sequence of FE decryptions which were incrementally evaluating C. When we have only fragment circuits, we can start using them only after we have progressed to corresponding input prefix. Thus the $\mathsf{Id}\mathcal{O}$ construction generates more ciphertexts which will be hardcoded in the decryption keys and will help navigate to the correct Z values for the evaluation. At a high level these values will correspond to nodes above the fragment tree cover and will contain indexes to each of their children. During evaluation they will be used to follow a path determined by the input prefix to reach the correct cover fragment circuit for evaluation. More formally these ciphertexts will be of the form:

$$(c^{(p)}_{i,0}, c^{(p)}_{i,1}) = \\ (\mathsf{FE.Enc}(\mathsf{mpk}^{(0)}_{i}, (\bot, \bot, j_0, \mathsf{sk}^{(0)}_{i,j_0})), \mathsf{FE.Enc}(\mathsf{mpk}^{(1)}_{i}, (\bot, \bot, j_1, \mathsf{sk}^{(1)}_{i,j_1}))),$$

where p will be a bit value corresponding to the last bit of the input prefix leading to the node corresponding to the ciphertexts. These FE ciphertexts are also encrypted with SKE and are shuffled together with the encryptions containing the circuit fragments before being hardcoded in the FE decryption keys.

The functionality of $f^{(b)}_i$ for $b \in \{0, 1\}$ is extended to handle the two additional possible input messages encrypted in the $Z^{(b)}_{i,j_b}$ values. The messages of the form $(C, K, \bot, \bot)$ are used for partial evaluation as above. On input messages with indexes of the form $(\bot, \bot, j_b, \mathsf{sk}^{(b)}_{i,j_b})$ the function $f^{(b)}_i$ outputs $\mathsf{SKE.Dec}(\mathsf{sk}^{(b)}_{i,j_b}, Z^{(b)}_{i,j_b})$.

Thus the $\mathsf{d}\mathcal{O}$ construction hardcodes in its FE decryption keys a number of ciphertexts proportional to the number of fragments in its circuit assignments. In the case when there are repeating circuits among the fragments corresponding to the assignments of the tree cover, the $\mathsf{d}\mathcal{O}$ still needs to use separate ciphertexts for each of them. That ensures that only ciphertexts corresponding to nodes on

the tree cover depend on the actual obfuscated circuit, while all nodes above the tree cover of the assignment contain only generic index information independent of the actual obfuscated circuit.

C Obfuscating a Circuit Assignment with Repeated Fragments

Construction 1 (Repeated Subcircuit $\mathsf{d}\mathcal{O}$) *Let* $\mathsf{FE} = \{\mathsf{FE.Gen}, \mathsf{FE.Enc}, \mathsf{FE.KeyGen}, \mathsf{FE.Dec}\}$ *be a functional encryption scheme and* $\mathsf{PRF} = (\mathsf{PRF.Gen}, \mathsf{PRF.Eval})$ *be a pseudorandom function. We construct a* $\mathsf{rescuei}\mathcal{O}$ *scheme that has an obfuscation algorithm shown in Algorithm 1 and evaluation algorithm shown in Algorithm 6. We define our* $\mathsf{rescuei}\mathcal{O}$ *obfuscation for circuits* C *with the following restriction: there exist a polynomial-size* l *and a level* $i^* \leq n$ *such that* $|\bigcup_{i=0}^{i^*} \mathsf{LevelAssignment}(C, i)| \leq l$.

Algorithm 1 generation of hardcoded parameters of obfuscated circuit

procedure $\mathsf{rescuei}\mathcal{O}(1^\lambda, C, i^*, l, s)$ ▷ C: the circuit to obfuscate, i^*: the level whose fragments to hardcode. l: the maximum number of fragments we will hardcode. s: the maximum size of each fragment. This implicitly limits the size of the messages and ciphertexts.

  for $i \in [n+1], b \in \{0,1\}$ **do**

    $(\mathsf{msk}_i^{(b)}, \mathsf{mpk}_i^{(b)}) \leftarrow \mathsf{FE.Gen}(1^\lambda)$

  end for

  for $i \in [n], b \in \{0,1\}$ **do**

    for $j \in [i^*]$ **do** ▷ Generate SKE ciphertexts that might hold fragments' FE ciphertexts. They will be hardcoded into the appropriate FE decryption functions

      $\mathsf{sk}_{i,j}^{(b)} \leftarrow \mathsf{SKE.KeyGen}(1^\lambda)$

      $Z_{i,j}^{(b)} \leftarrow \mathsf{SKE.Enc}(\mathsf{sk}_{i,j}^{(b)}, (\bot, \bot))$ ▷ But for now, only hold dummy information.

    end for

    $Z_i^{(b)} \leftarrow \{Z_{i,j}^{(b)}\}_{j \in [l]}$

  end for

  for $b \in \{0,1\}$ **do**

    Uniformly, at random, choose an injective function, $\pi_{C,b} : \bigcup_{i=0}^{i^*} \mathsf{LevelAssignment}(C, i) \to [l]$ ▷ Randomly map FE ciphertexts to SKE ciphertexts

  end for

$(c_0, c_1) \leftarrow \mathsf{rscCGen}(C, i^*, \pi_{C,0}, \pi_{C,1})$ ▷ Use Algorithm 2 to generate initial FE ciphertexts, and populate SKE ciphertexts.

for $i \in [n+1], b \in \{0,1\}$ **do**

$\mathsf{fsk}_i^{(b)} \leftarrow \mathsf{FE.KeyGen}(\mathsf{msk}_i^{(b)}, f_i^{(b)})$ ▷ Generate FE decryption keys that will perform Algorithms 3 and 5 during evaluation.

end for

return $((c_0, c_1), \{\mathsf{fsk}_i^{(b)}\}_{i \in [1,n+1], b \in \{0,1\}})$

end procedure

Algorithm 2 generate (c_0, c_1)

procedure $\mathsf{rscCGen}(C, i^*, \pi_{C,0}, \pi_{C,1})$ ▷ C: the circuit to obfuscate. i^*: the farthest level for which we want to hardcode ciphertexts. Subcircuits generated by inputs of length i^* will be included in the ciphertexts, while fragments with shorter inputs will only have their corresponding ciphertexts contain an index and SKE key to find and decrypt the SKE ciphertext in the next level. $(\pi_{C,0}, \pi_{C,1})$: determine which SKE ciphertext will hold an FE ciphertext

$K_{i^*} \leftarrow \mathsf{PRF.Gen}()$

for i from i^* to 0, $p \in \{0,1\}$, $C^* \in \mathsf{LevelAssignment}(C, i)$ **do** ▷ Generate FE ciphertexts for each fragment, and put it in a random SKE ciphertext. The plaintext data will give us the index and SKE key to decrypt the SKE ciphertext in the next level to get the next FE ciphertext, or the partial circuit itself, and a random seed, to generate an FE pseudorandom ciphertext in the same format, for the next fragments. ▷ p represents the last input bit used to reach the current fragment.

$j_{C^*}^{(p)} \leftarrow \pi_{C,p}(C^*)$

for $b \in \{0,1\}$ **do**

if $i < i^*$ **then**

$j_{C^*(b,\cdot)}^{(b)} \leftarrow \pi_{C,b}(C^*(b,\cdot))$

$m_{C^*,b}^{(p)} \leftarrow (\perp, \perp, j_{C^*(b,\cdot)}^{(b)}, sk^{(b)}_{i+1, j^{(b)}_{C^*(b,\cdot)}})$

else

$m_{C^*,b}^{(p)} \leftarrow (C^*, K_i, \perp, \perp)$

end if

$c_{C^*,b}^{(p)} \leftarrow \mathsf{FE.Enc}(\mathsf{mpk}_{i+1}^{(b)}, m_{C^*,b}^{(p)})$

end for

if $i > 0$ **then**

$Z^{(p)}_{i, j^{(p)}_{C^*}} \leftarrow \mathsf{SKE.Enc}(\mathsf{sk}^{(p)}_{i, j^{(p)}_{C^*}}, (c_{C^*,0}^{(p)}, c_{C^*,1}^{(p)}))$

end if

end for

return $(c_{C,0}^{(0)}, c_{C,1}^{(0)})$ ▷ Return the initial FE ciphertexts at the top level, which correspond to the original circuit, and the empty input. Since there is no previous input bit, the superscript of 0 is arbitrary.

end procedure

Algorithm 3 intermediate evaluation, for $i \in [n]$, using input bits

procedure $f_i^{(b^*)}(C, K, j, \mathsf{sk})$ ▷ Either (C, K) are valid, which means that this function will generate new FE ciphertexts corresponding $C(b^*, \cdot, \cdot)$, and using K to generate pseudorandom coins for encryption, and the next level's seeds, or (j, sk) are valid, in which case this algorithm will retrieve the FE ciphertexts from the SKE ciphertexts. ▷ Hardcoded: b^*: the next input bit to plug in; $Z_{i+1}^{(b^*)}$: the hardcoded SKE ciphertexts for the next level. If needed, they will be decrypted to the next level's FE ciphertexts; $(\mathsf{mpk}_{i+1}^{(0)}, \mathsf{mpk}_{i+1}^{(1)})$: the decryption keys for generating FE ciphertexts

  if $j \neq \perp$ **then**
    return $\mathsf{SKE.Dec}(\mathsf{sk}, Z_{i+1,j}^{(b^*)})$
  else
    $C^* \leftarrow C(b^*, \cdot)$
    return $f_{i,1}^{(b^*)}(C^*, K)$ ▷ Using Algorithm 4
  end if
end procedure

Algorithm 4 intermediate evaluation in the second case

procedure $f_{i,1}^{(b^*)}(C^*, K)$ ▷ C^*: The generic circuit. K: The PRF key for generating pseudorandom coins, to simulate the generation of the FE ciphertext, as if it were at level i^*. ▷ Hardcoded: b^*, $\mathsf{mpk}_{i+1}^{(0)}, \mathsf{mpk}_{i+1}^{(1)}$

  $K^+ \leftarrow \mathsf{PRF.Gen}(\mathsf{PRF.Eval}_K(0))$ ▷ Generate the randomness seed pseudorandomly. Using the PRF key generation algorithm handles cases where the PRF key is not a uniformly random string. As long as the old PRF key, K, is the same, all of the PRF keys generated at this level are the same. This allows us to generate consistent pseudorandom coins for the next level.

  for $b \in \{0, 1\}$ **do**
    $r_{i,b} \leftarrow \mathsf{PRF.Eval}_K((C^*, b^*, b))$ ▷ Generate pseudorandom coins for encryption. Using PRF, instead of a PRG (which is done in $\mathsf{Id}\mathcal{O}$), means that the pseudorandom coins are consistent for any ciphertext pairs generated using the same C^* and b^*, no matter what the previous bits were.
    $c_b \leftarrow \mathsf{FE.Enc}(\mathsf{mpk}_{i+1}^{(b)}, (C^*, K^+, \perp, \perp); r_{i,b})$
  end for
  return (c_0, c_1)
end procedure

Algorithm 5 final evaluation, using input bits

procedure $f_{n+1}^{(b^*)}(C, K, j, \mathsf{sk})$ ▷ Only C is necessary, and contains the output already. The rest are there for interface consistency.

  return $C(\epsilon)$
end procedure

Since in the next section we will need to pass an obfuscated *circuit*, or its partial evaluations, into the $\mathsf{Id}\mathcal{O}$ algorithm, instead of having $\mathsf{rescuei}\mathcal{O}$ output the

Algorithm 6 full evaluation of an obfuscation

procedure $\mathsf{rescuei}\mathcal{O}.\mathsf{Eval}(x, ((c_0, c_1), \{\mathsf{fsk}_i^{(b)}\}_{i\in[1,n+1],b\in\{0,1\}}))$ ▷ x: the input, for which we want to calculate $C(x)$. $((c_0, c_1), \{\mathsf{fsk}_i^{(b)}\}_{i\in[1,n+1],b\in\{0,1\}})$: the parameters that $\mathsf{rescuei}\mathcal{O}$ output.
  for $b \in \{0, 1\}$ **do**
    $c_{0,b} \leftarrow c_b$
  end for
  for $i \in [n]$ **do**
    $(c_{i,0}, c_{i,1}) \leftarrow \mathsf{FE.Dec}(\mathsf{fsk}_i^{(x_i)}, c_{i-1,x_i})$
  end for
  return $\mathsf{FE.Dec}(\mathsf{fsk}_{n+1}^{(0)}, c_{n,0})$ ▷ Using the index 0 is arbitrary, and we can also use 1.
end procedure

hardcoded parameters, $((c_0, c_1), \{\mathsf{fsk}_i^{(b)}\}_{i\in[1,n+1],b\in\{0,1\}})$, for the obfuscation, we can explicitly define $\mathsf{rescuei}\mathcal{O}$ to output a circuit, $\tilde{C}_{((c_0,c_1),\{\mathsf{fsk}_i^{(b)}\}_{i\in[1,n+1],b\in\{0,1\}})}$. We also define Algorithm 8, an alternative definition of the evaluation Algorithm 6 where the selection of the FE ciphertext and decryption key are explicitly expressed as per-bit boolean statements using the construct in Algorithm 7.

Algorithm 7 bit selection circuit

procedure $\mathsf{Select}(b, x_0, x_1)$ ▷ b: selector bit. x_0, x_1: values to select
  return $(\bar{b} \wedge x_0) \vee (b \wedge x_1)$ ▷ Outputs x_b
end procedure

C.1 Properties of $\mathsf{rescuei}\mathcal{O}$

We will show that increasing the level, i^*, is not noticeable, as long as the total number of fragments stays below a polynomial, l:

Lemma 3 *Assuming a single-key compact* FE *with selective polynomial security, we have*

$$\mathsf{rescuei}\mathcal{O}(1^\lambda, C, i^*, l, s) \overset{c}{\approx} \mathsf{rescuei}\mathcal{O}(1^\lambda, C, i^* + 1, l, s)$$

for every circuit C *of size at most* s *for which there exists a polynomial* l *such that* $\forall i^* < n$ *we have* $|\bigcup_{i=0}^{i^*+1} \mathsf{LevelAssignment}(C, i)| \leq l$, *and* $\forall C^* \in \bigcup_{i=0}^{i^*+1} \mathsf{LevelAssignment}(C, i)$ $|C^*| \leq s$.

We need to prove both efficiency and security.

Efficiency Efficiency follows from the compactness of the FE scheme. In particular, the encryption algorithm is independent of the size of the function for which there exists a decryption circuit. The runtime that of FE encryption is only a polynomial function of the size of the fragment, the security parameter, which determines the size of the random seed and the SKE key, and a logarithm

Algorithm 8 components of the obfuscation of C

procedure $\tilde{C}_{((c_0,c_1),\{\mathsf{fsk}_i^{(b)}\}_{i\in[1,n+1],b\in\{0,1\}})}(x)$ ▷ x is the full input to C
  for $b \in \{0,1\}$ **do**
    $c_{0,b} \leftarrow c_b$ ▷ Initialize ciphertexts
  end for
  for $i \in [1,n]$ **do**
    for $w \in [|c_{i-1,0}|]$ **do** ▷ Select each bit of the ciphertext and FE decryption key for the decryption in each iteration. The choice of using the length of $c_{i-1,0}$ is arbitrary, since $|c_{i-1,0}| = |c_{i-1,1}|$
      $\tilde{c}_i[w] \leftarrow \mathsf{Select}(x[i], c_{i-1,0}[w], c_{i-1,1}[w])$
      $\tilde{\mathsf{fsk}}_i[w] \leftarrow \mathsf{Select}(x[i], \mathsf{fsk}_i^{(0)}[w], \mathsf{fsk}_i^{(1)}[w])$
    end for
    $(c_{i,0}, c_{i,1}) \leftarrow \mathsf{FE.Dec}(\tilde{\mathsf{fsk}}_i, \tilde{c}_i)$
  end for
  return $\mathsf{FE.Dec}(\mathsf{fsk}_{n+1}^{(0)}, c_{n,0})$ ▷ Output the final calculation
end procedure

of l. Since the FE encryption algorithm's runtime is a polynomial of the message size and the security parameter, then the runtime is a polynomial of the aforementioned factors. Likewise the runtime of SKE encryption is only a polynomial of the security parameter and the message, which is an FE ciphertext, so its runtime is also polynomial.

Security In the formal security proof in Appendix D.1, we change rscCGen in 4 main hybrids, by adding to $Z_{i^*+1}^{(b)}$ for $b \in \{0,1\}$, the decryption of the FE ciphertexts from level i^*, which we can do by SKE security, have the FE plaintexts at level i^* point to the newly-added data, which we can do by FE security. The data we put in level i^*+1 are actually FE ciphertexts, but with pseudorandom coins. So in the last hybrid, we replace the pseudorandom coins with fresh randomness, which we can do by PRF security.

For all subsequent security proofs, we will use hybrids over an arbitrary ordering of fragments γ, starting from 1, and subject to the following constraint, which requires that fragments formed by plugging in shorter inputs come before fragments formed by plugging in longer inputs. For any fragment, C^*, let $\mathsf{PreLen}(C^*)$ be the number of input bits that are plugged into C to get C^*. Then the ordering has the following constraint:

$$\forall C_0^*, C_1^* : \mathsf{PreLen}(C_0^*) < \mathsf{PreLen}(C_1^*) \Rightarrow \gamma(C_0^*) < \gamma(C_1^*)$$

While Lemma 3 is the most general one that we can prove about $\mathsf{rescuei}\mathcal{O}$, it is actually enough for some pairs of consistent circuits:

Theorem 2 *Assuming a single-key, compact* FE *with polynomial selectively security, for any two circuits, C_0, C_1 are consistent according to Definition 7 for some level i^*, and for some polynomials l, and s, then*

$$\mathsf{rescuei}\mathcal{O}(1^\lambda, C_0, 0, l, s) \overset{c}{\approx} \mathsf{rescuei}\mathcal{O}(1^\lambda, C_1, 0, l, s).$$

We will prove this theorem in Appendix D.2, and show that the previous example, which could not be obfuscated by polynomially secure $\mathsf{d}\mathcal{O}$, can be obfuscated by $\mathsf{rescuei}\mathcal{O}$.

D Proofs of $\mathsf{rescuei}\mathcal{O}$ security

D.1 Proof of Lemma 3

Proof. 1. $\mathcal{H}_1$: Use the original $\mathsf{rscCGen}(C, i^*, \pi_{C,0}, \pi_{C,1})$.

2. $\mathcal{H}_2$: Put the FE decryptions of the ciphertexts from level i^* into level i^*+1. In other words, we have up to l subhybrids, between which we change another entry in $Z^{(b)}_{i^*+1}$ for $b \in \{0,1\}$. For $j \in [0,l]$, and $\beta \in \{0,1\}$, define $\mathcal{H}_{2,j,\beta}$, so that it uses $\mathsf{rscCGen}_2(C, i^*, \pi_{C,0}, \pi_{C,1}, j, \beta)$, which is defined in Algorithm 9. So $\mathcal{H}_{2,0,1}$ is identical to $\mathcal{H}_1$, and we finish at $\mathcal{H}_{2,l,1}$. We show that $\forall j \in [l]$, $\mathcal{H}_{2,j,\beta}$ is indistinguishable from $\mathcal{H}_{2,j-1,1}$ if $\beta = 0$, and from $\mathcal{H}_{2,j,0}$ if $\beta = 1$. If $\nexists C^\dagger \in \mathsf{LevelAssignment}(C, i^*+1)$, so that $\gamma(C^\dagger) = j$, then the two consecutive subhybrids are identical. Otherwise, we prove indistinguishability using SKE security. Since the total number of subhybrids is polynomially bounded, the number of non-identical subhybrids is polynomially bounded, so proving indistinguishability for non-identical subhybrids suffices.
   For contradiction, assume that there exists an adversary, $\mathcal{A}$, can distinguish between $\mathcal{H}_{2,j,\beta}$ and the previous hybrid. Then we can construct an adversary, $\mathcal{A}'$, which breaks SKE security as follows:
   (a) Submit challenge plaintexts, $m_0 = \perp$, and $m_1 = f^{(p)}_{i^*+1,1}(C^\dagger, K_{i^*})$.
   (b) Receive the challenge ciphertext, $c^\dagger$.
   (c) Run $\mathcal{A}$, with the following variation of $\mathsf{rscCGen}$, denoted by $\mathsf{rscCGen}^\dagger_{2,j,\beta}(C, i^*, \pi_{C,0}, \pi_{\pi_{,,1}}, j, \beta)$, as shown in Algorithm 10.
   (d) Return whatever $\mathcal{A}$ returns.
   The only place where the symmetric key, sk, could have appeared, is in $Z^{(b)}_{i^*}$, but those don't contain any symmetric keys, yet. So we can rely on SKE security. If $c^\dagger$ is an encryption of m_1, then $\mathsf{rscCGen}^\dagger_2$ followed $\mathcal{H}_{2,j,\beta}$; otherwise, it followed the previous hybrid.

3. $\mathcal{H}_3$: For all of the FE plaintexts in level i^*, replace them with indexes to the next level. In other words, we have up to l subhybrids, between which we change another entry in $m^{(p)}_{C^*,b}$ for $C^* \in \mathsf{LevelAssignment}(C, i^*)$, $b \in \{0,1\}$, and $p \in \{0,1\}$. For $\eta \in [3, 4 \times l + 3,]$, which enumerates the fragments, and two bits, define $\mathcal{H}_{3,\eta}$, so that it uses $\mathsf{rscCGen}_3(C, i^*, \pi_{C,0}, \pi_{C,1}, \eta)$, which is defined as shown in Algorithm 11. So $\mathcal{H}_{3,3}$ is identical to $\mathcal{H}_{2,l,1}$, and we finish at $\mathcal{H}_{3,4\times l+3}$. We show that $\forall \eta > 1$, $\mathcal{H}_{3,\eta}$ is indistinguishable from $\mathcal{H}_{3,\eta-1}$. If $\nexists C^\dagger \in \mathsf{LevelAssignment}(C, i^*+1)$, so that $\eta \in [\gamma(C^\dagger), \gamma(C^\dagger)+3]$, then the two consecutive sub-hybrids are identical. Otherwise, it is sufficient that we prove indistinguishability using FE security. Again, the number of non-identical subhybrids is polynomially bounded, since the total number of subhybrids is polynomially bounded.

For contradiction, assume that there exists an adversary, $\mathcal{A}$, can distinguish between $\mathcal{H}_{3,\eta}$ and $\mathcal{H}_{3,\eta-1}$. Then we can construct an adversary, $\mathcal{A}'$, which breaks FE security as follows:

(a) Let $\beta \leftarrow \eta \mod 2$, and $j^{(\beta)}_{C^\dagger(\beta,\cdot)} \leftarrow \pi_{C,\beta}(C^\dagger(\beta,\cdot))$

(b) Submit $m_0 = (C^\dagger, K_{i^*}, \perp, \perp)$ and $m_1 = (\perp, \perp, j^{(\beta)}_{C^\dagger(\beta,\cdot)}, \mathsf{sk}^{(\beta)}_{i^*+1,j^{(\beta)}_{C^\dagger(\beta,\cdot)}})$.

(c) Receive the master public key, mpk.

(d) Request and receive the FE decryption key, fsk, for function $f^{(\beta)}_{i^*+1}$.

(e) Receive the challenge ciphertext, $c^\dagger$.

(f) Run $\mathcal{A}$ with a modified $\mathsf{rescuei}\mathcal{O}$ algorithm, Algorithm 12. The master secret key, msk, does not appear anywhere in the obfuscation; only the master public key, mpk, and decryption key, fsk, which $\mathcal{A}'$ receives appear. If $c^\dagger$ is an encryption of m_0, then the hybrid is identical to $\mathcal{H}_{3,\eta-1}$, while if $c^\dagger$ is an encryption of m_1, then the hybrid is identical to $\mathcal{H}_{3,\eta}$. Moreover, in the former case, since the third and fourth arguments are not specified, $f^{(\beta)}_{i^*+1}(C^\dagger, K_{i^*}, \perp, \perp) = f^{(\beta)}_{i^*+1,1}(C^\dagger, K_{i^*})$. On the other hand, in the latter case:

$$f^{(\beta)}_{i^*+1}(\perp, \perp, j^{(\beta)}_{C^\dagger(\beta,\cdot)}, \mathsf{sk}^{(\beta)}_{i^*+1,j^{(\beta)}_{C^\dagger(\beta,\cdot)}}) = (c^{(\beta)}_{C^\dagger,0}, c^{(\beta)}_{C^\dagger,1})$$

Due to hybrid $\mathcal{H}_2$, this is also equal to $f^{(\beta)}_{i^*+1,1}(C^\dagger, K_{i^*})$. Therefore, if $\mathcal{A}$ can distinguish between $\mathcal{H}_{3,\eta-1}$ and $\mathcal{H}_{3,\eta}$, then $\mathcal{A}'$ breaks FE security.

4. $\mathcal{H}_4$: Generate K_{i^*+1} with fresh randomness, and replace every call to $f^{(\beta)}_{i^*+1,1}$, with FE encryption using fresh randomness. In other words, use the version of $\mathsf{rscCGen}$ in Algorithm 14.
This is identical to $\mathsf{rscCGen}(C, i^*+1, \pi_{C,0}, \pi_{C,1})$, except that the computations for level i^*+1 are done outside of the old loops, and the generation of K_{i^*} is replaced with K_{i^*+1}, where the former is not used in the algorithm.
To show that this hybrid is indistinguishable from $\mathcal{H}_{3,4\times l+3}$, we use the security of PRF. Assume for contradiction that there exists an adversary, $\mathcal{A}$, that can distinguish between the two hybrids. Then there exists an adversary $\mathcal{A}'$, which can distinguish between a random function, and a pseudorandom one:

(a) Receive oracle access to function F.

(b) Obfuscate the circuit with Algorithms 15 and 16. If F is a PRF, with key K, we can assign it to K_{i^*}, to yield hybrid $\mathcal{H}_{3,4\times l+3}$. Otherwise, inside $f^{(p)}_{i^*+1,4,1}$ generates K' identically to K_{i^*+1}, and $\forall C^* \in \mathsf{LevelAssignment}(C, i^*+1)$, $p \in \{0,1\}$, $r_{i^*+1,0}$ and $r_{i^*+1,1}$ are generated using unique tuple pairs, $((C^*, i^*+1, 0), (C^*, i^*+1, 1))$, so the randomness for the FE encryption is generated uniformly at random for each call to $f^{(p)}_{i^*+1,4,1}$ That means that if $F()$ is a random function, $\mathcal{A}$ is playing $\mathcal{H}_4$.

Algorithm 9 rscCGen, but for hybrid $\mathcal{H}_2$

procedure $\mathsf{rscCGen}_2(C, i^*, \pi_{C,0}, \pi_{C,1}, j, \beta)$ j: determines which fragments at level i^*+1 might now have hardcoded ciphertexts. β: determines which ciphertex pair for the fragment corresponding to the hybrid boundary, j, will contain valid values.

$K_{i^*} \leftarrow \mathsf{PRF.Gen}()$

for i from $i^*\boxed{+1}$ to 0, $p \in \{0,1\}$, $C^* \in \mathsf{LevelAssignment}(C, i)$ **do**

$j^{(p)}_{C^*} \leftarrow \pi_{C,p}(C^*)$

if $i \leq i^*$ **then** ▷ Perform normal computation of the existing ciphertexts.

for $b \in \{0,1\}$ **do**

if $i < i^*$ **then**

$j^{(b)}_{C^*(b,\cdot)} \leftarrow \pi_{C,b}(C^*(b,\cdot))$

$m^{(p)}_{C^*,b} \leftarrow (\perp, \perp, j^{(b)}_{C^*(b,\cdot)}, sk^{(b)}_{i+1,j^{(b)}_{C^*(b,\cdot)}})$

else

$m^{(p)}_{C^*,b} \leftarrow (C^*, K_i, \perp, \perp)$

end if

$c^{(p)}_{C^*,b} \leftarrow \mathsf{FE.Enc}(\mathsf{mpk}^{(b)}_{i+1}, m^{(p)}_{C^*,b})$

end for

else if $\boxed{\gamma(C^*) < j \vee (\gamma(C^*) = j \wedge p \leq \beta)}$ **then** ▷ The hybrid will gradually add valid ciphertexts for level i^*+1. However, because there are no *ske* keys for this level, they will never be used, and the changes will not be apparent, yet. But later, the fact that they contain the FE decryption of the FE ciphertexts of the previous level, the insertion of these ciphertexts will make the changes to level i^*, indistinguishable.

$\boxed{(c^{(p)}_{C^*,0}, c^{(p)}_{C^*,1}) \leftarrow f^{(p)}_{i,1}(C^*, K_{i^*})}$

else ▷ The rest are still dummy values.

$\boxed{(c^{(p)}_{C^*,0}, c^{(p)}_{C^*,1}) \leftarrow (\perp, \perp)}$

end if

if $i > 0$ **then**

$Z^{(p)}_{i,j^{(p)}_{C^*}} \leftarrow \mathsf{SKE.Enc}(\mathsf{sk}^{(p)}_{i,j^{(p)}_{C^*}}, (c^{(p)}_{C^*,0}, c^{(p)}_{C^*,1}))$

end if

end for

return $(c^{(0)}_{C,0}, c^{(0)}_{C,1})$

end procedure

Algorithm 10 $\mathsf{rscCGen}$ with challenge ciphertext for $\mathcal{H}_{2,j,\beta}$

procedure $\mathsf{rscCGen}_2^{\dagger}(C, i^*, \pi_{C,0}, \pi_{C,1}, j, \beta)$ ▷ j and β, determine where to put the challenge ciphertext.

$K_{i^*} \leftarrow \mathsf{PRF.Gen}()$
for i from i^*+1 to 0, $p \in \{0,1\}$, $C^* \in \mathsf{LevelAssignment}(C, i)$ **do**
$j_{C^*}^{(p)} \leftarrow \pi_{C,p}(C^*)$
if $i \leq i^*$ **then**
for $b \in \{0,1\}$ **do**
if $i < i^*$ **then**
$j_{C^*(b,\cdot)}^{(b)} \leftarrow \pi_{C,b}(C^*(b,\cdot))$
$m_{C^*,b}^{(p)} \leftarrow (\perp, \perp, j_{C^*(b,\cdot)}^{(b)}, sk_{i+1,j_{C^*(b,\cdot)}^{(b)}}^{(b)})$
else
$m_{C^*,b}^{(p)} \leftarrow (C^*, K_i, \perp, \perp)$
end if
$c_{C^*,b}^{(p)} \leftarrow \mathsf{FE.Enc}(\mathsf{mpk}_{i+1}^{(b)}, m_{C^*,b}^{(p)})$
end for
else if $\gamma(C^*) < j \vee (\gamma(C^*) = j \wedge p \leq \beta)$ **then**
$(c_{C^*,0}^{(p)}, c_{C^*,1}^{(p)}) \leftarrow f_{i,1}^{(p)}(C^*, K_{i^*})$
else
$(c_{C^*,0}^{(p)}, c_{C^*,1}^{(p)}) \leftarrow (\perp, \perp)$
end if
if $i > 0$ **then**
if $\boxed{i = i^*+1 \wedge \gamma(C^*) = j \wedge p = \beta}$ **then**
$\boxed{Z_{i,j_{C^*}^{(p)}}^{(p)} \leftarrow c^{\dagger}}$
else
$Z_{i,j_{C^*}^{(p)}}^{(p)} \leftarrow \mathsf{SKE.Enc}(\mathsf{sk}_{i,j_{C^*}^{(p)}}^{(p)}, (c_{C^*,0}^{(p)}, c_{C^*,1}^{(p)}))$
end if
end if
end for
return $(c_{C,0}^{(0)}, c_{C,1}^{(0)})$
end procedure

Algorithm 11 rscCGen, but for hybrid $\mathcal{H}_3$

procedure $\mathsf{rscCGen}_3(C, i^*, \pi_{C,0}, \pi_{C,1}, \eta)$ ▷ η: determines which FE plaintexts in level i^* will contain indexes, rather than fragments.

$K_{i^*} \leftarrow \mathsf{PRF.Gen}()$

for i from i^*+1 to 0, $p \in \{0,1\}$, $C^* \in \mathsf{LevelAssignment}(C, i)$ **do**

$j^{(p)}_{C^*} \leftarrow \pi_{C,p}(C^*)$

if $i \leq i^*$ **then**

for $b \in \{0,1\}$ **do**

if $i < i^*$ $\boxed{\vee 4 \times \gamma(C^*) + 2 \times p + b \leq \eta}$ **then**

$j^{(b)}_{C^*(b,\cdot)} \leftarrow \pi_{C,b}(C^*(b,\cdot))$

$m^{(p)}_{C^*,b} \leftarrow (\perp, \perp, j^{(b)}_{C^*(b,\cdot)}, sk^{(b)}_{i+1,j^{(b)}_{C^*(b,\cdot)}})$

else

$m^{(p)}_{C^*,b} \leftarrow (C^*, K_i, \perp, \perp)$

end if

$c^{(p)}_{C^*,b} \leftarrow \mathsf{FE.Enc}(\mathsf{mpk}^{(b)}_{i+1}, m^{(p)}_{C^*,b})$

end for

else

$(c^{(p)}_{C^*,0}, c^{(p)}_{C^*,1}) \leftarrow f^{(p)}_{i,1}(C^*, K_{i^*})$

end if

if $i > 0$ **then**

$Z^{(p)}_{i,j^{(p)}_{C^*}} \leftarrow \mathsf{SKE.Enc}(\mathsf{sk}^{(p)}_{i,j^{(p)}_{C^*}}, (c^{(p)}_{C^*,0}, c^{(p)}_{C^*,1}))$

end if

end for

return $(c^{(0)}_{C,0}, c^{(0)}_{C,1})$

end procedure

Algorithm 12 main obfuscation algorithm for games to distinguish between subhybrids in hybrid $\mathcal{H}_3$

procedure $\mathsf{rescuei}\mathcal{O}_3^{\dagger}(1^\lambda, C, i^*, l, s, \eta)$ ▷ η: the index of the game to distinguish subhybrids, which determines where the keys available to the adversary are put.
  for $i \in [n+1], b \in \{0,1\}$ **do**
    if $\boxed{i = i^* \wedge b = \beta}$ **then**
      $\boxed{\mathsf{mpk}_i^{(b)} \leftarrow \mathsf{mpk}}$
    else
      $(\mathsf{msk}_i^{(b)}, \mathsf{mpk}_i^{(b)}) \leftarrow \mathsf{FE.Gen}(1^\lambda)$
    end if
  end for
  for $i \in [n], b \in \{0,1\}$ **do**
    for $j \in [i^*]$ **do** ▷ Generate SKE ciphertexts that might hold fragments' FE ciphertexts. They will be hardcoded into the appropriate FE decryption functions
      $\mathsf{sk}_{i,j}^{(b)} \leftarrow \mathsf{SKE.KeyGen}(1^\lambda)$
      $Z_{i,j}^{(b)} \leftarrow \mathsf{SKE.Enc}(\mathsf{sk}_{i,j}^{(b)}, (\bot, \bot))$ ▷ But for now, only hold dummy information.
    end for
    $Z_i^{(b)} \leftarrow \{Z_{i,j}^{(b)}\}_{j \in [l]}$
  end for
  for $b \in \{0,1\}$ **do**
    Uniformly, at random, choose an injective function, $\pi_{C,b} : \bigcup_{i=0}^{i^*} \mathsf{LevelAssignment}(C, i) \to [l]$ ▷ The fragments' FE ciphertexts will be placed in random SKE ciphertexts
  end for
  $(c_0, c_1) \leftarrow \mathsf{rscCGen}(C, i^*, \pi_{C,0}, \pi_{C,1})$ ▷ Use Algorithm 13 to generate initial FE ciphertexts, and populate SKE ciphertexts.
  for $i \in [n+1], b \in \{0,1\}$ **do**
    if $\boxed{i = i^* \wedge b = \beta}$ **then**
      $\boxed{\mathsf{fsk}_i^{(b)} \leftarrow \mathsf{fsk}}$
    else
      $\mathsf{fsk}_i^{(b)} \leftarrow \mathsf{FE.KeyGen}(\mathsf{msk}_i^{(b)}, f_i^{(b)})$ ▷ Generate FE decryption keys that will perform Algorithms 3 and 5 during evaluation.
    end if
  end for
  return $((c_0, c_1), \{\mathsf{fsk}_i^{(b)}\}_{i \in [1, n+1], b \in \{0,1\}})$
end procedure

Algorithm 13 $\mathsf{rscCGen}$, but for the challenge in hybrid $\mathcal{H}_3$

procedure $\mathsf{rscCGen}_3^{\dagger}(C, i^*, \pi_{C,0}, \pi_{C,1}, \eta)$ ▷ η: determines where to place the challenge ciphertext

$K_{i^*} \leftarrow \mathsf{PRF.Gen}()$

for i from i^*+1 to 0, $p \in \{0,1\}$, $C^* \in \mathsf{LevelAssignment}(C, i)$ **do**

$j_{C^*}^{(p)} \leftarrow \pi_{C,p}(C^*)$

if $i \leq i^*$ **then**

for $b \in \{0,1\}$ **do**

if $i < i^* \vee 4 \times \gamma(C^*) + 2 \times p + b \leq \eta$ **then**

$j_{C^*(b,\cdot)}^{(b)} \leftarrow \pi_{C,b}(C^*(b,\cdot))$

$m_{C^*,b}^{(p)} \leftarrow (\perp, \perp, j_{C^*(b,\cdot)}^{(b)}, sk_{i+1,j_{C^*(b,\cdot)}^{(b)}}^{(b)})$

else

$m_{C^*,b}^{(p)} \leftarrow (C^*, K_i, \perp, \perp)$

end if

if $4 \times \gamma(C^*) + 2 \times p + b = \eta$ **then**

$\boxed{c_{C^*,b}^{(p)} \leftarrow c^{\dagger}}$

else

$c_{C^*,b}^{(p)} \leftarrow \mathsf{FE.Enc}(\mathsf{mpk}_{i+1}^{(b)}, m_{C^*,b}^{(p)})$

end if

end for

else

$(c_{C^*,0}^{(p)}, c_{C^*,1}^{(p)}) \leftarrow f_{i,1}^{(p)}(C^*, K_{i^*})$

end if

if $i > 0$ **then**

$Z_{i,j_{C^*}^{(p)}}^{(p)} \leftarrow \mathsf{SKE.Enc}(\mathsf{sk}_{i,j_{C^*}^{(p)}}^{(p)}, (c_{C^*,0}^{(p)}, c_{C^*,1}^{(p)}))$

end if

end for

return $(c_{C,0}^{(0)}, c_{C,1}^{(0)})$

end procedure

Algorithm 14 rscCGen, but for the challenge in hybrid $\mathcal{H}_4$

procedure $\mathsf{rscCGen}_4(C, i^*, \pi_{C,0}, \pi_{C,1})$

$\boxed{K_{i^*+1} \leftarrow \mathsf{PRF.Gen}()}$ ▷ K_{i^*+1} is generated like a fresh PRF key, *ie.* with implicit fresh randomness, rather than pseudorandomly from K_{i^*}, which is no longer used.

for i from i^*+1 to 0, $p \in \{0,1\}$, $C^* \in \mathsf{LevelAssignment}(C, i)$ **do**

$j^{(p)}_{C^*} \leftarrow \pi_{C,p}(C^*)$

if $i \leq i^*$ **then**

for $b \in \{0,1\}$ **do**

$j^{(b)}_{C^*(b,\cdot)} \leftarrow \pi_{C,b}(C^*(b,\cdot))$

$m^{(p)}_{C^*,b} \leftarrow (\bot, \bot, j^{(b)}_{C^*(b,\cdot)}, sk^{(b)}_{i+1,j^{(b)}_{C^*(b,\cdot)}})$

$c^{(p)}_{C^*,b} \leftarrow \mathsf{FE.Enc}(\mathsf{mpk}^{(b)}_{i+1}, m^{(p)}_{C^*,b})$

end for

else

for $b \in \{0,1\}$ **do** ▷ Generate the FE ciphertexts similarly to $f^{(p)}_{i,1}(C^*, K_{i^*})$, but with fresh randomness.

$\boxed{m^{(p)}_{C^*,b} \leftarrow (C^*, K_i, \bot, \bot)}$

$\boxed{c^{(p)}_{C^*,b} \leftarrow \mathsf{FE.Enc}(\mathsf{mpk}^{(b)}_{i+1}, m^{(p)}_{C^*,b})}$

end for

end if

if $i > 0$ **then**

$Z^{(p)}_{i,j^{(p)}_{C^*}} \leftarrow \mathsf{SKE.Enc}(\mathsf{sk}^{(p)}_{i,j^{(p)}_{C^*}}, (c^{(p)}_{C^*,0}, c^{(p)}_{C^*,1}))$

end if

end for

return $(c^{(0)}_{C,0}, c^{(0)}_{C,1})$

end procedure

Algorithm 15 $\mathsf{rscCGen}$, but for challenge for hybrid $\mathcal{H}_4$

procedure $\mathsf{rscCGen}_4^{\dagger}(C, i^*, \pi_{C,0}, \pi_{C,1})$

$\boxed{K_{i^*+1} \leftarrow \mathsf{PRF.Gen}(F(0))}$ ▷ The random coins used to generate the PRF key might be pseudorandom or explicitly random, depending on F

for i from i^*+1 to 0, $p \in \{0,1\}$, $C^* \in \mathsf{LevelAssignment}(C, i)$ **do**

$j_{C^*}^{(p)} \leftarrow \pi_{C,p}(C^*)$

if $i \leq i^*$ **then**

for $b \in \{0,1\}$ **do**

$j_{C^*(b,\cdot)}^{(b)} \leftarrow \pi_{C,b}(C^*(b,\cdot))$

$m_{C^*,b}^{(p)} \leftarrow (\perp, \perp, j_{C^*(b,\cdot)}^{(b)}, sk_{i+1,j_{C^*(b,\cdot)}^{(b)}}^{(b)})$

$c_{C^*,b}^{(p)} \leftarrow \mathsf{FE.Enc}(\mathsf{mpk}_{i+1}^{(b)}, m_{C^*,b}^{(p)})$

end for

else

$\boxed{(c_{C^*,0}^{(p)}, c_{C^*,1}^{(p)}) \leftarrow f_{i,4,1}^{(p)}(C^*, F)}$ ▷ Using the modified Algorithm 16, which will use pseudorandomness or fresh randomness depending on F

end if

if $i > 0$ **then**

$Z_{i,j_{C^*}^{(p)}}^{(p)} \leftarrow \mathsf{SKE.Enc}(\mathsf{sk}_{i,j_{C^*}^{(p)}}^{(p)}, (c_{C^*,0}^{(p)}, c_{C^*,1}^{(p)}))$

end if

end for

return $(c_{C,0}^{(0)}, c_{C,1}^{(0)})$

end procedure

Algorithm 16 $f_{i,1}^{(b^*)}$, but using unknown function for generating random coins.

procedure $f_{4\,i,1}^{(b^*)}(C^*, F)$ F: either a pseudorandom, or a random function. ▷ Hardcoded: b^*, $\mathsf{mpk}_{i+1}^{(0)}$, $\mathsf{mpk}_{i+1}^{(1)}$

$K^+ \leftarrow \mathsf{PRF.Gen}(F(0))$ ▷ Generate the next level's PRF key pseudorandomly or randomly. Note that it identical to how K_{i^*+1} is generated in Algorithm 15

for $b \in \{0,1\}$ **do**

$r_{i,b} \leftarrow F((C^*, b^*, b))$ ▷ Perform FE encryption using either pseudorandom or random coins. Even if $F()$ is random, we get the same random coins for the same (C^*, b^*)

$c_b \leftarrow \mathsf{FE.Enc}(\mathsf{mpk}_{i+1}^{(b)}, (C^*, K^+, \perp, \perp); r_{i,b})$

end for

return (c_0, c_1)

end procedure

D.2 Proof of Theorem 2

The proof of Theorem 2 follows from Lemma 3 and Lemma 4, which we prove next. We can switch between $\mathsf{rescuei}\mathcal{O}(1^\lambda, C_0, 0, l, s)$ and $\mathsf{rescuei}\mathcal{O}(1^\lambda, C_0, i^*, l, s)$, and between $\mathsf{rescuei}\mathcal{O}(1^\lambda, C_0, i^*, l, s)$ and $\mathsf{rescuei}\mathcal{O}(1^\lambda, C_0, 0, l, s)$, due to Lemma 3. So we only need to prove that we can switch between $\mathsf{rescuei}\mathcal{O}(1^\lambda, C_0, i^*, l, s)$, and $\mathsf{rescuei}\mathcal{O}(1^\lambda, C_0, i^*, l, s)$, which we do using consisntency.

Lemma 4 *If two circuits, C_0, C_1 are consistent according to Definition 7 for some level i^*, and for some polynomials l, and s, then*

$$\mathsf{rescuei}\mathcal{O}(1^\lambda, C_0, i^*, l, s) \overset{s}{\equiv} \mathsf{rescuei}\mathcal{O}(1^\lambda, C_1, i^*, l, s).$$

The proof of this lemma is similar to a special case of the proof of Lemma 7, which we will show in Appendix E. The difference is that in this lemma, there is no explicit $\mathsf{d}\mathcal{O}$ tree cover, so we perform the proof as if $\mathsf{TC} = \{\epsilon\}$.

Using the intermediate lemma, we can use it, along with Lemma 3, to prove Theorem 2.

Proof. We use the following hybrids:

1. $\mathcal{H}_1$: Calculate $\mathsf{rescuei}\mathcal{O}(1^\lambda, C_0, 0, l, s)$
2. $\mathcal{H}_2$: Calculate $\mathsf{rescuei}\mathcal{O}(1^\lambda, C_0, i^*, l, s)$, where we increase the tree cover level from 0 to i^*. This is indistinguishable from $\mathcal{H}_1$ due to Lemma 3.
3. $\mathcal{H}_3$: Calculate $\mathsf{rescuei}\mathcal{O}(1^\lambda, C_0, i^*, l, s)$, where we switch the circuit from C_0 to C_1. This is indistinguishable from $\mathcal{H}_2$ due to Lemma 4.
4. $\mathcal{H}_4$: Calculate $\mathsf{rescuei}\mathcal{O}(1^\lambda, C_0, 0, l, s)$ where we decrease the tree cover level from i^* to 0. This is indistinguishable from $\mathcal{H}_3$ due to Lemma 3.

Example. We show $\mathsf{rescuei}\mathcal{O}$ is sufficient to obfuscate the circuits in the example in Section 3, which $\mathsf{d}\mathcal{O}$ could not handle. The two example circuits were $C_0(x) = \bigoplus_{i=1}^n x[i]$ and $C_1(x) = \bigoplus_{i=1}^n \overline{x[i]}$, for even n, are consistent for $i^* = n$. We show that the circuits are consistent by defining τ in Algorithm 17.

Algorithm 17 Transformation between example circuits.

procedure $\tau(C^*)$ ▷ C^* is a fragment of C_0
  For some $i_0 \in [n], p \in \{0,1\}$, decompose C^* into $p \oplus (\bigoplus_{i=i_0+1}^n x[i])$
  if i^* is even **then**
    return $p \oplus (\bigoplus_{i=i_0+1}^n \overline{x[i]})$
  else
    return $\overline{p} \oplus (\bigoplus_{i=i_0+1}^n \overline{x[i]})$
  end if
end procedure

To prove that τ satisfies consistency, we need to prove both properties. The second property, where fragments at level i^* are identical is trivially true, due to the fact that the two circuits are functionally equivalent, and $i^* = n$, at which point they have been evaluated. To prove the first property, we have two cases:

1. i_0 is even: $i_0 + 1$ is odd, so when we first plug in $x[i_0 + 1]$, and then apply τ, the new fragment is

$$\begin{aligned}\tau(C^*(x[i_0+1],\cdot)) &= \tau(p \oplus x[i_0+1] \oplus (\bigoplus_{i=i_0+2}^{n} x[i])) \\ &= \overline{p \oplus x[i_0+1]} \oplus (\bigoplus_{i=i_0+2}^{n} \overline{x[i]})\end{aligned}$$

   In the reverse order, the next fragment is the same:

$$\begin{aligned}\tau(C^*)(x[i_0+1],\cdot) &= \tau(p \oplus (\bigoplus_{i=i_0+1}^{n} x[i]))(x[i_0+1],\cdot) \\ &= p \oplus (\bigoplus_{i=i_0+1}^{n} \overline{x[i]})(x[i_0+1],\cdot) \\ &= p \oplus \overline{x[i_0+1]} \oplus (\bigoplus_{i=i_0+2}^{n} \overline{x[i]}) \\ &= \overline{p \oplus x[i_0+1]} \oplus (\bigoplus_{i=i_0+2}^{n} \overline{x[i]})\end{aligned}$$

2. i_0 is odd: $i_0 + 1$ is even, so when we first plug in $x[i_0 + 1]$, and then apply τ, the new fragment is

$$\begin{aligned}\tau(C^*(x[i_0+1],\cdot)) &= \tau(p \oplus x[i_0+1] \oplus (\bigoplus_{i=i_0+2}^{n} x[i])) \\ &= p \oplus x[i_0+1] \oplus (\bigoplus_{i=i_0+2}^{n} \overline{x[i]})\end{aligned}$$

   In the reverse order, the next fragment is the same:

$$\begin{aligned}\tau(C^*)(x[i_0+1],\cdot) &= \tau(p \oplus (\bigoplus_{i=i_0+1}^{n} x[i]))(x[i_0+1],\cdot) \\ &= \overline{p} \oplus (\bigoplus_{i=i_0+1}^{n} \overline{x[i]})(x[i_0+1],\cdot) \\ &= \overline{p} \oplus \overline{x[i_0+1]} \oplus (\bigoplus_{i=i_0+2}^{n} \overline{x[i]}) \\ &= p \oplus x[i_0+1] \oplus (\bigoplus_{i=i_0+2}^{n} \overline{x[i]})\end{aligned}$$

E Security of radi$\mathcal{O}$

We will show that combined-cover consistency is enough for the obfuscation of the two circuits to be indistinguishable. More precisely, for the ldO algorithm, let l_{ldO} be the tree cover size limit, and s_{ldO} be the circuit size limit, while for $\mathsf{rescueiO}$, let l be the tree cover size limit, and s be the circuit size limit. In other words:

Theorem 3 *Assuming a single-key compact* FE *scheme with polynomial selective security, for any two circuits, C_0, C_1, if there exist a polynomial size l, a tree cover* TC *and length i^*, where* TC *has at most l_{ldO} ancestors, including* TC *itself, for which C_0 and C_1 are combined-cover consistent, then:*

$$\mathsf{dO}(1^\lambda, \mathsf{rescueiO}(1^\lambda, C_0, 0, l, s), l_{\mathsf{ldO}}, s_{\mathsf{ldO}}) \overset{c}{\approx} \mathsf{dO}(1^\lambda, \mathsf{rescueiO}(1^\lambda, C_1, 0, l, s), l_{\mathsf{ldO}}, s_{\mathsf{ldO}}).$$

We will prove this theorem according to each fragment region. The second region will not show any difference between the two circuits. The third condition of combined-cover consistency suffices to show that $\forall x \in \mathsf{AfterInput}(\mathsf{TC}, i^*)$, $C_0(x, \cdot)$ and $C_1(x, \cdot)$ are identical. That is because they were both derived from identical $C_0(x_{\leq}, \cdot)$ and $C_1(x_{\leq}, \cdot)$, where $x_{\leq} \in \mathsf{MinAfterInput}(\mathsf{TC}, i^*)$.

Then the only way that a circuit, C, could have an effect on ldO $(1^\lambda, \mathsf{Assignment}(\mathsf{rescueiO}(1^\lambda, C, i^*, l, s), \mathsf{TC}), l_{\mathsf{ldO}}, s_{\mathsf{ldO}})$ is through $\mathsf{Assignment}(\mathsf{rescueiO}(1^\lambda, C, i^*, l, s), \mathsf{TC})$. At most, the obfuscated circuit depends on $(m^{(p)}_{C^*,0}, m^{(p)}_{C^*,1})$ $\forall C^* \in \mathsf{TC}$ and $p \in \{0,1\}$, and the FE decryption keys, since if we plugged in an input prefix from TC, we would have ended up with their respective ciphertexts, and $Z^{(p)}_{i,j}$ $\forall i \in [n], j \in [l], p \in \{0,1\}$, which we can collect from the FE decryption keys. For the former, we need to transform all of the FE ciphertexts that point to the corresponding data structure of their succeeding fragments, *ie.* the fragments in the third region, while for the latter, we need to remove the effect of any fragments that occur only before the tree cover for the ldO algorithm, *ie.* the fragments in the first region.

We now formally prove Theorem 3 with lemmas regarding each of these two remaining regions.

Fragments Before the ldO Tree Cover In particular, for a circuit, C, for the first region, define the following set of fragments:

We will want to remove any effect of these circuits on the final output. We do this using the following relationship between $\mathsf{Assignment}(C, \mathsf{TC})$ and $\mathsf{Assignment}(\tilde{C}_{((c_0,c_1),\{\mathsf{fsk}_i^{(b)}\}_{i\in[1,n+1],b\in\{0,1\}})}, \mathsf{TC})$ are related: only the FE ciphertexts for the fragments in $\mathsf{Assignment}(C, \mathsf{TC})$ will appear unencrypted by SKE, in $\mathsf{Assignment}(\tilde{C}_{((c_0,c_1),\{\mathsf{fsk}_i^{(b)}\}_{i\in[1,n+1],b\in\{0,1\}})}, \mathsf{TC})$. More precisely:

Lemma 5 *Let $\forall i^* \in [n]$, $\forall i_0 \in [n]$, $x_{\leq} \in \{0,1\}^{i_0}$*

$$\tilde{C}_{((c_0,c_1),\{\mathsf{fsk}_i^{(b)}\}_{i\in[1,n+1],b\in\{0,1\}})}(i_0, \cdot) = \mathsf{rescueiO}(1^\lambda, C, i^*, l, s).$$

For $i_0 \le i^*$, let $(c^{(x_{\le}[i_0])}_{C(x_{\le},\cdot),0}, c^{(x_{\le}[i_0])}_{C(x_{\le},\cdot),1})$ be the values of the same name that $\mathsf{rescuei}\mathcal{O}$ *produces. For $i_0 > i^*$, let $K_{i_0} = \mathsf{PRF.Gen}(\mathsf{PRF.Eval}_{K_{i_0-1}0}())$, and let $(c^{(x_{\le}[i_0])}_{C(x_{\le},\cdot),0}, c^{(x_{\le}[i_0])}_{C(x_{\le},\cdot),1})$ be $f^{(x_{\le}[i_0])}_{i,1}(C(x_{\le},\cdot), K_{i_0-1})$.*

Then Algorithm 18 presents the circuit of $\tilde{C}_{((c_0,c_1),\{\mathsf{fsk}_i^{(b)}\}_{i\in[1,n+1],b\in\{0,1\}})}(i_0,\cdot)$ as a function of

$$((c^{(x_{\le}[i_0])}_{C(x_{\le},\cdot),0}, c^{(x_{\le}[i_0])}_{C(x_{\le},\cdot),1}), \{\mathsf{fsk}_i^{(b)}\}_{i\in[i_0,n+1],b\in\{0,1\}}).$$

Algorithm 18 partial evaluation of $\tilde{C}_{((c_0,c_1),\{\mathsf{fsk}_i^{(b)}\}_{i\in[1,n+1],b\in\{0,1\}})}$

procedure $\tilde{C}_{((c^{(x_{\le}[i_0])}_{C(x_{\le},\cdot),0}, c^{(x_{\le}[i_0])}_{C(x_{\le},\cdot),1}),\{\mathsf{fsk}_i^{(b)}\}_{i\in[i_0,n+1],b\in\{0,1\}})}(x)$ ▷ This time, x is only a suffix of length $n-i_0$, after $x_{\le}$
  for $b \in \{0,1\}$ **do**
    $c_{i_0,b} \leftarrow c^{(x_{\le}[i_0])}_{C(x_{\le},\cdot),b}$ ▷ Initialize ciphertexts starting from i_0
  end for
  for $i \in [i_0+1, n]$ **do**
    for $w \in [|c_{i-1,0}|]$ **do**
      $\tilde{c}_i[w] \leftarrow \mathsf{Select}(x[i], c_{i-1,0}[w], c_{i-1,1}[w])$
      $\tilde{\mathsf{fsk}}_i[w] \leftarrow \mathsf{Select}(x[i], \mathsf{fsk}_i^{(0)}[w], \mathsf{fsk}_i^{(1)}[w])$
    end for
    $(c_{i,0}, c_{i,1}) \leftarrow \mathsf{FE.Dec}(\tilde{\mathsf{fsk}}_i, \tilde{c}_i)$
  end for
  return $\mathsf{FE.Dec}(\mathsf{fsk}_{n+1}^{(0)}, c_{n,0})$ ▷ Output the final calculation
end procedure

Algorithm 19 partial evaluation, $\tilde{C}_{((c_0,c_1),\{\mathsf{fsk}_i^{(b)}\}_{i\in[1,n+1],b\in\{0,1\}})}(x_{\le}[1],\cdot)$

procedure $\tilde{C}_{((c^{(x_{\le}[1])}_{C(x_{\le},\cdot),0}, c^{(x_{\le}[1])}_{C(x_{\le},\cdot),1}),\{\mathsf{fsk}_i^{(b)}\}_{i\in[1,n+1],b\in\{0,1\}})}(x)$
  for $b \in \{0,1\}$ **do**
    $c_{i,b} \leftarrow c^{(x_{\le}[1])}_{C(x_{\le}[1],\cdot),b}$
  end for
  for $i \in [2, n]$ **do**
    for $w \in [|c_{i-1,0}|]$ **do**
      $\tilde{c}_i[w] \leftarrow \mathsf{Select}(x[i], c_{i-1,0}[w], c_{i-1,1}[w])$
      $\tilde{\mathsf{fsk}}_i[w] \leftarrow \mathsf{Select}(x[i], \mathsf{fsk}_i^{(0)}[w], \mathsf{fsk}_i^{(1)}[w])$
    end for
    $(c_{i,0}, c_{i,1}) \leftarrow \mathsf{FE.Dec}(\tilde{\mathsf{fsk}}_i, \tilde{c}_i)$
  end for
  return $\mathsf{FE.Dec}(\mathsf{fsk}_{n+1}^{(0)}, c_{n,0})$ ▷ Output the final calculation
end procedure

Algorithm 20 partial evaluation, $\tilde{C}_{((c_0,c_1),\{\mathsf{fsk}_i^{(b)}\}_{i\in[1,n+1],b\in\{0,1\}})}(x_{\leq}[i_0+1],\cdot)$

procedure $\tilde{C}_{((c_{C(x_{\leq},\cdot),0}^{(x_{\leq}[i_0+1])},c_{C(x_{\leq},\cdot),1}^{(x_{\leq}[i_0+1])}),\{\mathsf{fsk}_i^{(b)}\}_{i\in[i_0+1,n+1],b\in\{0,1\}})}(x)$
  for $b \in \{0,1\}$ **do**
    $c_{i_0+1,b} \leftarrow c_{C(x_{\leq}||b^*,\cdot),b}^{(b^*)}$
  end for
  for $i \in [i_0+2, n]$ **do**
    for $w \in [|c_{i-1,0}|]$ **do**
      $\tilde{c}_i[w] \leftarrow \mathsf{Select}(x[i], c_{i-1,0}[w], c_{i-1,1}[w])$
      $\tilde{\mathsf{fsk}}_i[w] \leftarrow \mathsf{Select}(x[i], \mathsf{fsk}_i^{(0)}[w], \mathsf{fsk}_i^{(1)}[w])$
    end for
    $(c_{i,0}, c_{i,1}) \leftarrow \mathsf{FE.Dec}(\tilde{\mathsf{fsk}}_i, \tilde{c}_i)$
  end for
  return $\mathsf{FE.Dec}(\mathsf{fsk}_{n+1}^{(0)}, c_{n,0})$ ▷ Output the final calculation
end procedure

Proof. We prove this inductively:

- Base Case $i_0 = 1 \leq i^*$: For $i = 1$, since we know $x_{\leq}[1]$, we can simplify $\tilde{c}_1$ to $c_{0,x_{\leq}[1]}$, which is equal to $c_{C,0}^{(x_{\leq}[1])}$, and $\tilde{\mathsf{fsk}}_1$ to $\mathsf{fsk}_1^{(x_{\leq}[1])}$. Assuming that $1 \leq i^*$, then we know that $0 < i^*$, so we can simplify the first FE decryption by revealing the proper entry in $Z_{1,j_{C(x_{\leq}[1],\cdot)}^{(x_{\leq}[1])}}^{(x_{\leq}[1])}$:

$$\begin{aligned}(c_{0,0}, c_{0,1}) &= \mathsf{FE.Dec}(\mathsf{fsk}_1^{(x_{\leq}[1])}, c_{C,0}^{(x_{\leq}[1])}) \\ &= f_1^{(x_{\leq}[1])}(\perp, \perp, j_{C(x_{\leq}[1],\cdot)}^{(x_{\leq}[1])}, \mathsf{sk}_{j_{C(x_{\leq}[1],\cdot)}^{(x_{\leq}[1])}}^{(x_{\leq}[1])}) \\ &= \mathsf{SKE.Dec}(\mathsf{sk}_{j_{C(x_{\leq}[1],\cdot)}^{(x_{\leq}[1])}}^{(x_{\leq}[1])}, Z_{1,j_{C(x_{\leq}[1],\cdot)}^{(x_{\leq}[1])}}^{(x_{\leq}[1])}) \\ &= (c_{C(x_{\leq}[1],\cdot),0}^{(x_{\leq}[1])}, c_{C(x_{\leq}[1],\cdot),1}^{(x_{\leq}[1])})\end{aligned}$$

  So the obfuscated circuit simplifies to Algorithm 19.
- Inductive Assumption i_0: $\forall x_{\leq} \in \{0,1\}^{i_0}$, the lemma holds.
- Inductive Step $i_0 + 1$: $\forall b^* \in \{0,1\}$, we want to prove the lemma for $x_{\leq}||b^*$. Again, we can simplify the first remaining step in the largest loop, *ie.* $i = i_0 + 1$, and say that $\tilde{c}_{i_0+1} = c_{i_0,b^*}$, which, by the inductive assumption, is equal to $c_{C(x_{\leq},\cdot),b^*}^{(x_{\leq}[i_0])}$, and $\tilde{\mathsf{fsk}}_{i_0+1} = \mathsf{fsk}_{i_0}^{(b^*)}$.

If $i_0 < i^*$, then we use the first case in the decryption key function:

$$\begin{aligned}(c_{i+1,0}, c_{i+1,1}) &= \mathsf{FE.Dec}(\mathsf{fsk}_{i+1}^{(b^*)}, c_{C(x_{\leq},\cdot),b^*}^{(x_{\leq}[i_0])}) \\ &= f_{i_0+1}^{(b^*)}(\perp, \perp, j_{C(x_{\leq}||b^*,\cdot)}^{(b^*)}, \mathsf{sk}_{j_{C(x_{\leq}||b^*,\cdot)}^{(b^*)}}^{(b^*)}) \\ &= \mathsf{SKE.Dec}(\mathsf{sk}_{j_{C(x_{\leq}||b^*,\cdot)}^{(b^*)}}^{(b^*)}, Z_{i_0+1,j_{C(x_{\leq}||b^*,\cdot)}^{(b^*)}}^{(b^*)}) \\ &= (c_{C(x_{\leq}||b^*,\cdot),0}^{(b^*)}, c_{C(x_{\leq}||b^*,\cdot),1}^{(b^*)})\end{aligned}$$

On the other hand, if $i_0 \geq i^*$, then, by the inductive case, we know that $(c_{C(x_{\leq},\cdot),0}^{(x_{\leq}[i_0])}, c_{C(x_{\leq},\cdot),1}^{(x_{\leq}[i_0])})b^*$ are encryptions of $(C(x_{\leq},\cdot), K_{i_0}, \perp, \perp)$. So we use the second case:

$$\begin{aligned}(c_{i+1,0}, c_{i+1,1}) &= \mathsf{FE.Dec}(\mathsf{fsk}_{i+1}^{(b^*)}, c_{C(x_{\leq},\cdot),b^*}^{(x_{\leq}[i_0])}) \\ &= f_{i_0+1}^{(b^*)}(C(x_{\leq},\cdot), K_{i_0} \perp, \perp) \\ &= f_{i_0+1,1}^{(b^*)}(C(x_{\leq},\cdot), K_{i_0}) \\ &= (c_{C(x_{\leq}||b^*,\cdot),0}^{(b^*)}, c_{C(x_{\leq}||b^*,\cdot),1}^{(b^*)})\end{aligned}$$

By the definition of $f_{i_0+1,1}^{(b^*)}$, $(c_{C(x_{\leq}||b^*,\cdot),0}^{(b^*)}, c_{C(x_{\leq}||b^*,\cdot),1}^{(b^*)})$ are encryptions of $(C(x_{\leq}||b^*,\cdot), K_{i_0+1}, \perp, \perp)$, where $K_{i_0+1} = \mathsf{PRF.Gen}(\mathsf{PRF.Eval}_{K_{i_0}}())$.
So in both cases, the circuit simplifies to Algorithm 20.

Now that we have narrowed down the data that is present in each fragment in the circuit assignment, we can further examine what parts are present when we combine the data from all the fragments, and what we can eliminate. The $\mathsf{Id}\mathcal{O}$ algorithm will only depend on $\mathsf{Assignment}(\tilde{C}_{((c_0,c_1),\{\mathsf{fsk}_i^{(b)}\}_{i\in[1,n+1],b\in\{0,1\}})}, \mathsf{TC})$, so the output of $\mathsf{Id}\mathcal{O}$ will only depend at most on $\{(c_{C(x,\cdot),0}^{(p)}, c_{C(x,\cdot),1}^{(p)}) : p \in \{0,1\}, x \in \mathsf{TC}\}$, and $\{\mathsf{fsk}_i^{(b)}\}_{i\in[\min_{x\in\mathsf{TC}}|x|,n+1],b\in\{0,1\}}$. While the proof meant that we are able to remove all of the FE ciphertext pairs that are not SKE encrypted, corresponding to any $C^* \in \mathsf{PreTC}_{C,\mathsf{TC}}$, we have not necessarily removed them from the SKE ciphertexts, even though the circuit will never decrypt them. But we can effectively remove them due to SKE security, which we will do in hybrids, even if we pessimistically assume that the adversary has access to all of $\{\mathsf{fsk}_i^{(b)}\}_{i\in[1,n+1],b\in\{0,1\}}$.

For all $j \in [0,l,]$, $\beta \in \{0,1\}$, define the following hybrids, $\mathsf{rescuei}\mathcal{O}_{\mathsf{erase}}$, where we run $\mathsf{rscCGen}$, but with some fragments' data removed, as shown in Algorithm 21.

Then $\mathsf{rscCGen}_{\mathsf{erase}}(C, i^*, \pi_{C,0}, \pi_{C,1}, \mathsf{TC}, 0, 1)$ is identical to $\mathsf{rscCGen}(C, i^*, \pi_{C,0}, \pi_{C,1})$, while no fragment, $C^* \in \mathsf{PreTC}_{C,\mathsf{TC}}$ affects $\mathsf{rscCGen}_{\mathsf{erase}}(C, i^*, \pi_{C,0}, \pi_{C,1}, \mathsf{TC}, l, 1)$.

Lemma 6 *When calculating* $\mathsf{Assignment}(\mathsf{rescuei}\mathcal{O}(1^\lambda, C, i^*, l, s), \mathsf{TC})$ *for all* $j \in [l]$, $\beta \in \{0,1\}$, *replacing* $\mathsf{rscCGen}(C, i^*, \pi_{C,0}, \pi_{C,1})$ *with* $\mathsf{rscCGen}_{\mathsf{erase}}(C, i^*, \pi_{C,0}, \pi_{C,1}, \mathsf{TC}, j, \beta)$ *is indistinguishable from replacing it with* $\mathsf{rscCGen}_{\mathsf{erase}}(C, i^*, \pi_{C,0}, \pi_{C,1}, \mathsf{TC}, j - 1, 1)$ *if* $\beta = 0$, *or* $\mathsf{rscCGen}_{\mathsf{erase}}(C, i^*, \pi_{C,0}, \pi_{C,1}, \mathsf{TC}, j, 0)$ *if* $\beta = 1$.

Algorithm 21 rscCGen, with some fragments in $\mathsf{PreTC}_{C,\mathsf{TC}}$ erased

procedure $\mathsf{rscCGen}_{\mathsf{erase}}(C, i^*, \pi_{C,0}, \pi_{C,1}, \mathsf{TC}, j, \beta)$ ▷

TC: the $\mathsf{Id}\mathcal{O}$ tree cover, which determines if a fragment's ciphertexts should eventually be erased.
j and β: determine if an eligible fragment's ciphertexts are erased in this hybrid.

$K_{i^*} \leftarrow \mathsf{PRF.Gen}()$
for i from i^* to 0, $p \in \{0,1\}$, $C^* \in \mathsf{LevelAssignment}(C, i)$ **do**
  $j^{(p)}_{C^*} \leftarrow \pi_{C,p}(C^*)$
  for $b \in \{0,1\}$ **do**
    if $i < i^*$ **then**
      $j^{(b)}_{C^*(b,\cdot)} \leftarrow \pi_{C,b}(C^*(b,\cdot))$
      $m^{(p)}_{C^*,b} \leftarrow (\bot, \bot, j^{(b)}_{C^*(b,\cdot)}, sk^{(b)}_{i+1,j^{(b)}_{C^*(b,\cdot)}})$
    else
      $m^{(p)}_{C^*,b} \leftarrow (C^*, K_i, \bot, \bot)$
    end if
    $c^{(p)}_{C^*,b} \leftarrow \mathsf{FE.Enc}(\mathsf{mpk}^{(b)}_{i+1}, m^{(p)}_{C^*,b})$
  end for
  if $i > 0$ **then**
    if $(\gamma(C^*) < j \vee (\gamma(C^*) = j \wedge p \leq \beta)) \wedge C^* \in \mathsf{PreTC}_{C,\mathsf{TC}}$ **then**
      $\boxed{Z^{(p)}_{i,j^{(p)}_{C^*}} \leftarrow \mathsf{SKE.Enc}(\mathsf{sk}^{(p)}_{i,j^{(p)}_{C^*}}, (\bot, \bot))}$
    else
      $Z^{(p)}_{i,j^{(p)}_{C^*}} \leftarrow \mathsf{SKE.Enc}(\mathsf{sk}^{(p)}_{i,j^{(p)}_{C^*}}, (c^{(p)}_{C^*,0}, c^{(p)}_{C^*,1}))$
    end if
  end if
end for
return $(c^{(0)}_{C,0}, c^{(0)}_{C,1})$
end procedure

Algorithm 22 rscCGen for challenge to guess if a circuit's information was erased or not

procedure $\mathsf{rscCGen}^{\dagger}_{\mathsf{erase}}(C, i^*, \pi_{C,0}, \pi_{C,1}, \mathsf{TC}, j, \beta)$ j and β determine where to place the challenge SKE ciphertext.

$K_{i^*} \leftarrow \mathsf{PRF.Gen}()$

for i from i^* to 0, $p \in \{0,1\}$, $C^* \in \mathsf{LevelAssignment}(C, i)$ **do**

$j^{(p)}_{C^*} \leftarrow \pi_{C,p}(C^*)$

for $b \in \{0,1\}$ **do**

if $i < i^*$ **then**

$j^{(b)}_{C^*(b,\cdot)} \leftarrow \pi_{C,b}(C^*(b,\cdot))$

$m^{(p)}_{C^*,b} \leftarrow (\bot, \bot, j^{(b)}_{C^*(b,\cdot)}, sk^{(b)}_{i+1,j^{(b)}_{C^*(b,\cdot)}})$

else

$m^{(p)}_{C^*,b} \leftarrow (C^*, K_i, \bot, \bot)$

end if

$c^{(p)}_{C^*,b} \leftarrow \mathsf{FE.Enc}(\mathsf{mpk}^{(b)}_{i+1}, m^{(p)}_{C^*,b})$

end for

if $i > 0$ **then**

if $(\gamma(C^*) < j \vee (\gamma(C^*) = j \wedge p \leq \beta)) \wedge C^* \in \mathsf{PreTC}_{C,\mathsf{TC}}$ **then**

if $\boxed{\gamma(C^*) = j \wedge p = \beta}$ **then**

$\boxed{Z^{(p)}_{i,j^{(p)}_{C^*}} \leftarrow c^{\dagger}}$

else

$Z^{(p)}_{i,j^{(p)}_{C^*}} \leftarrow \mathsf{SKE.Enc}(\mathsf{sk}^{(p)}_{i,j^{(p)}_{C^*}}, (\bot, \bot))$

end if

else

$Z^{(p)}_{i,j^{(p)}_{C^*}} \leftarrow \mathsf{SKE.Enc}(\mathsf{sk}^{(p)}_{i,j^{(p)}_{C^*}}, (c^{(p)}_{C^*,0}, c^{(p)}_{C^*,1}))$

end if

end if

end for

return $(c^{(0)}_{C,0}, c^{(0)}_{C,1})$

end procedure

Proof. If $\nexists C^\dagger \in \mathsf{PreTC}_{C,\mathsf{TC}}$, so that $\gamma(C^\dagger) = j$, then the current hybrid is actually identical to the previous one.

Otherwise, we prove indistinguishability using sk security, since $(c^{(\beta)}_{C^\dagger,0}, c^{(\beta)}_{C^\dagger,1})$ can only be found in the SKE ciphertext, $Z^{(\beta)}_{i^*,j^{(\beta)}_{i^*,C^\dagger}}$; any other appearance would have already been used or discarded by the time any partial evaluation of the obfuscated circuit with an input in TC was reached. But first, we must prove that $\mathsf{sk}^{(p)}_{i^*,j^{(\beta)}_{i^*,C^\dagger}}$ does not appear anywhere else. $\forall C^*, b \in \{0,1\}$, if $C^*(\{0,1\},\cdot) = C^\dagger$, we have two cases:

1. $C^* \in \mathsf{PreTC}_{C,\mathsf{TC}}$: We know that $\mathsf{PreLen}(C^*) < \mathsf{PreLen}(C^\dagger)$, so $\gamma(C^*) < \gamma(C^\dagger)$, and all appearances of C^* have already been erased.
2. $C^* \notin \mathsf{PreTC}_{C,\mathsf{TC}}$: Then $\exists x \in \{0,1\}^*, x_\leq \in \mathsf{TC}$, so that $C(x,\cdot) = C^*$ and $x_\leq \sqsubseteq x$. But that means that $C(x_\leq||b,\cdot) = C^\dagger$ and $x_\leq \sqsubseteq x||b$, so $C^\dagger \notin \mathsf{PreTC}_{C,\mathsf{TC}}$, which contradicts our assumption.

Now we can show that if there is an adversary, $\mathcal{A}$, that can distinguish between the two hybrids, then we can construct an adversary, $\mathcal{A}'$, that can break SKE security:

1. Set and submit $m_0 = (c^{(\beta)}_{C^\dagger,0}, c^{(\beta)}_{C^\dagger,1})$, and $m_1 = (\bot, \bot)$.
2. Get challenge ciphertext, $c^\dagger$,
3. Run $\mathcal{A}$ with the a modified $\mathsf{rscCGen}$, as shown in Algorithm 22.
4. Return whatever $\mathcal{A}$ returns. If $c^\dagger$ was an encryption of m_0, then we have the previous hybrid, otherwise, we have the current one. So if $\mathcal{A}$ can distinguish between the two hybrids, $\mathcal{A}'$ can break SKE security.

This proof shows that we can remove the first region of fragments, and we can proceed to the third region. Let us denote the version of the main algorithm, $\mathsf{rescuei}\mathcal{O}$, in the last hybrid, by $\mathsf{rescuei}\mathcal{O}_{\mathsf{erase}}$. In other words, instead of running $\mathsf{rscCGen}(C_0, i^*, \pi_{C,0}, \pi_{C,1})$, $\mathsf{rescuei}\mathcal{O}_{\mathsf{erase}}$ runs $\mathsf{rscCGen}_{\mathsf{erase}}(C_0, i^*, \pi_{C,0}, \pi_{C,1}, \mathsf{TC}, l, 1)$.

Fragments Between the Tree Covers Now that $\mathsf{rescuei}\mathcal{O}_{\mathsf{erase}}$ no longer contains any of the fragments above TC, *ie.* the second region, we only need to handle the third region by proving the following lemma:

Lemma 7

$$\mathsf{Assignment}(\mathsf{rescuei}\mathcal{O}_{\mathsf{erase}}(1^\lambda, C_0, i^*, l, s), \mathsf{TC}) \overset{s}{\equiv} \mathsf{Assignment}(\mathsf{rescuei}\mathcal{O}_{\mathsf{erase}}(1^\lambda, C_1, i^*, l, s), \mathsf{TC})$$

Proof. We use τ to generate a new mapping for C_0, to match that of C_1, while still keeping the same probability distribution.

We already have that $\forall x \in \mathsf{InterInput}(\mathsf{TC}, i^*)$, $\tau(C_0(x,\cdot)) = C_1(x,\cdot)$, from $i_0 = \min_{x \in \mathsf{InterInput}(\mathsf{TC},i^*)} |x|$ to $i_f = \max_{x \in \mathsf{InterInput}(\mathsf{TC},i^*)} |x|$, so that we can use τ to change the mapping, $\pi_{C_0,0}$ and $\pi_{C_1,1}$.

$\forall x \in \mathsf{InterInput}(\mathsf{TC}, i^*), p \in \{0,1\}$, $(m^{(p)}_{C_0(x,\cdot),0}, m^{(p)}_{C_0(x,\cdot),1})$ and $(m^{(b)}_{C_1(x,\cdot),0}, m^{(b)}_{C_1(x,\cdot),1})$, differ by the index, $j^{(p)}_{C(x,\cdot)}$ and $j^{(p)}_{C_1(x,\cdot)}$, where they are stored. But we can simply replace $\pi_{C,p}$ with $\pi'_{C,p}$, which is generated as follows:

1. Initialize the set of chosen indexes, $Y = \emptyset$
2. Generate $\pi_{C',p}$
3. For all $C^* \in \mathsf{InterCirc}(C, \mathsf{TC}, i^*)$, Let $\pi'_{p,C}(C^*) = \pi_{p,C'}(\tau(C^*))$, and insert $\pi'_{p,C}(C^*)$ into Y.
4. For every other fragment, C^*, randomly choose an element, $j \leftarrow [l] \setminus Y$, set $\pi'_{p,C}(C^*) = j$, and put j into Y.

The last step is identical to the generation of $\pi_{C,p}$, if we first generated the outputs for all $C^* \in \mathsf{InterCirc}(C, \mathsf{TC}, i^*)$. And since τ is a bijection, and $\pi_{C',p}$ chooses outputs uniformly at random from the remaining choices, so does the third step of generating $\pi'_{C,p}$, just like generating $\pi_{C,p}$.

Putting it All Together Now we combine the previous lemmas to prove Theorem 3:

Proof. We use the following hybrids:

1. $\mathcal{H}_1$: Calculate $\mathsf{d}\mathcal{O}(1^\lambda, \mathsf{rescuei}\mathcal{O}(1^\lambda, C_0, 0, l, s), l_{\mathsf{ld}\mathcal{O}}, s_{\mathsf{ld}\mathcal{O}})$
2. $\mathcal{H}_2$: Calculate $\mathsf{d}\mathcal{O}(1^\lambda, \mathsf{rescuei}\mathcal{O}(1^\lambda, C_0, i^*, l, s), l_{\mathsf{ld}\mathcal{O}}, s_{\mathsf{ld}\mathcal{O}})$, where we moved the $\mathsf{rescuei}\mathcal{O}$ tree cover from length 0 to i^*. This is indistinguishable from $\mathcal{H}_1$ due to Lemma 3.
3. $\mathcal{H}_3$: Calculate $\mathsf{ld}\mathcal{O}(1^\lambda, \mathsf{Assignment}(\mathsf{rescuei}\mathcal{O}(1^\lambda, C_0, i^*, l, s), \mathsf{TC}), l_{\mathsf{ld}\mathcal{O}}, s_{\mathsf{ld}\mathcal{O}})$, where we moved the $\mathsf{ld}\mathcal{O}$ tree cover from $\{\epsilon\}$ to TC. This is indistinguishable from $\mathcal{H}_2$ due to Lemma 1.
4. $\mathcal{H}_4$: Calculate $\mathsf{ld}\mathcal{O}(1^\lambda, \mathsf{Assignment}(\mathsf{rescuei}\mathcal{O}_{\mathsf{erase}}(1^\lambda, C_0, i^*, l, s, \mathsf{TC}), \mathsf{TC}), l_{\mathsf{ld}\mathcal{O}}, s_{\mathsf{ld}\mathcal{O}})$, where we erased the effects of all fragments in $\mathsf{PreTC}_{C_0,\mathsf{TC}}$. This is indistinguishable from $\mathcal{H}_3$ due to Lemma 6.
5. $\mathcal{H}_5$: $\mathsf{ld}\mathcal{O}(1^\lambda, \mathsf{Assignment}(\mathsf{rescuei}\mathcal{O}_{\mathsf{erase}}(1^\lambda, C_1, i^*, l, s, \mathsf{TC}), \mathsf{TC}), l_{\mathsf{ld}\mathcal{O}}, s_{\mathsf{ld}\mathcal{O}})$, where we switched from C_0 to C_1. This is indistinguishable from $\mathcal{H}_4$ due to Lemma 7.
6. $\mathcal{H}_6$: From now on, the hybrids apply to C_1, and the arguments are symmetric to the previous hybrids.
   Calculate $\mathsf{ld}\mathcal{O}(1^\lambda, \mathsf{Assignment}(\mathsf{rescuei}\mathcal{O}(1^\lambda, C_1, i^*, l, s), \mathsf{TC}), l_{\mathsf{ld}\mathcal{O}}, s_{\mathsf{ld}\mathcal{O}})$, where we added back the effects of all fragments in $\mathsf{PreTC}_{C_1,\mathsf{TC}}$. This is indistinguishable from $\mathcal{H}_5$ due to Lemma 6.
7. $\mathcal{H}_7$: Calculate $\mathsf{d}\mathcal{O}(1^\lambda, \mathsf{rescuei}\mathcal{O}(1^\lambda, C_1, i^*, l, s), l_{\mathsf{ld}\mathcal{O}}, s_{\mathsf{ld}\mathcal{O}})$, where we moved the $\mathsf{ld}\mathcal{O}$ tree cover from TC to $\{\epsilon\}$. This is indistinguishable from $\mathcal{H}_6$ due to Lemma 1.
8. $\mathcal{H}_8$: Calculate $\mathsf{d}\mathcal{O}(1^\lambda, \mathsf{rescuei}\mathcal{O}(1^\lambda, C_1, 0, l, s), l_{\mathsf{ld}\mathcal{O}}, s_{\mathsf{ld}\mathcal{O}})$, where we moved the $\mathsf{rescuei}\mathcal{O}$ tree cover from length i^* to 0. This is indistinguishable from $\mathcal{H}_7$ due to Lemma 3.

F Example Combined-Cover Consistent Circuits

Theorem 3, allows us to obfuscate not only the circuits that $\mathsf{d}\mathcal{O}$ can obfuscate, but $\mathsf{rescuei}\mathcal{O}$ cannot, but also circuits beyond the $\mathsf{d}\mathcal{O}$ obfuscation class. Next we

give a example of such circuits. For some odd $n \geq 5$, define the following circuit:

$$C_0(x) = x[1] \oplus x[2] \oplus x[3] \oplus \bigoplus_{i=x+3}^{n} x[i]$$

$$C_1(x) = \mathsf{Select}(x[1], \mathsf{Select}(x[2], x[3], \overline{x[3]}), \mathsf{Select}(x[2], \overline{\overline{x[3]}}, \overline{\overline{x[3]}})) \oplus \bigoplus_{i=x+3}^{n} x[i]$$

These circuits contain parts, that, as we have shown, make them unobfuscatable by both $\mathsf{d}\mathcal{O}$ and $\mathsf{rescuei}\mathcal{O}$, if we only used them alone.

But they are combined-cover consistent for $\mathsf{TC} = \{0,1\}^3$, and $i^* = n$. Since i^* is odd, both $\bigoplus_{i=x+1} x[i]$ and $\bigoplus_{i=x+1} x[i]$ are XORing an even number of bits, so they are equivalent. Since the first part is also equivalent, the entirety of both circuits are equivalent. Again, since we set $i^* = n$, the last property of combined-cover consistency follows. We now define τ in Algorithm 23. Since the first three bits have already been used by the time we reach TC, the remaining fragments all contain only $\bigoplus_{i=x+1} x[i]$ or $\bigoplus_{i=x+1} x[i]$, so the algorithm can decompose the fragment as an XOR of several variable bits, and a constant.

Algorithm 23 Transformation between example circuits.

procedure $\tau(C^*)$ ▷ C^* is a fragment of C_0
  For some $i_0 \in [3, n], p \in \{0,1\}$, decompose C^* into $p \oplus (\bigoplus_{i=i_0+1}^{n} x[i])$
  if i^* is odd **then**
    return $p \oplus (\bigoplus_{i=i_0+1}^{n} \overline{x[i]})$
  else
    return $\overline{p} \oplus (\bigoplus_{i=i_0+1}^{n} \overline{x[i]})$
  end if
end procedure

Now we need to prove the first property, for $x \in \{0,1\}^3$. $i_0 = 3$, so it is odd, which means that $\forall x \in \mathsf{TC}$ (note that the first three XORs have already been evaluated):

$$\begin{aligned}\tau(C_0(x, \cdot)) &= \tau((x[1] \oplus x[2] \oplus x[3]) \oplus \bigoplus_{i=x+1} x[i]) \\ &= (x[1] \oplus x[2] \oplus x[3]) \oplus \bigoplus_{i=x+1} \overline{x[i]} = C_1(x, \cdot)\end{aligned}$$

Now we prove the second property, similarly to how we proved the first property in plain consistency, with two cases:

1. i_0 is odd: $i_0 + 1$ is even, so when we first plug in $x[i_0 + 1]$, and then apply τ, the new fragment is

$$\begin{aligned}\tau(C^*(x[i_0+1],\cdot)) &= \tau(p \oplus x[i_0+1] \oplus (\bigoplus_{i=i_0+2}^{n} x[i])) \\ &= \overline{p \oplus x[i_0+1]} \oplus (\bigoplus_{i=i_0+2}^{n} \overline{x[i]})\end{aligned}$$

   In the reverse order, the next fragment is the same:

$$\begin{aligned}\tau(C^*)(x[i_0+1],\cdot) &= \tau(p \oplus (\bigoplus_{i=i_0+1}^{n} x[i]))(x[i_0+1],\cdot) \\ &= p \oplus (\bigoplus_{i=i_0+1}^{n} \overline{x[i]})(x[i_0+1],\cdot) \\ &= p \oplus \overline{x[i_0+1]} \oplus (\bigoplus_{i=i_0+2}^{n} \overline{x[i]}) \\ &= \overline{p \oplus x[i_0+1]} \oplus (\bigoplus_{i=i_0+2}^{n} \overline{x[i]})\end{aligned}$$

2. i_0 is even: $i_0 + 1$ is odd, so when we first plug in $x[i_0 + 1]$, and then apply τ, the new fragment is

$$\begin{aligned}\tau(C^*(x[i_0+1],\cdot)) &= \tau(p \oplus x[i_0+1] \oplus (\bigoplus_{i=i_0+2}^{n} x[i])) \\ &= p \oplus x[i_0+1] \oplus (\bigoplus_{i=i_0+2}^{n} \overline{x[i]})\end{aligned}$$

   In the reverse order, the next fragment is the same:

$$\begin{aligned}\tau(C^*)(x[i_0+1],\cdot) &= \tau(p \oplus (\bigoplus_{i=i_0+1}^{n} x[i]))(x[i_0+1],\cdot) \\ &= \overline{p} \oplus (\bigoplus_{i=i_0+1}^{n} \overline{x[i]})(x[i_0+1],\cdot) \\ &= \overline{p} \oplus \overline{x[i_0+1]} \oplus (\bigoplus_{i=i_0+2}^{n} \overline{x[i]}) \\ &= p \oplus x[i_0+1] \oplus (\bigoplus_{i=i_0+2}^{n} \overline{x[i]})\end{aligned}$$