Structure-Preserving Compilers from New Notions of Obfuscations

Matteo Campanelli[1], Danilo Francati[2](✉), and Claudio Orlandi[2]

[1] Protocol Labs, San Francisco, USA
matteo@protocol.ai

[2] Aarhus University, Aarhus, Denmark
{dfrancati,orlandi}@cs.au.dk

Abstract. The dream of software obfuscation is to take programs, *as they are*, and then generically compile them into obfuscated versions that hide their secret inner workings. In this work we investigate notions of obfuscations weaker than virtual black-box (VBB) but which still allow obfuscating cryptographic primitives preserving their original functionalities as much as possible.

In particular we propose two new notions of obfuscations, which we call *oracle-differing-input* obfuscation (odiO) and *oracle-indistinguishability* obfuscation (oiO). In a nutshell, odiO is a natural strengthening of *differing-input* obfuscation (diO) and allows obfuscating programs for which it is hard to find a *differing-input* when given only *oracle access* to the programs. An oiO obfuscator allows to obfuscate programs that are *hard to distinguish* when treated as oracles.

We then show applications of these notions, as well as positive and negative results around them. A few highlights include:
- Our new notions are weaker than VBB and stronger than diO.
- As it is the case for VBB, we show that there exist programs that cannot be obfuscated with odiO or oiO.
- Our new notions allow to generically compile several flavours of secret-key primitives (e.g., SKE, MAC, designated verifier NIZK) into their public-key equivalent (e.g., PKE, signatures, publicly verifiable NIZK) while preserving one of the algorithms of the original scheme (function-preserving), or the structure of their outputs (format-preserving).

1 Introduction

Obfuscation and Its (Dream) Applications. Obfuscation—the ability of running a program hiding its inner working—is a cryptographer's dream. This is especially true of its most powerful instantiation, virtual black-box (VBB) obfuscation: anything a VBB-obfuscated program leaks can be simulated through oracle access to the function it computes [8]. It follows that one important application of VBB is to *generically transform* secret-key cryptographic primitives into their public-key counterparts (an approach sometimes referred to as *white-box cryptography*). For example, the seminal work of Diffie and Hellman [25]

A. Boldyreva and V. Kolesnikov (Eds.): PKC 2023, LNCS 13941, pp. 663–693, 2023.
https://doi.org/10.1007/978-3-031-31371-4_23

already imagined compiling secret key encryption (SKE) into public key encryption (PKE) by letting the public key consist of the obfuscated encryption program $\mathsf{Enc}(\mathsf{k}, \cdot)$. Note that this compiler has the advantage of preserving the *format* of the underlying ciphertext, as well as the *function* used to perform decryption.

Transforming Primitives, Nicely. In this paper, we are interested in obfuscators that allow generic *structure preserving* transformation of large classes of cryptographic primitives, i.e., obfuscators that allow to compile cryptographic primitives while *preserving* parts of the original primitive. In particular, with the term structure-preserving, we refer to two main classes of transformations (from secret-key to public-key primitives), dubbed *function-preserving* and *format-preserving* transformations:

- *Function-preserving.* This first type of transformation does not alter the algorithms (one of which is then obfuscated during the transformation) of the secret-key primitive. An example of such a transformation is the one described by Diffie and Hellman, i.e., compile a SKE into a PKE by obfuscating (without any modification) the encryption algorithm and keep the decryption one unchanged.
- *Format-preserving.* This other type of transformation modifies the algorithms of the original secret-key primitive but it preserves the format of the output.[1] For example, in order to convert a SKE into a PKE, a format-preserving transformation may require to modify both (before obfuscating) the encryption and decryption algorithm. However, these modifications do not alter the format of ciphertexts (i.e., the ciphertexts of the resulting PKE is of the same format as the original SKE one).

We see this as an interesting design approach to transformation of primitives, worth of study of its own. Structure-preserving compilers are desirable because of: *(i) reusability/retrocompatibility* and *(ii) efficiency.* First, with function-preserving transformations we can reuse existing code, programs, libraries, constructions and their cryptanalysis. Cryptographic primitives deployed in hardware could reuse that same hardware for the transformed primitive, instead of having to be redesigned from scratch and possibly replaced in a production environment. Moreover, transformations that preserve the *format* of their output allow to reuse parsing-related software and to be retrocompatible with older standards (particularly important for legacy systems). Also, function- and format-preserving transformations maintain some of the scheme's original efficiency guarantees such as preserving the running time of the (possibly heavily optimized) original function and its communication complexity, respectively.

Nice Transformations from *Weaker* Obfuscation? The seminal work in [7,8] has shown that the "dream version" of obfuscation, VBB, is in gen-

[1] Note that a function-preserving transformation is also format-preserving. This is because the former does not modify the algorithms of the original primitive. Hence, the format of the output is preserved by definition.

eral impossible, i.e., there exist programs that cannot be obfuscated through VBB. Since then cryptographers have defined new, weaker notions of obfuscations that could hopefully be constructed. One of the plausible weaker candidates in this sense is *indistinguishability obfuscation* (iO) that guarantees the indistinguishability of a pair obfuscated programs, only if the latter have the exact same input-output behavior. It is truly surprising that a notion of obfuscation as weak as iO has managed to generate so many applications [41]. However, most of the applications of iO are out of the spectrum of the "design once; obfuscate later"-approach that was dreamed in the beginning, i.e., generically compile an existing secret-key primitive that it has been designed without the intend of being obfuscated later in time. In fact, most iO-based constructions are quite involved and they are not generic since only carefully designed programs can be successfully obfuscated through iO. A clear example is [41] that leverages puncturable PRFs and pseudorandom generators (PRG) to build (from scratch) a SKE scheme that satisfies very specific properties (e.g., puncturability) which, in turn, allows iO to convert it into a PKE scheme. Intuitively, this is far from having a generic transformation since the SKE is built with the intent of being obfuscated through iO. It is therefore natural to ask the following question:

Can we obtain generic structure-preserving transformations from notions of obfuscation weaker than VBB*?*

Our Results: New Primitives, Compilers, Connections to Prior Notions. In this work we propose two new definitions of obfuscation, *oracle-differing-input* obfuscation (odiO) and *oracle-indistinguishability* obfuscation (oiO), and apply them to structure-preserving transformations for several classes of primitives.

Recall that iO [8] only guarantees indistinguishability of obfuscations between pair of programs that have the exact same input/output behaviour. *Differing-input obfuscation* (diO) [1,8,17] is a stronger kind of obfuscation which guarantees the same indistinguishability property of iO but for pair of programs which might have different input/output behaviour, as long as it is computationally hard to find inputs on which the output of the programs differ, even when looking at the code of the programs. Our first notion, odiO, enriches the class of programs that can be securely obfuscated including any pair of programs for which it is hard to find differing-inputs, but when the distinguisher is given only oracle access to the programs. oiO then takes it a step further and allows to obfuscate any pair of programs that are indistinguishable when given as oracle.

In the paper we formally study the relationship between our new notions of obfuscation and the existing one. Note that:

$$\mathsf{VBB} > \mathsf{oiO} > \mathsf{odiO} > \mathsf{diO} > \mathsf{iO}$$

meaning that a VBB-obfuscator is also an oiO-obfuscator, and so on. Intuitively, the separation are strict. Again, focusing only on the first inequality: while a VBB-obfuscator cannot leak anything about the program that cannot be learned

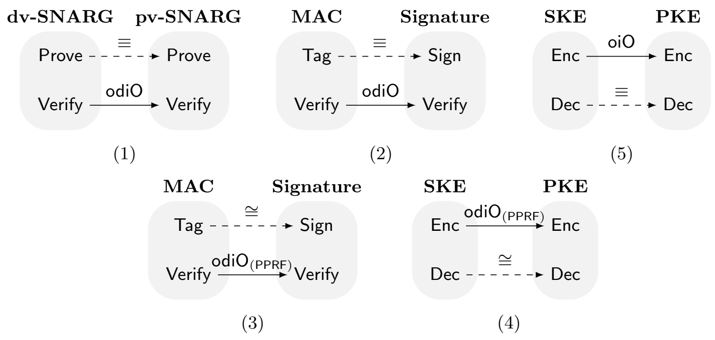

Fig. 1. The transformations (1)-(5) of this work: function-preserving on top row; format-preserving on bottom row. By odiO/oiO we denote an algorithm obtained through direct obfuscation of the one on the left; by $\equiv$ one that is completely unchanged; by $\cong$ one with minor changes but still able to take the same input; by (PPRF) we denote where we modify the algorithm through puncturable PRFs before obfuscation.

by the oracle version of the program, an oiO-obfuscator is allowed to leak any secret contained in its circuit, as long as these secrets do not allow to distinguish between the oracle programs. Focusing on oiO, odiO, diO, and iO, we have that all these notions provide the same flavor of security (i.e., two obfuscations are indistinguishable) but for different classes of circuits, each progressively contained into the other. For this reason, we have that $\mathsf{oiO} > \mathsf{odiO} > \mathsf{diO} > \mathsf{iO}$.

Note that odiO is stronger than diO. Hence, as considered by previous works for diO, this work assumes that current candidates of iO obfuscators are candidates obfuscators for odiO and oiO.

Still, since oiO and odiO are weaker than VBB, it is plausibly easier to build oiO and odiO obfuscators than VBB ones, at least for specific classes of programs: For example, it is known that point functions can be VBB-obfuscated, despite the general impossibility results for VBB; similarly, programs that differ in a single input (or polynomial number of inputs) can be diO-obfuscated, even if we believe that diO is unlikely to exist in general. In the same spirit, our results can be interpreted as showing that we can lift certain symmetric-key primitives to public-key primitives as long as specific functions (e.g., verification algorithms) can be odiO-obfuscated.

We then show that our new notions of obfuscation are enough for generic structure-preserving transformations of important cryptographic primitives. In particular we provide the following transformation (see also Fig. 1):

1. A *function-preserving* transformation from selectively sound succinct designated verifier non-interactive argument systems (dv-SNARG) into publicly verifiable ones (pv-SNARG) (Sect. 5.1); The same transformation allows transforming non-interactive argument systems that satisfy straight-line knowledge

soundness, i.e., it is possible to extract (through a trapdoor) a valid witness from verifying proofs without interacting with the adversary;[2]
2. A *function-preserving* transformation from strong existentially unforgeable MACs into digital signatures that remains strongly unforgeable only in the presence of adversaries that can ask signatures of arbitrary messages in a selective fashion;
3. A *format-preserving* transformation that leverages puncturable PRFs to convert selectively existentially unforgeable MACs into selectively existentially unforgeable digital signatures. In constrast to the previous (MACs to signatures) transformation, this is only format-preserving but achieves existential unforgeability under the standard notion of chosen message attacks (i.e., the adversary has adaptive oracle access to the signature algorithm);
4. A *format-preserving* transformation that leverages puncturable PRFs to convert IV-based selectively secure SKEs into selectively IND-CPA secure PKEs. Here, IV-based SKEs refer to encryption schemes of the form $\mathsf{Enc}(\mathsf{k}, m; \mathsf{iv}) = (\mathsf{iv}, c)$ where iv is the initialization vector (i.e., randomness) used to encrypt a message m. Note that most SKE used in practice are IV-based e.g., those based on block ciphers mode operations such as AES-CBC-mode, AES-CTR-mode, and so on.
5. A *function-preserving* transformation from any semantically secure and key indistinguishable SKE into a selective IND-CPA PKE (Sect. 5.2). Here, the SKE's key indistinguishability property must hold under chosen message randomness attacks, i.e., it is infeasible to determine under which key a target message has been encrypted even if the adversary has oracle access to $\mathsf{Enc}(\mathsf{k}, \cdot; \cdot)$ that accepts adversarially chosen messages and randomnesses.

Note that only the last transformation requires oiO (in order to use the key indistinguishability property of the SKE) whereas odiO is sufficient to achieve the other ones. Also, all the transformations that use puncturable PRFs are (only) *format-preserving*, i.e., the programs/algorithms of the compiled primitive are (slightly) modified but the format of the output is preserved.[3] We anticipate that all our transformations require $\mathsf{odiO}/\mathsf{oiO}$ and they cannot be implemented using iO (or diO). In a nutshell, this is because we focus on generic transformations from secret-key to public-key primitives. In order to be generic, we need to eventually reduce the security of the transformation to the security of the original secret-key primitive. If we wish to accomplish this reduction using either iO or diO, we need to put, into the obfuscated circuit, the key sk of the original (secret-key)

[2] As for straight-line knowledge soundness, we do not consider succinctness (i.e., we do not cover $\mathsf{dv\text{-}SNARG}/\mathsf{pv\text{-}SNARG}$) since, in order to have a straight-line extraction, the size of the proof is proportional to the size of the witness.

[3] We will elaborate on this later, but intuitively this is because the obfuscated program will use the puncturable PRF to generate a fresh symmetric key for different input (e.g., messages, initialization vectors). Hence, on decryption/verification, the receiver needs to evaluate the same PRF in order to recompute the symmetric key used to decrypt/verify a particular ciphertext/signature.

primitive. However, this is not possible. During the reduction sk is sampled and kept secret by the challenger. We provide more details in Sect. 1.1.

Although both odiO and oiO are weaker than VBB, this does not tell us anything about the plausibility of these new notions of obfuscation (and their applications). As a last contribution, we investigate whether VBB's impossibility results of Barak et al. [7,8] extends to either odiO or oiO (or both). In particular, Barak et al. [7,8] shows the following two impossibility results regarding VBB. *(i)* The first states an *universal* VBB obfuscator does not exist, i.e. there exists (not necessarily natural) computations that cannot be obfuscated through VBB. *(ii)* The second states that even the specific applications/transformations of VBB we can naturally hope for are impossible (e.g., converting any SKE into a PKE).

What about the above and $\mathsf{odiO}/\mathsf{oiO}$? We answer this question as follows:

- Result *(i)* does apply to odiO and oiO. This does not say much about how useful they are, so in the paper we explore result (ii) as well (see Sect. 6.1).
- Result *(ii)* applies only to transformation (5) for oiO. Moreover, we need to rework the original result *(ii)* from [7,8] to extend it to transformation (5) since it does not apply as it is (see Sect. 6.2).
- Result *(ii)* does not apply at all to the applications/transformations we have for odiO and there seems no natural way to extend it to them. Hence, all our odiO-based transformations remain plausible.

Summing up, while the first result *(i)* applies to both our proposed notions, odiO is *not* at all subject to the second result *(ii)* (impossibility of applications), which is the most limiting one.

Expanding more about the above results, we provide two different negative results by adapting the techniques of [7,8] to the case of odiO and oiO. First, we show that there exists an ensemble of circuits that neither odiO nor oiO cannot obfuscate, unconditionally (Sect. 6.1). Second, we show that the oiO-based function-preserving transformation (5) from any semantically secure and key indistinguishable SKEs into selective IND-CPA secure PKEs is inherently impossible (no matter what type of obfuscator is used to implement it).[4] We elaborate further on this in the technical overview (Sect. 1.1) and in the related Sect. 6.

Why Study these New Notions if they are Still Subject to the [7,8] Impossibility Results? There are multiple responses to that (some of which we expand below):

1. The work of Barak et al. [7,8] does not really say much about how useful $\mathsf{odiO}/\mathsf{oiO}$ are (see also technical overview). Indeed, [7,8] shows *two* types of impossibility results, i.e., impossibility of an universal obfuscator and impossibility of applications. As we mentioned above, these impossibilities do not

[4] Note that Barak et al. [8] demonstrates the impossibility of transforming a SKE into a PKE (through the obfuscation of its encryption algorithm $\mathsf{Enc}(\mathsf{k}, \cdot)$) by building a (contrived) secure SKE that, after applying the transformation, yields an insecure PKE. However, their contrived SKE is not key indistinguishable. For this reason, in order to prove the impossibility of our oiO-based transformation (5) (from key indistinguishable SKE to PKE) we need to rework their result.

equally extend to both notions, e.g., impossibility of applications do not extend to our odiO-based transformations.

2. It is still important to study foundational aspects of obfuscation. We think ours are natural questions and natural notions to propose and to turn our attention to. As it is often the case in theoretical research, these notions may be connected to others in the future in unexpected ways. They might also motivate further work on notions that will turn out to be achievable (the [7,8] prompted the quest for iO and other notions related to VBB).
3. Related to the above, both odiO and oiO might be useful for many "proof-of-concept" type of results that rely on VBB-obfuscation. In cases where these weaker definitions might suffice, the proposed notions could shed light on what security property is actually needed from the obfuscator to imply security of the overall construction.[5] As an example, our results can be interpreted as follows: If a particular circuit (e.g., secret-key verification/encryption algorithm) can be odiO-obfuscated (resp. oiO-obfuscated) then we can lift a particular secret-key primitive into its public-key flavor (e.g., MAC to signatures, SKE to PKE).
4. VBB and odiO/oiO are still distinct notions and with distinct flavors of security (simulation- vs indistinguishability-based). Moreover, nonetheless the known impossibility results, research on VBB is still active such as identifying specific and interesting class of circuits that can be securely VBB-obfuscated [33,43,45]. The same can be investigated for the case of odiO/oiO. For example, there could exist a specific class of circuits that can be oiO-/odiO-obfuscated but not VBB-obfuscated. Or there could exist circuit classes that are VBB-obfuscatable, but that can still be odiO/oiO-obfuscated more efficiently or from significantly weaker assumptions.

Lastly, we stress that odiO/oiO may have other interesting applications. This work focuses on secret-key to public-key transformations since these are most prominent applications of VBB and, we believe that studying odiO/oiO in the same context provides a better understanding about the relations between odiO/oiO and VBB, including their limitations.

1.1 Technical Overview

Oracle-Differing-Input Obfuscation (odiO). The notion of odiO is a variant of the notion of differing-input obfuscation, or diO. What is common with diO, for example, is that: *(i)* we are given a sampler S that outputs two circuits C_0 and C_1 and some auxiliary information α; *(ii)* the output of the sampler should satisfy some property P (we call such sampler "permissible"); *(iii)* if the sampler sastisfies property P then the obfuscated circuits $\mathsf{Obf}(C_0)$ and $\mathsf{Obf}(C_1)$ should look indistinguishable to a PPT adversary given also in input α. Also, in both diO and odiO, the property P corresponds to "no PPT D can find a *differing*

[5] This follows the same spirit of the UCE framework proposed by Bellare et al. [9] that allows to identify which property of the random oracle model (ROM) is needed to imply security of the (ROM-based) construction..

input x for C_0 and C_1 (given in input α)", that is an input x such that $C_0(x) \neq C_1(x)$. Where the two definitions diverge is that in diO algorithm D takes as input the *actual representation* (the code) of the two circuits, whereas in odiO D only has *oracle access* to the functions computed by C_0 and C_1.

An example of sampler that is permissible for odiO but not diO is the following: consider two programs C_0 and C_1 where their only (high-entropy) differing input is encoded as a comment in their code. Given their code it is easy to find such input, but not with oracle access to them. We provide more examples when we discuss our transformations below.

Public-key "Forgery-based" Transformations through odiO. We show that odiO is particularly suitable for transforming a general class of primitives—which we informally dub *forgery-based*—from their secret-key to their public-key version. By forgery-based we mean a primitive where the security is defined roughly as follows: "No adversary can produce (forge) a string passing a given test without knowledge of a certain secret (or if a certain condition does not hold)". Straightforward examples of this type of primitives include message-authentication codes (MACs) and digital signature, but non-interactive proof systems and signatures of knowledge [24] also capture this intuition.

The properties of odiO are sufficient for compiling the forgery-based primitives (1)-(3) listed above. We now give the main intuitions behind our transformations and their security. Our goal is to transform a primitive allowing us to verify a string through knowledge of secret into one that can do the same without such knowledge. Let us denote the first generic verification algorithm by $\mathsf{Verify}(\mathsf{sk}, \dots)$;[6] we aim to transform it into a public key equivalent $\mathsf{Verify}'(\mathsf{pk}, \dots)$. Our construction is straightforward: We define pk as the odiO-obfuscation of $\mathsf{Verify}(\mathsf{sk}, \dots)$, and the program $\mathsf{Verify}'(\mathsf{pk}, \dots)$ simply runs the program encoded in pk.

We now argue that the above is secure in a selective-security-flavored setting. In general, in such a setting, the adversary first claims some input (e.g., a message or an NP statement) for which it would like to forge a valid string (e.g., a signature or a proof). The rest of the intuition is better conveyed being specific. We thus focus on the setting of non-adaptive (selective) security in non-interactive proof systems where the verifier has the syntax $\mathsf{Verify}(\mathsf{vrs}, x, \pi)$ and vrs is the (secret) verification key, x is a public statement (allegedly in a language $\mathcal{L}$), π is the proof. In this security game, for any input $\hat{x} \notin \mathcal{L}$, the adversary should not be able to forge a corresponding valid proof *after* seeing the public parameters (aka, common reference string or crs). We now show how to reduce the security of the publicly verifiable construction to that of the original (designated verifier) one applying odiO security. Recall that the security property of odiO must refer to a given sampler returning pairs of circuits. We require that our odiO obfuscator is secure against a sampler that returns (C_0, C_1) (we ignore the auxiliary input here) where:

[6] The rest of the input besides the key is irrelevant for this discussion.

- C_0 takes as input x and π and returns $\mathsf{Verify}(\mathsf{vrs}, x, \pi)$.
- C_1 behaves like C_0 *except* that it immediately returns 0 whenever $x = \hat{x}$.

The two circuits clearly satisfy the odiO permissibility notion since finding a differing input through oracle access to them would violate the original hypothesis of soundness (the only differing inputs are valid proofs for $\hat{x}$).[7]

Thus we can move to an hybrid where the crs is an obfuscation of C_1, and indistinguishability of the hybrids follows from the security of the odiO obfuscator. But now note that by construction of C_1, when $\mathsf{crs} = \mathsf{Obf}(C_1)$, an adversary by definition cannot produce a valid for $\hat{x}$. Moreover, we obtain (for free) that our transformation preserves zero-knowledge since it is function-preserving and the Prove algorithm is not modified (see Remark 5.3).

The blueprint for the construction and security proof above can be adapted (with the appropriate care) to the other forgery-settings (2)-(3) for which we propose transformations. For transformation (2)—which yields selectively-secure strongly unforgeable signatures—one technical challenge is that we need to simulate the queries to the signing oracle. Since these queries are selective we can embed them in one of the circuits we obfuscate during the hybrid arguments. Transformation (3) requires additional care since it yields a signature scheme secure against an adversary with *adaptive* queries to the signing oracle. To do so we slightly modify the signature algorithm and use a (puncturable) PRF to generate a fresh one-time symmetric-key used to sign a single message. The verification algorithm is similarly adapted and then obfuscated. Due to the use of the PRF, the transformation is not function-preserving but only format-preserving.

Compiling Extractable Argument Systems. We are able to extend our result for argument schemes satisfying *soundness* to arguments that satisfy *knowledge soundness*. This is achieved by the exact same function-preserving construction from odiO.[8] We are able to compile an adaptively-secure straight-line extractable designated verifier argument into an adaptively-secure straight-line extractable publicly verifiable argument. Note that, when considering straight-line extractability, proofs are not succinct anymore; hence, in this case we cover dv-NIZK and pv-NIZK. In contrast to soundness—which achieves only selective security—here we are able to preserve adaptive security. Again, the

[7] In particular, soundness (of underlying designated-verifier non-interactive proof system) must hold even if the adversary has oracle access to the verification algorithm. The latter is essential during the reduction to simulate the input-output behavior of the two circuits (treated as oracles). Hence, our transformation does not apply to non-interactive proofs systems that suffer from the so called verifier rejection problem, i.e., giving oracle access to the verifier allows the adversary to break soundness..

[8] Despite the construction is the same, the sampler required to prove knowledge soundness is different.

transformation is function-preserving and it does not alter the Prove algorithm. Hence, zero-knowledge is preserved (see also Remark 5.3). To the best of our knowledge ours is the first work applying obfuscation in the context of extractability in proof schemes.

Using odiO for Public-key Encryption through Puncturable PRFs. So far we discussed how odiO is particularly useful for forgery-flavored primitives. We observe, however, that we are able to prove security of another type of primitive, encryption. In the full version of this work [21], we show how to compile IV-based selectively secure SKEs (whose ciphertexts have the form $\mathsf{Enc}(\mathsf{k}, m; \mathsf{iv}) = (\mathsf{iv}, c)$) into selectively IND-CPA secure PKEs. Our obfuscated circuit (that will be our pk) uses two puncturable PRFs: The first to generate the initialization vector iv from the randomness given to the PKE's Enc and, the second to generate a one-time fresh symmetric-key (used to encrypt) from iv.[9] The decryption algorithm has access to the key for the second PRF and takes as input the ciphertext (iv, c). It can then regenerate the key and thus decrypt. Note that this transformation is only format-preserving since we slightly modify both encryption and decryption algorithm to embed the evaluation of the PRF.

Oracle-Indistinguishability Obfuscation (oiO). The notion of oiO represents a natural strengthening of odiO. It has similar features to diO and odiO in that it requires samplers that output pairs of circuits satisfying some permissibility predicate P. While the permissibility predicate in diO and odiO requires hardness of finding a differing-input, in oiO we have a weaker permissibility predicate (which in turn makes oiO stronger than odiO): in oiO the sampler must output pairs of circuits such that an adversary (given also as input related auxiliary string α) cannot distinguish the circuits while having only oracle oracle access to them. An example of a sampler that is permissible for oiO but not odiO is the one where C_0 and C_1 are both PRFs but with different keys, since they differ on (almost) every input but their output distributions are indistinguishable.

Public-key "Indistinguishability-based" Transformations through oiO. While odiO is suitable for transforming forgery-based primitives, oiO has synergies with indistinguishability-based primitives, i.e. where "No adversary can distinguish between two distributions without knowledge of a certain secret". Natural examples are encryption schemes where the distributions to distinguish are the encryption of different messages (e.g., IND-CPA security).

[9] If, instead of generating iv using the first PRF, we allow the circuit to take directly in input iv then the PKE (output by the transformation) is trivially broken. This is because (following the syntax of the IV-based SKE) iv is included into the ciphertext. Hence, an adversary can break the selective IND-CPA security of the compiled PKE by simply re-encrypting a message using the iv that is included into the challenge ciphertext.

Through oiO we are able to prove the security of a more general transformation (compared to (4)) from SKEs to PKEs. Starting from a symmetric encryption algorithm $\mathsf{Enc}(\mathsf{k}, \cdot; \cdot)$, our aim is to transform it into something with the following syntax $\mathsf{Enc}(\mathsf{pk}, \cdot; \cdot)$, where pk is a public key. Our transformation is identical to the one proposed by Diffie and Hellman [25]: We define pk as the oiO-obfuscation of $\mathsf{Enc}(\mathsf{k}, \cdot; \cdot)$ for some honestly chosen symmetric key k. To claim the IND-CPA security of the above transformation, we need to assume that the initial SKE is key indistinguishable under (adversarially) chosen message randomness attacks. The latter allows us to build a sampler that satisfies the permissibility predicate of oiO. In particular, the sampler returns (C_0, C_1) (again, we ignore the auxiliary input here) where:

- C_0 takes as input m and r and returns $\mathsf{Enc}(\mathsf{k}, m; r)$.
- C_1 is identical to the above except that it uses a different (honestly generated) symmetric key k'.

Intuitively, the circuits satisfy the oiO permissibility notion since any adversary that is able to distinguish between oracles C_0 and C_1 would also violates the key indistinguishability security of the SKE. Now, since the obfuscations of these two circuits are indistinguishable, we can reduce the security of the PKE to the security of the original SKE. Consider the standard IND-CPA experiment of PKE where pk is set to the obfuscation of C_0 and the challenge ciphertext c is computed as $c = \mathsf{pk}(m_b; r) = \mathsf{Enc}(\mathsf{k}, m_b; r)$ for r randomly chosen. We can now do an hybrid where pk is set to the obfuscation of C_1 whereas the challenge ciphertext is still computed as $c = \mathsf{Enc}(\mathsf{k}, m_b; r)$ where k is the key hardcoded in C_0. Since the ciphertext c is computed using a key k that is not the obfuscated one (recall C_1 uses an independent key k'), we can now conclude the proof by doing a reduction to the semantic security of the original SKE. We highlight that this proof technique works only if we consider selective IND-CPA security. This is because the sampler needs to output an auxiliary input that is an honest encryption of m_b under the key k (hardcoded into C_0). This is fundamental to simulate the challenge ciphertext (of the selective IND-CPA experiment) and concludes the hybrid argument.

Why aren't diO/iO Sufficient for these Transformations Above? We observe that each of the compilation described above would not be feasible with either iO or diO. Intuitively, this is because we would eventually need to reduce the security of our transformations ($\mathsf{pv\text{-}SNARG}$, signature, PKE) to the security of the original secret-key primitive ($\mathsf{dv\text{-}SNARG}$, MAC, SKE). However, in the latter experiment the secret-key sk (e.g., a vrs or a symmetric-key), that we need to obfuscate in order to conclude the reduction, is sampled and kept secret by the challenger. This makes iO and diO insufficient since we are not able to satisfy their permissibility notion during this reduction. For the case of iO, during the reduction, the only thing we could do is to to obfuscate different circuit C_1 that does not use the secret-key sk sampled by the challenger. However, this C_1 will have (with overwhelming probability) a different input/output behavior compared to C_0 (the original obfuscated circuit of the transformation that, in turn, contains sk).

A similar discussion applies to diO. For the sake of concreteness, consider transforming a $\mathsf{dv\text{-}SNARG}$ into a $\mathsf{pv\text{-}SNARG}$ by publishing an obfuscation of the circuit C_0 which implements the $\mathsf{dv\text{-}SNARG}$ verification algorithm using an hard-coded verification key vrs. During the reduction to the security of the underlying scheme we are not allowed to use the secret verification key vrs. Thus, during the reduction, we can only move to a hybrid where we obfuscate a circuit C_1 that does not use the vrs. But then we cannot argue that it is hard to find differing-inputs for C_0, C_1. In this specific case, the distinguisher could simply produce proofs π for true statements x and submit them to the circuits. While C_0 (using the vrs) returns 1, C_1 (without the vrs) is unable to verify the proof and cannot return a consistent output. Similar arguments apply to the other transformations.

The Landscape of Limitations of $\mathsf{odiO}/\mathsf{oiO}$. The seminal work of [7,8] explores the boundaries of obfuscation in several directions. As it is well known they show that there are (not necessarily natural) computations which are impossible to obfuscate using VBB. Moreover, [7,8] also shows that VBB-obfuscation cannot be used for securely performing certain structure-preserving transformations. In this direction, they show a (contrived but secure) SKE that turns into an insecure PKE scheme when compiled using obfuscation. We show that the results of [7,8] can be extended to the setting of odiO and oiO. In particular, we show that there (unconditionally) exist samplers that are $\mathsf{odiO}/\mathsf{oiO}$ permissible but are not obfuscatable. Specifically we sample (somewhat contrived) circuits C_s with an embedded secret s that remains "hidden enough" when only oracle access is allowed (thus being $\mathsf{odiO}/\mathsf{oiO}$ permissible). We then show that, once given access to the obfuscated circuit, it becomes possible to "partially extract" this secret s. Finally, we show that (since this sampler cannot be obfuscated) our oiO-based transformation (5) (from semantically secure and key indistinguishable SKE to selectively IND-CPA PKE) is inherently impossible, regardless of the strength of the obfuscator used. This is done by using the unobfuscatable circuits to build a contrived SKE (satisfying semantic security and key indistinguishability) that, once compiled, yields an insecure PKE. As mentioned, a similar impossibility result was given in [8, Theorem 4.10]. However their contrived SKE does not satisfy key indistinguishability and, for this reason, it cannot be directly used to show the infeasibility of our transformation (5). Thus, our negative result strengthens the one of [8] since ours apply to a smaller class of SKEs (i.e., SKEs with stronger notions of security) that satisfy key indistinguishability under chosen message randomness attacks. Note that while we just argued that the oiO-based transformation in (5) is inherently impossible, our odiO-based transformations (1)-(4) remain plausible as the impossibility results do not seem to extend. We elaborate further in Sect. 6.2.

1.2 Future Directions

Our work opens up several interesting future directions. How to generally formalize structure-preserving transformations? Can we characterize what type of games can be transformed (from "secret" to "public" key) through odiO? Several, but not all those we achieve, seem to have a "forgery" flavor to them (MAC, NIZKs, etc.). What are further connections between our proposed notions of obfuscation and VBB, iO and diO? While the techniques in [8] seem to fail to show that some of our transformations are paradoxical, what are other techniques that could shed light on further limitations of odiO oiO? Can we leverage our techniques for going from secret-key to public-key variants of different cryptographic primitives than those we consider here, e.g., proofs of retrievability [42]?

2 Related Work

Barak et al. [7,8] investigate the feasibility of obfuscation. They focus on virtual black-box (VBB) obfuscation, where an obfuscated program/circuit should leak no information except for its input-output behaviour. They show: 1) that a general VBB obfuscator cannot exist since there are circuits that cannot be unconditionally obfuscated in the VBB paradigm; 2) that most of the intriguing applications of VBB are impossible (including the suggestion of Diffie and Hellman's of building a PKE by obfuscating the SKE encryption algorithm with an embedded symmetric key). On the positive side, several works have shown that some restricted classes of circuits can be securely VBB-obfuscated [23,33,43,45] . Goldwasser and Kalai [30,31] and Bitansky et al. [13] extended VBB's impossibility results to the case of auxiliary information demonstrating that other "natural" circuits cannot be VBB obfuscated when some (dependent or independent) auxiliary information are available. In addition, [13] demonstrated that the availability of auxiliary information is equivalent to VBB with universal simulation. Goldwasser and Rothblum [32] proposed the notion of best-possible obfuscation that guarantees that the obfuscation of a circuit leaks as little information as any other circuit implementing the same functionality. They show that a separation between VBB and best-possible obfuscation and an impossibility result (for both) in the random oracle model. Other works [6,20,22,37,38,40] studied the (in)feasibility of VBB in different idealized models.

To avoid the VBB paradigm (and its impossibility results), [8] suggested two weaker security definitions of obfuscation: indistinguishability obfuscation (iO) and differing-input obfuscation (diO). The former has obtained a lot of interest thanks to its applications, as initially shown by Sahai and Waters [41]. The first work that proposed a candidate iO construction is by Garg et al. [26] that built iO via multilinear maps. Subsequent works [2–4,15,16,19,28,29,36,39] focused on both the relations of iO and other primitives (e.g., functional encryption) and new candidates construction from weaker assumptions. These works led to the recent works of Jain et al. [35] and Wee and Wichs [44]. [35] built (sub-exponentially secure) iO from the sub-exponential hardness of LWE, learning parity with noise, and boolean pseudorandom generators in NC^0. On the other hand, [44] proposed the first construction based solely on lattices and LWE. Their construction relies on a new falsifiable LWE assumption.

As for diO, [1,10,17,17] proposed different formalization of diO (for both circuits and Turing machines) and showed different applications. On the negative side, [11,18,27] showed that, in the presence of (some) auxiliary information (e.g., samplers), a general diO obfuscator may not exist. Notably, Bellare et al. [11] showed that if sub-exponentially secure one-way functions exist then a sub-exponentially secure general diO obfuscator for Turing machines does not exist, i.e., there exists a sampler that outputs two Turing machines and some auxiliary information that cannot be obfuscated through diO. Moreover, they show that the impossibility result extends to diO for circuits, if SNARKs exist. Garg et al. [27] showed a similar result for diO for circuits under the conjecture that a special-purpose obfuscator exists (i.e., an obfuscator that does not follow from diO). All the negative results of [11,18,27] rely on the fact that the sampler can silently provide a trapdoor that allows an adversary to distinguish between two obfuscations whereas the trapdoor does not help in finding a differing-input., Because of this, Ishai et al. [34] proposed the weaker notion of public-coin diO where the random coins of the sampler are public, i.e., a sampler cannot hide any trapdoor in the auxiliary information.

Among weaker notions of obfuscation, we also find virtual gray-box obfuscation (VGB) [12,14]. This notion is close to that of VBB but models the simulator as semi-bounded, i.e., unbounded in running time but limited to a polynomial number of oracle queries. VGB is equivalent to another notion, strong iO (siO), where it holds that $\mathsf{Obf}(C_0) \approx_c \mathsf{Obf}(C_1)$ whenever the pair (C_0, C_1) is sampled from a concentrated distribution $\mathbf{D}$: For every input x, the probability that $C_0(x)$ and $C_1(x)$ do not return to common output $maj_{\mathbf{D}}(x)$ is negligible (where $maj_{\mathbf{D}}(\cdot)$ is defined with respect to the concentrated distribution $\mathbf{D}$ taken into account). Observe that concentrated distributions are a generalization of evasive functions [5]. Intuitively, siO is weaker than odiO (and oiO) since circuits (sampled from concentrated distributions) are oracle-diffing-input even against semi-bounded adversaries. Also, note that siO is not powerful enough to achieve structure-preserving transformations. Intuitively, because siO is able to obfuscate distributions of circuits that “pass” an information theoretical test. This is a obstacle when trying to implement our structure-preserving transformations since our objective is to compile/obfuscate primitives whose security follows from computational assumptions.

3 Preliminaries on Obfuscation

We assume the reader to be familiar with standard cryptographic notation and definitions. Our notation and all the standard definitions used in the paper can be found in the full version.

Indistinguishability Obfuscation and Differing-input Obfuscation. Let $\mathcal{C} = \{\mathcal{C}_\lambda\}_{\lambda \in \mathbb{N}}$ be an ensemble of functionally equivalent circuits (of same size), i.e., $\forall \lambda \in \mathbb{N}, \forall C_0, C_1 \in \mathcal{C}_\lambda, \forall x \in \{0,1\}^{\ell_{in}}$, $C_0(x) = C_1(x)$ and $|C_0| = |C_1|$. Indistinguishability obfuscation (iO) [7] guarantees that the obfuscation of any two functionally equivalent circuits $C_0, C_1 \in \mathcal{C}_\lambda$ are computationally indistinguishable. The stronger notion of differing-input obfuscation (diO) [1,8,17] considers the larger class of differing-input circuits, i.e., circuits that differ on hard to find inputs. Below, we introduce the definition of diO with respect to samplers responsible of sampling two differing-input circuits and (some) auxiliary information.

Definition 3.1. *A sampler* S *for an ensemble of circuits* $\mathcal{C} = \{\mathcal{C}_\lambda\}_{\lambda \in \mathbb{N}}$ *is a PPT algorithm that, on input the security parameter* 1^λ*, it outputs two circuits* $C_0, C_1 \in \mathcal{C}_\lambda$ *such that* $|C_0| = |C_1|$ *and (possibly) some auxiliary information* α.

Definition 3.2. *(*diO*-sampler) We say a sampler* S *(Definition 3.1) is a* diO*-sampler if for every PPT adversary* A *we have*

$$\mathbb{P}\left[C_0(x) \neq C_1(x) \middle| (C_0, C_1, \alpha) \leftarrow\!\!\$\ \mathsf{S}(1^\lambda), x \leftarrow\!\!\$\ \mathsf{A}(1^\lambda, C_0, C_1, \alpha)\right] \leq \mathsf{negl}(\lambda).$$

Definition 3.3. [Differing-input obfuscation] *Let* $\mathcal{S}$ *be an ensemble of* diO*-samplers (Definition 3.2). For every* $\mathsf{S} \in \mathcal{S}$*, let* $\mathcal{C}^{\mathsf{S}} = \{\mathcal{C}^{\mathsf{S}}_\lambda\}_{\lambda \in \mathbb{N}}$ *be the ensemble of circuits output by* S*. A PPT algorithm* Obf *is a* $(\mathcal{S})$-diO*-obfuscator for the ensemble* $\mathcal{S}$ *if the following conditions are satisfied:*

Correctness. $\forall \mathsf{S} \in \mathcal{S}$, $\forall \lambda \in \mathbb{N}$, $\forall C \in \mathcal{C}^{\mathsf{S}}_\lambda$, $\forall x \in \{0,1\}^{\ell_{in}}$*, we have* $C'(x) = C(x)$ *where* $C' \leftarrow\!\!\$\ \mathsf{Obf}(1^\lambda, C)$.

Polynomial slowdown. *There exists a polynomial* p *such that* $\forall \mathsf{S} \in \mathcal{S}$, $\forall C \in \mathcal{C}^{\mathsf{S}}_\lambda$*, we have* $|\mathsf{Obf}(1^\lambda, C)| \leq p(|C|)$.

Indistinguishability. *For every* $\mathsf{S} \in \mathcal{S}$*, every PPT adversary* D*, we have that*

$$\left|\mathbb{P}\left[\mathsf{D}(1^\lambda, \mathsf{Obf}(1^\lambda, C_0), \alpha) = 1\right] - \mathbb{P}\left[\mathsf{D}(1^\lambda, \mathsf{Obf}(1^\lambda, C_1), \alpha) = 1\right]\right| \leq \mathsf{negl}(\lambda),$$

where $(C_0, C_1, \alpha) \leftarrow\!\!\$\ \mathsf{S}(1^\lambda)$.

The above definition is parametrized by an ensemble of diO-samplers since some negative results for diO are known [11,27] (see next). Because of this, an *universal* (general) diO-obfuscator may not exists, i.e., a diO-obfuscator that obfuscates any diO-sampler.

Negative Results. In the setting of Turing machines (not covered by this paper), Bellare et al. [11] show that if sub-exponentially secure one-way functions exist then a sub-exponentially secure diO-obfuscator Obf for any sampler for Turing machines does not exist (i.e., there exists a particular sampler that cannot be diO-obfuscated). We stress that the main impossibility result covers Turing machines but, as described by [11], if SNARKs exist the negative result can be extended to diO for circuits. Garg et al. [27] show that under the conjecture that a special-purpose obfuscator exists (i.e., an obfuscator that does not follow from the existence of a diO-obfuscator) then a diO-obfuscator Obf for any sampler for circuits does not exist. We highlight that both [11,27] show that only "some" diO-samplers cannot be obfuscated. Indeed, both works rely on samplers that output complex auxiliary information α (α is itself an obfuscation of contrived circuit/Turing machine). Hence, this does not rule out the possibility of obfuscating the same class of circuits/Turing machines under simpler auxiliary information.

Virtual Black-box Obfuscation. Virtual black-box obfuscation (VBB) [7], is the strongest known notion of obfuscation. In a nutshell, a VBB-obfuscator guarantees that having an obfuscation of a circuit C is "equivalent" to having oracle access to C. We consider the weakest notion of VBB that requires the adversary (and the simulator) to output a single bit. This is equivalent to asking the adversary/simulator to compute/determine an arbitrary predicate $\pi(C)$ of the original circuit [7]. Similarly to diO, we consider VBB with respect to samplers responsible to sample a circuit and (some) auxiliary information. This will allow us to provide a meaningful comparison between VBB and diO, odiO, oiO.

Definition 3.4. *(VBB-sampler) A VBB-sampler S for an ensemble of circuits $\mathcal{C} = \{\mathcal{C}_\lambda\}_{\lambda \in \mathbb{N}}$ is a PPT algorithm that, on input the security parameter 1^λ, it outputs a circuit $C \in \mathcal{C}_\lambda$ and some auxiliary information α.*

Definition 3.5 (Virtual black-box obfuscation). *Let $\mathcal{S}$ be an ensemble of VBB-samplers (Definition 3.4). For every $\mathsf{S} \in \mathcal{S}$, let $\mathcal{C}^{\mathsf{S}} = \{\mathcal{C}^{\mathsf{S}}_\lambda\}_{\lambda \in \mathbb{N}}$ be the ensemble of circuits output by S. A PPT algorithm Obf is a $(\mathcal{S})$-VBB-obfuscator for the ensemble $\mathcal{S}$ if the following conditions are satisfied:*

Correctness. *$\forall \mathsf{S} \in \mathcal{S}$, $\forall \lambda \in \mathbb{N}$, $\forall C \in \mathcal{C}^{\mathsf{S}}_\lambda$, $\forall x \in \{0,1\}^{\ell_{in}}$, we have $C'(x) = C(x)$ where $C' \leftarrow_{\$} \mathsf{Obf}(1^\lambda, C)$.*

Polynomial slowdown. *There exists a polynomial p such that $\forall \mathsf{S} \in \mathcal{S}$, $\forall C \in \mathcal{C}_\lambda$, we have $|\mathsf{Obf}(1^\lambda, C)| \leq p(|C|)$.*

Virtual black-box simulation. *For every PPT adversary A, there exists a PPT simulator Sim such that for every $\mathsf{S} \in \mathcal{S}$, we have*

$$\left| \mathbb{P}\left[\mathsf{A}(1^\lambda, \mathsf{Obf}(1^\lambda, C), \alpha) = 1\right] - \mathbb{P}\left[\mathsf{Sim}^{C(\cdot)}(1^\lambda, 1^{|C|}, \alpha) = 1\right] \right| \leq \mathsf{negl}(\lambda),$$

where $(C, \alpha) \leftarrow_{\$} \mathsf{S}(1^\lambda)$.

Note that VBB is a much stronger flavor of obfuscation than diO and iO for two reasons. First, VBB defines the concept of ideal/oracle obfuscation, i.e., an obfuscated circuit behaves as an oracle. Second, VBB is a simulation-based definition (whereas both iO and diO are indistinguishability-based), i.e., any bit of leakage (that can be retrieved from the obfuscation of a circuit) can be simulated (except with negligible probability) having only oracle access to the unobfuscated circuit.

Impossibility Results. VBB is a very interesting notion of obfuscation since it has several important applications (e.g., it permits to convert a SKE into PKE). However, VBB-obfuscation turned out to be impossible for several and reasonably simple class of circuits/samplers [7,8,13]. Moreover, also several applications of

VBB are impossible to achieve. As an example, Barak et al. [8, Theorem 4.10] have shown that there exist a SKE that cannot be transformed into a PKE by (simply) obfuscating the SKE's encryption algorithm (a similar impossibility result applies also to PRFs, MACs, and signatures). Still, VBB-obfuscation is still possible for other class of circuits/samplers. Examples are compute-and-compare programs [45] (also known as lockable obfuscation [33]) and point functions [43].

4 Oracle-Differing-Input and Oracle-Indistinguishability Obfuscation

In this section, we propose two new notions of obfuscation, dubbed *oracle-differing-input obfuscation* and *oracle-indistinguishability obfuscation* (odiO and oiO in short). Both odiO and oiO are the result of two natural extensions of diO (resp. iO): they introduce the notion of oracle circuits (as in VBB) while keeping the indistinguishability property of diO (resp. iO). In a nutshell, odiO requires that the obfuscations of two circuits C_0, C_1 are computationally indistinguishable if the latter two are differing-input circuits when treated as oracles, i.e., an adversary cannot find an input x such that $C_0(x) \neq C_1(x)$ when given oracle access to both C_0 and C_1. On the other hand, oiO provides the same indistinguishability guarantee with respect to circuits C_0, C_1 that are computationally indistinguishable when treated as oracles.

As usual, we define odiO and oiO with respect to an ensemble of samplers responsible of generating the circuits C_0, C_1 and (possibly) some auxiliary information α.

Definition 4.1. *(odiO- and oiO-sampler) Let $\mathsf{type} \in \{\mathsf{odiO}, \mathsf{oiO}\}$. We say a sampler S (Definition 3.1) is an type-sampler if for every PPT adversary A we have*

If $\mathsf{type} = \mathsf{odiO}$**:** $\mathbb{P}\left[C_0(x) \neq C_1(x) \middle| x \leftarrow\!\!\$\ \mathsf{A}^{C_0(\cdot), C_1(\cdot)}(1^\lambda, 1^{|C_0|}, \alpha)\right] \leq \mathsf{negl}(\lambda)$,

If $\mathsf{type} = \mathsf{oiO}$**:** $\left|\mathbb{P}\left[\mathsf{A}^{C_0(\cdot)}(1^\lambda, 1^{|C_0|}, \alpha) = 1\right] - \mathbb{P}\left[\mathsf{A}^{C_1(\cdot)}(1^\lambda, 1^{|C_1|}, \alpha) = 1\right]\right| \leq \mathsf{negl}(\lambda)$,

where $(C_0, C_1, \alpha) \leftarrow\!\!\$\ \mathsf{S}(1^\lambda)$.[10]

Definition 4.2 (Oracle-differing-input and oracle-indistinguishability obfuscation). *For $\mathsf{type} \in \{\mathsf{odiO}, \mathsf{oiO}\}$, let $\mathcal{S}$ be an ensemble of type-samplers (Definition 4.1). For every $\mathsf{S} \in \mathcal{S}$, let $\mathcal{C}^{\mathsf{S}} = \{\mathcal{C}^{\mathsf{S}}_\lambda\}_{\lambda \in \mathbb{N}}$ be the ensemble of circuits output by S. A PPT algorithm Obf is a $(\mathcal{S})$-type-obfuscator for the ensemble $\mathcal{S}$ if the following conditions are satisfied:*

Correctness. *$\forall \mathsf{S} \in \mathcal{S}$, $\forall \lambda \in \mathbb{N}$, $\forall C \in \mathcal{C}^{\mathsf{S}}_\lambda$, $\forall x \in \{0,1\}^{\ell_{in}}$, we have $C'(x) = C(x)$ where $C' \leftarrow\!\!\$\ \mathsf{Obf}(1^\lambda, C)$.*

Polynomial slowdown. *There exists a polynomial p such that $\forall \mathsf{S} \in \mathcal{S}$, $\forall C \in \mathcal{C}^{\mathsf{S}}_\lambda$, we have $|\mathsf{Obf}(1^\lambda, C)| \leq p(|C|)$.*

[10] Recall that $|C_0| = |C_1|$ by definition of sampler (Definition 3.1).

Indistinguishability. *For every* $\mathsf{S} \in \mathcal{S}$*, every PPT adversary* D*, we have that*

$$\left|\mathbb{P}\left[\mathsf{D}(1^\lambda, \mathsf{Obf}(1^\lambda, C_0), \alpha) = 1\right] - \mathbb{P}\left[\mathsf{D}(1^\lambda, \mathsf{Obf}(1^\lambda, C_1), \alpha) = 1\right]\right| \leq \mathsf{negl}(\lambda),$$

where $(C_0, C_1, \alpha) \leftarrow\!\!\$\ \mathsf{S}(1^\lambda)$*.*

Comparing diO**-,** odiO**-,** oiO**-, and** VBB**-obfuscation.** We now study the relations between diO, odiO, oiO, and VBB. In order to provide a meaningful comparison, we work in terms of *best-possible universal* obfuscators, i.e., we compare the classes of circuits/samplers that each flavor of obfuscation is able to handle. We start by defining the notion of *best-possible universal* type-*obfuscator* Obf (for $\mathsf{type} \in \{\mathsf{diO}, \mathsf{odiO}, \mathsf{oiO}, \mathsf{VBB}\}$) whose definition is tied with the (universal) set $\mathcal{S}_{\mathsf{type}}$ composed of all the type-samplers that can be securely type-obfuscated (as defined in Definitions 4.2 to 3.4).

Definition 4.3 (Best-possible universal type**-obfuscator).** *Let* $\mathsf{type} \in \{\mathsf{diO}, \mathsf{odiO}, \mathsf{oiO}, \mathsf{VBB}\}$*. Consider the ensemble* $\mathcal{S}_{\mathsf{type}}$ *composed of every* type*-sampler* S *(Definitions 4.1, 3.2 and 3.4) that can be securely* type*-obfuscated (Definitions 4.2, 3.3 and 3.5), i.e.,*

$$\mathcal{S}_{\mathsf{type}} = \{\mathsf{type}\text{-}\textit{sampler}\ \mathsf{S} \mid \exists\ \mathsf{Obf}\ \textit{s.t.}\ \mathsf{Obf}\ \textit{is a}\ (\{\mathsf{S}\})\text{-}\mathsf{type}\text{-}\textit{obfuscator}\}.$$

A PPT algorithm Obf *is a* best-possible universal type-obfuscator *if* Obf *is a* $(\mathcal{S}_{\mathsf{type}})$-$\mathsf{type}$*-obfuscator (Definitions 3.3, 4.2 and 3.5).*

Remark 4.4. There are two technical reasons behind the need of considering only best-possible universal obfuscators, while comparing diO, odiO, oiO, and VBB. First, for any notion of type-obfuscation, it is possible to find two contrived type-obfuscators Obf_0 and Obf_1 that result to be incomparable, even within the same flavor of obfuscation. As an example, we could have that Obf_0 (resp. Obf_1) is able to type-obfuscate S_0 (resp. S_1) but not S_1 (resp. S_0) where $\mathsf{S}_0, \mathsf{S}_1$ are two type-samplers.[11] The same argument holds between different notions. For example, if we consider diO and odiO, we could have that Obf_0 diO-obfuscates a diO-sampler S (that in turn, as we will see, is also a odiO-obfuscator) but Obf_1 does not odiO-obfuscate S. Also, we can have the symmetric case: there exist two obfuscators Obf'_0 and Obf'_1 such that Obf'_1 odiO-obfuscates S but Obf'_0 does not diO-obfuscate S. Hence by changing the obfuscator we could reach any conclusions: (*i*) odiO and diO are incomparable, (*ii*) odiO implies diO, or (*iii*) diO implies odiO. This clearly does not allow for a meaningful comparison. Definition 4.3 naturally solves the above problem since a best-possible universal type-obfuscator uniquely represents the power of a particular notion of obfuscation, i.e., the set $\mathcal{S}_{\mathsf{type}}$ of samplers that can be securely type-obfuscated. This allows us to have a meaninful (and unique) formal comparison between diO, odiO, oiO, and VBB.

[11] For instance, we can have that S_b only outputs circuits whose description starts with a bit b, and that Obf_b rejects any circuit whose description starts with the bit $1-b$.

Second, Definition 4.3 allows us to exclude from the comparison the known impossibility results of VBB [7,13] (and odiO, oiO as we will show in Sect. 6). This is because, instead of quantifying over any possible type-sampler, best-possible universal type-obfuscation is defined over any possible type-sampler that can be type-obfuscated.

In the setting of best-possible universal obfuscation, odiO (resp. oiO) is stronger than diO since (*i*) any diO-sampler is also an odiO-sampler (resp. oiO-sampler) and (*ii*) both diO and odiO (resp. oiO) have the same indistinguishability-based security definition. The same argument applies to odiO and oiO, i.e., oiO is stronger than odiO.

Theorem 4.5 **($\mathsf{oiO} \Rightarrow \mathsf{odiO} \Rightarrow \mathsf{diO}$).** *For* $\mathsf{type} \in \{\mathsf{diO}, \mathsf{odiO}, \mathsf{oiO}\}$*, we have that* $\mathcal{S}_{\mathsf{diO}} \subseteq \mathcal{S}_{\mathsf{odiO}} \subseteq \mathcal{S}_{\mathsf{oiO}}$ *where* $\mathcal{S}_{\mathsf{type}}$ *as defined in Definition 4.3.*

The proof of this theorem is deferred to full version.

About (best-possible universal) odiO-, oiO-, and VBB-obfuscation, we have that VBB is stronger than odiO (resp. oiO) for two main reasons:

1. VBB leverages a simulation-based definition: any bit of information that can be leaked from an obfuscated circuit C can be simulated by only having oracle access to C. On the other hand, odiO (resp. oiO) provides a much weaker security guarantee: the obfuscation of two circuits C_0, C_1 (output by an odiO-sampler (resp. oiO-sampler)) are computationally indistinguishable. This implies that a odiO-obfuscator (resp. oiO-obfuscator) could leak significant information about the circuit, as long as the leaked information does not help in distinguishing (except with negligible probability) between the obfuscations of C_0 and C_1.
2. Both VBB and odiO (resp. oiO) incorporate the notion of oracle circuits in their definitions. However, oracles are used to define two different concepts. VBB uses oracle circuits to define the amount of information a VBB-obfuscator may leak. Since oracles leak no information (except their input-output behavior), this implies that a VBB-obfuscator does not leak any information, except with negligible probability.
   Conversely, odiO and oiO leverage the notion of oracle circuits to characterize the class of circuits (or samplers) that an odiO-/oiO-obfuscator can handle. The definition of security (i.e., the indistinguishability property of Definition 4.2) is independent from the oracles. Both odiO and oiO "only" guarantee that the information leaked by the obfuscation of two circuits are the same. This does not imply that the odiO-/oiO-obfuscated circuits must "behave" as oracles (as required by VBB (Definition 3.5)).

The relation between VBB, oiO, and odiO is formalized by the following theorem, whose proof is deferred to full version.

Theorem 4.6 **($\mathsf{VBB} \Rightarrow \mathsf{oiO}$ and $\mathsf{VBB} \Rightarrow \mathsf{odiO}$).** *Let* S *be a sampler (Definition 3.1). For* $b \in \{0,1\}$*, let* S_b *be a sampler such that* $(C_b, \alpha) = \mathsf{S}_b(1^\lambda; r)$ *where* $r \in \{0,1\}^*$*, and* $(C_0, C_1, \alpha) = \mathsf{S}(1^\lambda; r)$*. If* $\mathsf{S}_0, \mathsf{S}_1 \in \mathcal{S}_{\mathsf{VBB}}$ *then* $\mathsf{S} \in \mathcal{S}_{\mathsf{type}}$ *where* $\mathcal{S}_{\mathsf{VBB}}$ *and* $\mathcal{S}_{\mathsf{type}}$ *are defined in Definition 4.3.*

By leveraging a similar argument to that used to prove Theorem 4.5, we can demonstrate that any negative result for diO extends to odiO. This because any diO-sampler S is also an odiO-sampler and, since diO and odiO leverage the same indistinguishability-based definition, if $\mathsf{S} \notin \mathcal{S}_{\mathsf{diO}}$ then $\mathsf{S} \notin \mathcal{S}_{\mathsf{odiO}}$.[12] The same applies between odiO and oiO, and between oiO and VBB (with respect to samplers as defined in Thoerem 4.6).

Corollary 4.7. *For* $\mathsf{type} \in \{\mathsf{diO}, \mathsf{odiO}, \mathsf{oiO}, \mathsf{VBB}\}$*, let* $\mathcal{S}_{\mathsf{type}}$ *be an ensemble of* type*-samplers as defined in Definition 4.3. The following conditions holds:*

1. *For every* diO*-sampler* S *such that* $\mathsf{S} \notin \mathcal{S}_{\mathsf{diO}}$ *then* $\mathsf{S} \notin \mathcal{S}_{\mathsf{odiO}}$*.*
2. *For every* odiO*-sampler* S *such that* $\mathsf{S} \notin \mathcal{S}_{\mathsf{odiO}}$ *then* $\mathsf{S} \notin \mathcal{S}_{\mathsf{oiO}}$*.*
3. *For every* oiO*-sampler* S *and every pair of* VBB*-samplers* $(\mathsf{S}_0, \mathsf{S}_1)$ *such that* $(C_b, \alpha) = \mathsf{S}_b(1^\lambda; r)$ *where* $r \in \{0,1\}^*$*,* $(C_0, C_1, \alpha) = \mathsf{S}(1^\lambda; r)$ *and* $b \in \{0,1\}$ *(as defined in Theorem 4.6), if* $\mathsf{S} \notin \mathcal{S}_{\mathsf{oiO}}$ *then* $\mathsf{S}_0 \notin \mathcal{S}_{\mathsf{VBB}}$ *or* $\mathsf{S}_1 \notin \mathcal{S}_{\mathsf{VBB}}$*.*

Lastly, odiO (resp. oiO) does not imply VBB, i.e., both odiO and oiO are strictly weaker than VBB. This follows by leveraging two observations. First, Barak et al. [7, Lemma 3.5, Corollary 3.8] have demonstrated that there (unconditionally) exists a distribution of circuits that cannot be VBB-obfuscated (see also Sect. 6.1). This, in turn, implies that there exists a VBB-sampler $\mathsf{S}_0 \notin \mathcal{S}_{\mathsf{VBB}}$, i.e., S_0 outputs $(C, \perp)$ where C comes from the distribution of [7, Lemma 3.5]. Second, we have that any sampler S_1, that outputs $(C_0, C_1, \perp)$ such that $C_0 = C_1$, is an odiO-sampler (resp. oiO-sampler) that can be easily odiO-obfuscated (resp. oiO-obfuscated).[13] By combining these two observations, we conclude that if S_1 outputs $(C_0, C_1, \perp)$ where $C_0 = C_1$ and $(C_0, \perp) \leftarrow\$\ \mathsf{S}_0(1^\lambda)$, it follows that neither C_0 nor C_1 (sampled by S_0) can be VBB-obfuscated but S_1 can be odiO-obfuscated (resp. oiO-obfuscated). While this counterexample might be trivial at first sight, it indeed captures the fact that an odiO-/oiO-obfuscator is allowed to reveal any information which is common to the two circuits, as long as this information does not allow to win the respective distinguishing game between the oracles.

Theorem 4.8 ($\mathsf{odiO} \not\Rightarrow \mathsf{VBB}$ **and** $\mathsf{oiO} \not\Rightarrow \mathsf{VBB}$)**.** *Let* S_0 *be a* VBB*-sampler (Definition 3.4). Consider the* odiO*-sampler (resp.* oiO*-sampler)* S_1 *defined as* $(C_0, C_1, \alpha) = \mathsf{S}_1(1^\lambda; r)$ *where* $C_0 = C_1$ *and* $(C_0, \alpha) = \mathsf{S}_0(1^\lambda; r)$ *for* $r \in \{0,1\}^*$*. For* $\mathsf{type} \in \{\mathsf{odiO}, \mathsf{oiO}\}$*, there exists a* VBB*-sampler* S_0 *such that* $\mathsf{S}_0 \notin \mathcal{S}_{\mathsf{VBB}}$ *and* $\mathsf{S}_1 \in \mathcal{S}_{\mathsf{type}}$ *where* $\mathcal{S}_{\mathsf{VBB}}$ *and* $\mathcal{S}_{\mathsf{type}}$ *as defined in Definition 4.3.*

5 Applications of odiO and oiO

In this section, we show that odiO and oiO are able to compile several symmetric key primitives into their corresponding public key versions and designated verifier non-interactive argument systems into their public verifiable version. These

[12] Otherwise, if $\mathsf{S} \in \mathcal{S}_{\mathsf{odiO}}$, there exists a $(\{\mathsf{S}\})$-odiO-obfuscator that in turn is also a $(\{\mathsf{S}\})$-diO-obfuscator.

[13] Indeed, any PPT obfuscator Obf that satisfies correctness and polynomial slowdown is a $(\{\mathsf{S}\})$-odiO-obfuscator (resp. $(\{\mathsf{S}\})$-oiO-obfuscator), e.g., Obf is the identity function or Obf is an iO-obfuscator.

transformations achieve (and use) different flavors of security whose definitions can be found in the full version of this paper. In more details, we demonstrate the following transformations:

Function-Preserving PV-NIZK from DV-NIZK: odiO is able to compile any designated verifier non-interactive argument system (that satisfies either selective soundness or straight-line knowledge soundness) into its public verifiable version (Sect. 5.1).

Function-Preserving Signatures from MACs: odiO is able to compile any (q)-sEUF-sel-CMA MAC into a (q)-sEUF-sel-CMA signature scheme (full version).

Format-Preserving Signatures from MACs: odiO is able to compile EUF MAC into a sel-EUF-CMA digital signature scheme, using puncturable PRF (full version).

Format-Preserving PKE from IV-based SKE: odiO is able to compile semantically secure IV-based SKE (i.e., SKE whose encryption algorithm has the following sintax $\mathsf{Enc}(\mathsf{k}, m; \mathsf{iv}) = (\mathsf{iv}, c)$) into a sel-IND-CPA PKE, using puncturable PRF (full version).

Function-Preserving PKE from SKE: oiO is able to compile any semantically and sel-IND-CPRA-key secure SKE into a sel-IND-CPA PKE (Sect. 5.2).

Note that transformations that use the puncturable PRFs are only format-preserving whereas the others are fully function-preserving.

We show the first and the last of our applications in detail in the main body; proofs and the remaining applications are deferred to the full version of this work.

5.1 From Designated Verifier to Public Verifiable Non-interactive Argument Systems

Construction 1. *Let $\Pi^* = (\mathsf{Setup}^*, \mathsf{Prove}^*, \mathsf{Verify}^*)$ and Obf be a DV non-interactive argument system for a relation $\mathcal{R}$ and an obfuscator, respectively. We compile Π^* into a PV non-interactive argument system $\Pi = (\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify})$ for the same relation $\mathcal{R}$ as follows:*

$\mathsf{Setup}(1^\lambda, \mathcal{R})$**:** *On input the security parameter 1^λ and a relation $\mathcal{R}$, the setup algorithm computes $(\mathsf{crs}^*, \mathsf{vrs}^*) \leftarrow\!\!\$\ \mathsf{Setup}^*(1^\lambda, \mathcal{R})$ and outputs $\mathsf{crs} = \mathsf{crs}^*$ and $\mathsf{vrs} = \widetilde{C}$ where $\widetilde{C} \leftarrow\!\!\$\ \mathsf{Obf}(1^\lambda, C^{\mathsf{Verify}}_{\mathsf{vrs}^*})$ and $C^{\mathsf{Verify}}_{\mathsf{vrs}}$ is depicted in Fig. 2.*

$\mathsf{Prove}(\mathsf{crs}, x, \omega)$**:** *On input the common reference string $\mathsf{crs} = \mathsf{crs}^*$, a statement x, and a witness ω, the prover algorithm outputs $\pi \leftarrow\!\!\$\ \mathsf{Prove}^*(\mathsf{crs}^*, x, \omega)$.*

$\mathsf{Verify}(\mathsf{vrs}, x, \pi)$**:** *On input the verification key $\mathsf{vrs} = \widetilde{C}$, a statement x, and a proof π, the verification algorithm returns $b = \widetilde{C}(x, \pi)$.*

Below we establish the following result.

```
$C^{\mathsf{Verify}}_{\mathsf{vrs}}(x, \pi)$
return $b = \mathsf{Verify}^*(\mathsf{vrs}, x, \pi)$

$C^{\mathsf{Verify}}_{\mathsf{vrs},x^*}(x, \pi)$
If $x = x^*$, return $0$
return $b = \mathsf{Verify}^*(\mathsf{vrs}, x, \pi)$

$C^{\mathsf{Verify}}_{\mathsf{vrs},\mathsf{td},r}(x, \pi)$
$\omega = \mathsf{Ext}^*_1(1^\lambda, \mathsf{td}, x, \pi; r)$
If $\mathsf{Verify}^*(\mathsf{vrs}, x, \pi) = 1$ and
    $(x, \omega) \in \mathcal{R}$, return $1$
return $0$
```

```
$\mathsf{S}_x(1^\lambda; r)$
$(\mathsf{crs}, \mathsf{vrs}) = \mathsf{Setup}^*(1^\lambda; r)$
Set $C_0 = C^{\mathsf{Verify}}_{\mathsf{vrs}}$, $C_1 = C^{\mathsf{Verify}}_{\mathsf{vrs},x}$, $\alpha = \mathsf{crs}$
return $(C_0, C_1, \alpha)$

$\mathsf{S}_{\mathsf{Ext}^*}(1^\lambda; r)$
Let $r = (r_0, r_1)$
$(\mathsf{crs}, \mathsf{vrs}, \mathsf{td}) = \mathsf{Ext}^*_0(1^\lambda, \mathcal{R}; r_0)$
Set $C_0 = C^{\mathsf{Verify}}_{\mathsf{vrs}}$, $C_1 = C^{\mathsf{Verify}}_{\mathsf{vrs},\mathsf{td},r_1}$, $\alpha = \mathsf{crs}$
return $(C_0, C_1, \alpha)$
```

Fig. 2. The circuits $C^{\mathsf{Verify}}_{\mathsf{vrs}}$, $C^{\mathsf{Verify}}_{\mathsf{vrs},x^*}$, $C^{\mathsf{Verify}}_{\mathsf{vrs},\mathsf{td},r}$, and the samplers S_x,$\mathsf{S}_{\mathsf{Ext}^*}$. $C^{\mathsf{Verify}}_{\mathsf{vrs}}$ and $C^{\mathsf{Verify}}_{\mathsf{vrs},x^*}$ (resp. $C^{\mathsf{Verify}}_{\mathsf{vrs}}$ and $C^{\mathsf{Verify}}_{\mathsf{vrs},\mathsf{td},r}$) are padded to match the size $\gamma = \max\{|C^{\mathsf{Verify}}_{\mathsf{vrs}}|, |C^{\mathsf{Verify}}_{\mathsf{vrs},x^*}|\}$ (resp. $\gamma = \max\{|C^{\mathsf{Verify}}_{\mathsf{vrs}}|, |C^{\mathsf{Verify}}_{\mathsf{vrs},\mathsf{td},r}|\}$).

Theorem 5.1. *Let Π^* and* Obf *as defined in Construction 1. For every $x \notin \mathcal{L}$, consider the sampler S_x depicted in Fig. 2.*

1. *If Π^* satisfies selective soundness then, for every $x \notin \mathcal{L}$, S_x is an* odiO*-sampler (Definition 4.1), and*
2. *if* Obf *is a $(\{\mathsf{S}_x\}_{x \notin \mathcal{L}})$-*$\mathsf{odiO}$*-obfuscator (Definition 4.2) then the publicly verifiable non-interactive argument system Π of Construction 1 satisfies selective soundness.*

We extend the above result to the case of straight-line knowledge soundness.

Theorem 5.2. *Let Π^* and* Obf *as defined in Construction 1.*

1. *If Π^* satisfies straight-line knowledge soundness then the sampler $\mathsf{S}_{\mathsf{Ext}^*}$ of Fig. 2 is an* odiO*-sampler (Definition 4.1) where $\mathsf{Ext}^* = (\mathsf{Ext}^*_0, \mathsf{Ext}^*_1)$ is the PPT extractor of Π^*, and*
2. *if* Obf *is a $(\{\mathsf{S}_{\mathsf{Ext}^*}\})$-*$\mathsf{odiO}$*-obfuscator (Definition 4.2) then the publicly verifiable non-interactive argument system Π of Construction 1 satisfies straight-line knowledge soundness.*

Remark 5.3 (On zero-knowledge). Observe that Construction 1 preserves zero-knowledge if the underlying designated verifier non-interactive argument system Π^* is zero-knowledge. This is straightforward and follows intuitively because Construction 1 only obfuscates vrs (that it is known by a malicious verifier against zero-knowledge) and it does not alter Π^*'s Prove. A proof sketch of the zero-knowledge property would be as follows. The simulator for the publicly verifiable case is the same as the one for the designated verifier case. Now assume there exists an adversary A_{pv} distinguishing simulated proofs from honest ones. We could then design adversary A_{dv} breaking zero-knowledge of the

```
$C^{\mathsf{Enc}}_{\mathsf{k}}(m, r)$
return $c = \mathsf{Enc}^*(\mathsf{k}, m; r)$

$\mathsf{S}_m(1^\lambda; r)$
Let $r = (r_0, r_1, r_2)$
$\mathsf{k}_0 = \mathsf{KGen}^*(1^\lambda; r_0), \mathsf{k}_1 = \mathsf{KGen}^*(1^\lambda; r_1)$
$c = \mathsf{Enc}^*(\mathsf{k}_0, m; r_2)$
Set $C_0 = C^{\mathsf{Enc}}_{\mathsf{k}_0}, C_1 = C^{\mathsf{Enc}}_{\mathsf{k}_1}, \alpha = c$
return $(C_0, C_1, \alpha)$
```

Fig. 3. The circuit $C^{\mathsf{Enc}}_{\mathsf{k}}$ and the sampler S_m. $C^{\mathsf{Enc}}_{\mathsf{k}_0}$ and $C^{\mathsf{Enc}}_{\mathsf{k}_1}$ (output by S_m) are padded to match the size $\gamma = \mathsf{max}\{|C^{\mathsf{Enc}}_{\mathsf{k}_0}|, |C^{\mathsf{Enc}}_{\mathsf{k}_1}|\}$)

original scheme. This adversary can in fact internally run A_{pv} passing to it the obfuscation $\mathsf{Obf}(1^\lambda, C^{\mathsf{Verify}}_{\mathsf{vrs}})$. It can do that because the designated-verifier zero-knowledge has access to vrs.

5.2 From Semantically and sel-IND-CPRA-key SKEs to sel-IND-CPA PKEs

Construction 2. *Let $\Pi^* = (\mathsf{KGen}^*, \mathsf{Enc}^*, \mathsf{Dec}^*)$ and Obf be a SKE with message space $\mathcal{M}$ and an obfuscator, respectively. We compile Π^* into a PKE scheme $\Pi = (\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ with message space $\mathcal{M}$ as follows:*

$\mathsf{KGen}(1^\lambda)$**:** *On input the security parameter 1^λ, the key generation algorithm computes $\mathsf{k}^* \leftarrow\!\$\, \mathsf{KGen}^*(1^\lambda)$ and outputs $\mathsf{pk} = \widetilde{C}$ and $\mathsf{sk} = \mathsf{k}^*$ where $\widetilde{C} \leftarrow\!\$\, \mathsf{Obf}(1^\lambda, C^{\mathsf{Enc}}_{\mathsf{k}^*})$ and $C^{\mathsf{Enc}}_{\mathsf{k}}$ is depicted in Fig. 3.*

$\mathsf{Enc}(\mathsf{pk}, m; r)$**:** *On input the public key $\mathsf{pk} = \widetilde{C}$, a message $m \in \mathcal{M}$, and randomness $r \in \{0,1\}^*$, the encryption algorithm outputs $c = \widetilde{C}(m, r)$.*

$\mathsf{Dec}(\mathsf{sk}, c)$**:** *On input the secret key $\mathsf{sk} = \mathsf{k}^*$ and a ciphertext c, the deterministic decryption algorithm returns $m = \mathsf{Dec}^*(\mathsf{k}^*, c)$.*

Below we establish the following result.

Theorem 5.4. *Let Π^* and Obf as defined in Construction 2. For every $m \in \mathcal{M}$, consider the sampler S_m depicted in Fig. 3.*

1. *If Π^* is sel-IND-CPRA-key then, for every $m \in \mathcal{M}$, S_m is an oiO-sampler (Definition 4.1), and*
2. *If Π^* is semantically secure and Obf is a $(\{\mathsf{S}_m\}_{m \in \mathcal{M}})$-$\mathsf{oiO}$-obfuscator (Definition 4.2) then the PKE scheme Π of Construction 2 is sel-IND-CPA.*

```
$C^0_{\mathsf{k},a,b}(x, r)$
If $x = a$, return $b$
return $\mathsf{Enc}_0(\mathsf{k}, 0; r)$
```

```
$C^1_{\mathsf{k},a}(i, r)$
Let $a = a_1 || \dots || a_\lambda$
return $\mathsf{Enc}_0(\mathsf{k}, a_i; r)$
```

```
$C^2_{\mathsf{k}}(c_1, c_2, \odot, r)$
$x = \mathsf{Dec}_0(\mathsf{k}, c_1) \odot \mathsf{Dec}_0(\mathsf{k}, c_2)$
return $\mathsf{Enc}_0(\mathsf{k}, x; r)$
```

```
$C^3_{\mathsf{k},a,b,\mathsf{y},e}(d_1, \dots, d_\lambda, r)$
Let $b = b_1 || \dots || b_\lambda$
For $i \in [\lambda]$ do:
  If $\mathsf{Dec}_0(\mathsf{k}, d_i) \neq b_i$,
        return $\mathsf{Enc}_0(\mathsf{k}, 0; r)$
return $(\mathsf{k}, a, \mathsf{y}, e)$
```

```
$C^*_{\mathsf{s},(\mathsf{k},a,b,\mathsf{y},e)}(\ell, v, r)$
Let $v = (x, i, c_1, c_2, \odot, d_1, \dots, d_\lambda)$
$r' = \mathsf{F}_1(\mathsf{s}, (\ell, v, r))$
If $\ell = 0$, return $C^0_{\mathsf{k},a,b}(x, r')$
If $\ell = 1$, return $C^1_{\mathsf{k},a}(i, r')$
If $\ell = 2$, return $C^2_{\mathsf{k}}(c_1, c_2, \odot, r')$
If $\ell = 3$, return $C^3_{\mathsf{k},a,b,\mathsf{y},e}(d_1, \dots, d_\lambda, r')$
```

Fig. 4. The circuit $C^*_{\mathsf{s},(\mathsf{k},a,b,\mathsf{y},e)}$ where $(\mathsf{s}, \mathsf{k}, a, b, \mathsf{y}, e) \in \{0,1\}^{5\lambda+1}$ and $\odot$ is the binary representation of a 2×2 table of an arbitrary binary operator (e.g., AND, OR, NOT).

6 Extending the Impossibility Results of Barak et al. [7,8] to the Setting of odiO and oiO

In Sect. 4, we have demonstrated that both odiO and oiO are weaker than VBB and, despite this, these new notions are enough to implement several of the most important applications of VBB (Sect. 5). At this point, the natural question is how weak odiO and oiO are, compared to VBB. In order to give an answer to this question, we investigate whether the impossibility results for VBB (of Barak et al. [7,8]) extend to either odiO or oiO (or both). Unfortunately, this turned out to be true: As we show in Sect. 6.1, for type $\in$ {odiO, oiO}, there exist a type-sampler that cannot be type-obfuscated (unconditionally).

In addition, Barak et al. [8, Theorem 4.10] have shown that converting an *arbitrary* SKE into a PKE (by simply obfuscating the SKE's encryption algorithm together with a symmetric key) is not possible: Indeed, there exists a contrived SKE Π that cannot be obfuscated (as described above) into a PKE. *However, such an impossibility result does not apply to our* oiO*-based transformation from semantically secure and sel-IND-CPRA-key secure SKEs into sel-IND-CPA PKEs (Sect. 5.2) since the contrived SKE* Π *of* [8] *is not sel-IND-CPRA-key.* Following the same spirit, we study whether a similar argument applies to our format-preserving (deferred to full version) and function-preserving transformations (Construction 2). In this case, we have a negative answer but only for the oiO-based function-preserving transformation (Construction 2): We demonstrate that there exists a SKE Π that is semantically and sel-IND-CPRA-key secure that cannot be converted into a sel-IND-CPA PKE by simply obfuscating the SKE's encryption algorithm together with a symmetric key, as done by our oiO-based Construction 2. On the other hand, it remains unclear how we can prove a similar impossibility result for our odiO-based format-preserving transformation from SKEs to PKEs (through puncturable PRFs). See full version of this work for more details.

We stress that both our impossibility results leverage similar techniques to that of Barak et al. [7,8] that we describe in the next sections.

Also, to meet space constraints, the formal proofs of the theorems that appear in next sections are deferred to full version.

6.1 Unobfuscatable odiO-samplers (Resp. oiO-samplers) Exist Unconditionally

We build an ensemble of circuits $\mathcal{C} = \{C^*_{\mathsf{s},(\mathsf{k},a,b,\mathsf{y},e)}\}$ (indexed by $(\mathsf{s},\mathsf{k},a,b,\mathsf{y},e) \in \{0,1\}^{5\lambda+1}$) that (i) $C^*_{\mathsf{s},(\mathsf{k},a,b,\mathsf{y},e)}$ leaks no information when treated as oracles, and (ii) the obfuscation of any $C^*_{\mathsf{s},(\mathsf{k},a,b,\mathsf{y},e)} \in \mathcal{C}$ allows to extract the hard-coded values $(\mathsf{k},a,b,\mathsf{y},e)$. We anticipate that the value $e \in \{0,1\}$ will allow us to prove that a circuit $C^*_{\mathsf{s},(\mathsf{k},a,b,\mathsf{y},e)}$ cannot be odiO-obfuscated (resp. oiO-obfuscated) (see Sect. 6.1). On the other hand, the value y is a key of a PRF that is fundamental to build a contrived semantically and sel-IND-CPRA-key secure SKE that cannot be obfuscated (as described in Construction 2) into a sel-IND-CPA PKE (Sect. 6.2). We build such an ensemble $\mathcal{C}$ (depicted in Fig. 4) by using a similar technique to that of [7,8] (for more details, we refer the reader to [7,8]).

In a nutshell, $C^*_{\mathsf{s},(\mathsf{k},a,b,\mathsf{y},e)}$ (depicted in Fig. 4) is the composition of four circuits $(C^0_{\mathsf{k},a,b}, C^1_{\mathsf{k},a}, C^2_{\mathsf{k}}, C^3_{\mathsf{k},a,b,y,e})$ and it is defined with respect to a SKE scheme $\Pi_0 = (\mathsf{KGen}_0, \mathsf{Enc}_0, \mathsf{Dec}_0)$ and a PRF $\Pi_1 = (\mathsf{Gen}_1, \mathsf{F}_1)$ (required to generate "fresh" randomnesses). On input (ℓ, v, r) where $v = (x, i, c_1, c_2, \odot, d_1, \ldots, d_\lambda)$, $C^*_{\mathsf{s},(\mathsf{k},a,b,\mathsf{y},e)}$ uses ℓ to select which circuit to execute:

1. If $\ell = 0$, $C^0_{\mathsf{k},a,b}(x, \mathsf{F}_1(\mathsf{s}, (\ell, v, r)))$ is executed. This circuit presents a trigger input a. If $x = a$, $C^0_{\mathsf{k},a,b}(x, \mathsf{F}_1(\mathsf{s}, (\ell, v, r)))$ returns b. Otherwise, it returns $\mathsf{Enc}_0(\mathsf{k}, 0; \mathsf{F}_1(\mathsf{s}, (\ell, v, r)))$.
2. If $\ell = 1$, $C^1_{\mathsf{k},a}(i, \mathsf{F}_1(\mathsf{s}, (\ell, v, r)))$ is executed. This circuit simply outputs the encryption of the i-th bit of a, i.e., $\mathsf{Enc}_0(\mathsf{k}, a_i; \mathsf{F}_1(\mathsf{s}, (\ell, v, r)))$.
3. If $\ell = 2$, $C^2_{\mathsf{k}}(c_1, c_2, \odot, \mathsf{F}_1(\mathsf{s}, (\ell, v, r)))$ is executed. This circuit allows an evaluator to perform (gate by gate) computations over encrypted inputs. In more detail, it outputs the encryption of the evaluation of $w \odot z$ (i.e., $\mathsf{Enc}_0(\mathsf{k}, w \odot z; \mathsf{F}_1(\mathsf{s}, (\ell, v, r))))$ where $\odot$ is a binary operator, and w and z are the bits encrypted by c_1 and c_2, respectively.
4. If $\ell = 3$, $C^3_{\mathsf{k},a,b,\mathsf{y},e}(d_1, \ldots, d_\lambda, \mathsf{F}_1(\mathsf{s}, (\ell, v, r)))$ is executed. This is another circuit that presents a trigger input b. In more detail, if each d_i is the encryption of the i-th of b, the circuit returns $(\mathsf{k}, a, \mathsf{y}, e)$. Otherwise, it returns $\mathsf{Enc}_0(\mathsf{k}, 0; \mathsf{F}_1(\mathsf{s}, (\ell, v, r)))$.

Following [7,8], if the SKE scheme Π_0 is IND-CCA1 and Π_1 is a secure PRF, then oracle access to $C^*_{\mathsf{s},(\mathsf{k},a,b,\mathsf{y},e)}$ is computationally indistinguishable to oracle access to a circuit $\widetilde{C}_{\mathsf{k}}$ that, on every input (ℓ, v, r), it always outputs a fresh encryption of 0. This is because an adversary only sees ciphertexts and, as a consequence, it cannot distinguish between $C^*_{\mathsf{s},(\mathsf{k},a,b,\mathsf{y},e)}$ and $\widetilde{C}_{\mathsf{k}}$ unless it guesses the trigger inputs $a, b \in \{0,1\}^\lambda$. As a consequence, this implies that (i) an adversary cannot leak the hardcoded values $(\mathsf{k}, a, b, \mathsf{y}, e)$ and, (ii) the pair of circuits $(C^*_{\mathsf{s},(\mathsf{k},a,b,\mathsf{y},0)}, C^*_{\mathsf{s},(\mathsf{k},a,b,\mathsf{y},1)})$ are both oracle-differing-input and oracle-indistinguishable circuits (Definition 4.1).

On the other hand, on input $\widetilde{C} \leftarrow\!\!\$\ \mathsf{Obf}(1^\lambda, C^*_{\mathsf{s},(\mathsf{k},a,b,\mathsf{y},e)})$, an adversary can easily extract $(\mathsf{k}, a, b, \mathsf{y}, e)$, i.e., the circuit is partially reversible. This can be done as follows:

– Evaluate $\widetilde{C}(1,\cdot,\cdot)$ to get the encryptions $(c_1, \ldots, c_\lambda)$ of the a's bits (see Item 2).
– Use $(c_1, \ldots, c_\lambda)$ to compute $(d_1, \ldots, d_\lambda)$ where d_i is the encryption of b's i-th bit. Observe that this can be done by leveraging $\widetilde{C}(2,\cdot,\cdot)$ to evaluate (gate by gate) $\widetilde{C}(0,\cdot,\cdot) = C^0_{\mathsf{k},a,b}(\cdot,\cdot)$ on a (see Item 3), and
– Compute $(\mathsf{k}, a, b, \mathsf{y}, e)$ by $\widetilde{C}(3,\cdot,\cdot)$ on $(d_1, \ldots, d_\lambda)$ (see Item 4).

The properties of the ensemble $\mathcal{C}$ are formalized in Theorem 6.1. We highlight that our technique of generating Enc_0's randomness as $\mathsf{F}_1(\mathsf{s}, (\ell, v, r))$ (instead of $\mathsf{F}_1(\mathsf{s}, (\ell, v))$ as done by Barak et al. [7,8]) permits to have multiple randomnesses for a fixed pair (ℓ, v). This is allows us to prove a new property (not achieved by [7,8]) named *input-indistinguishability* that, in turn, is fundamental to prove the impossibility (Sect. 6.2) of converting semantically and sel-IND-CPRA-key secure SKE into sel-IND-CPA PKE. We stress that the ensemble of circuits built by Barak et al. [7, Lemma 3.5] does not satisfy input-indistinguishability.

Theorem 6.1. *Let $\Pi_0 = (\mathsf{KGen}_0, \mathsf{Enc}_0, \mathsf{Dec}_0)$, $\Pi_1 = (\mathsf{Gen}_1, \mathsf{F}_1)$, and $C^*_{\mathsf{s},(\mathsf{k},a,b,\mathsf{y},e)}$ be a SKE scheme with key space $\{0,1\}^\lambda$, a PRF scheme with key space $\{0,1\}^\lambda$, and the circuit defined in Fig. 4 with respect to Π_0 and Π_1, respectively. Then, the ensemble $\mathcal{C} = \{C^*_{\mathsf{s},(\mathsf{k},a,b,\mathsf{y},e)}\}_{\mathsf{s},\mathsf{k},a,b,\mathsf{y} \in \{0,1\}^\lambda, e \in \{0,1\}}$ satisfies the following properties:*

Oracle-differing-input: *If Π_0 is IND-CCA1 and Π_1 is secure then for every PPT adversary D, we have*

$$\mathbb{P}\left[C^*_{\mathsf{s},(\mathsf{k},a,b,\mathsf{y},0)}(\ell, v, r) \neq C^*_{\mathsf{s},(\mathsf{k},a,b,\mathsf{y},1)}(\ell, v, r)\right] \leq \mathsf{negl}(\lambda),$$

where $(\ell, v, r) \leftarrow\!\!\$\ \mathsf{A}^{C^*_{\mathsf{s},(\mathsf{k},a,b,\mathsf{y},0)}(\cdot,\cdot,\cdot), C^*_{\mathsf{s},(\mathsf{k},a,b,\mathsf{y},1)}(\cdot,\cdot,\cdot)}(1^\lambda)$, $\mathsf{k} \leftarrow\!\!\$\ \mathsf{KGen}_0(1^\lambda)$, $\mathsf{s} \leftarrow\!\!\$\ \mathsf{Gen}_1(1^\lambda)$, $\mathsf{y} \leftarrow\!\!\$\ \mathsf{Gen}_1(1^\lambda)$, and $(a, b) \leftarrow\!\!\$\ \{0,1\}^{2\lambda}$.

Input-indistinguishability: *If Π_0 is IND-CCA1 and IND-CPA-key, and Π_1 is secure, then for every $\ell, v \in \{0,1\}^*$, every PPT adversary D, we have*

$$\left|\mathbb{P}\left[\mathsf{D}^{C^*_{\mathsf{s}_0(\mathsf{k}_0,a_0,b_0,\mathsf{y}_0,0)}(\cdot,\cdot,\cdot), C^*_{\mathsf{s}_1,(\mathsf{k}_1,a_1,b_1,\mathsf{y}_1,1)}(\cdot,\cdot,\cdot)}(1^\lambda, m_0) = 1\right] - \right.$$
$$\left.\mathbb{P}\left[\mathsf{D}^{C^*_{\mathsf{s}_0,(\mathsf{k}_0,a_0,b_0,\mathsf{y}_0,0)}(\cdot,\cdot,\cdot), C^*_{\mathsf{s}_1,(\mathsf{k}_1,a_1,b_1,\mathsf{y}_1,1)}(\cdot,\cdot,\cdot)}(1^\lambda, m_1) = 1\right]\right| \leq \mathsf{negl}(\lambda),$$

where $(a_0, b_0, a_1, b_1) \leftarrow\!\!\$\ \{0,1\}^{4\lambda}$, $\mathsf{k}_j \leftarrow\!\!\$\ \mathsf{KGen}_0(1^\lambda)$ for $j \in \{0,1\}$, $\mathsf{s}_j \leftarrow\!\!\$\ \mathsf{Gen}_1(1^\lambda)$ for $j \in \{0,1\}$, $\mathsf{y}_j \leftarrow\!\!\$\ \mathsf{Gen}_1(1^\lambda)$ for $j \in \{0,1\}$, and $m_d = C^*_{\mathsf{s}_d,(\mathsf{k}_d,a_d,b_d,\mathsf{y}_d,d)}(\ell, v, r_d)$ for $r_d \leftarrow\!\!\$\ \{0,1\}^*$ and $d \in \{0,1\}$.

```
$C^{\mathsf{owf}}_{r,b}(x)$
If $x = r$, return $b$
return $0$
```

```
$\mathsf{S}_{\mathsf{owf}}(1^\lambda; r)$
Set $C_0 = C^{\mathsf{owf}}_{r,0}, C_1 = C^{\mathsf{owf}}_{r,1}, \alpha = \bot$
return $(C_0, C_1, \alpha)$
```

Fig. 5. The circuit $C^{\mathsf{owf}}_{r,b}$ and the sampler $\mathsf{S}_{\mathsf{owf}}$.

Partial reversibility: *There exists a PPT algorithm* Ext *such that for every* $(\mathsf{s}, \mathsf{k}, a, b, \mathsf{y}, e) \in \{0,1\}^{5\lambda+1}$ *and every circuit* $\widetilde{C}$ *such that* $\widetilde{C}(\ell, v, r) = C^*_{\mathsf{s},(\mathsf{k},a,b,\mathsf{y},e)}(\ell, v, r)$ *for all* $\ell, v, r \in \{0,1\}^*$, $\mathbb{P}\left[(\mathsf{k}, a, b, \mathsf{y}, e) \leftarrow\!\!\$\ \mathsf{Ext}(1^\lambda, \widetilde{C})\right] = 1$.

Theorem 6.1 implies that there exists an odiO-sampler (resp. oiO-sampler) $\widehat{\mathsf{S}}$ that cannot be odiO-obfuscated (resp. oiO-obfuscated), if OWFs exist (indeed, OWF implies both IND-CCA1 and IND-CPA-key security of SKE. See full version for more details).

Corollary 6.2. *For* $\mathsf{type} \in \{\mathsf{odiO}, \mathsf{oiO}\}$*, if OWFs exist then there exists a* type*-sampler* $\widehat{\mathsf{S}}$ *(Definition 4.1) such that* $\widehat{\mathsf{S}} \notin \mathcal{S}_{\mathsf{type}}$ *where* $\mathcal{S}_{\mathsf{type}}$ *is defined in Definition 4.3.*

Similarly to VBB, both odiO and oiO imply the existence of OWFs. As a consequence, for $\mathsf{type} \in \{\mathsf{odiO}, \mathsf{odiO}\}$, a type-unobfuscatable type-sampler exists *unconditionally.*

Theorem 6.3. *s Let* Obf *and* $\mathsf{S}_{\mathsf{owf}}$ *be an obfuscator and the sampler as defined in Fig. 5. Let* $p(\cdot)$ *and* $\mathcal{F} = \{\mathsf{F}_\lambda\}_{\lambda \in \mathbb{N}}$ *be a polynomial and an ensemble of functions such that* F_λ *is defined as* $\mathsf{F}_\lambda(b, r_0, r_1) = \mathsf{Obf}(1^\lambda, C^{\mathsf{owf}}_{r_0,b}; r_1)$ *where* $(b, r_0, r_1) \in \{0,1\} \times \{0,1\}^\lambda \times \{0,1\}^{p(\lambda)}$*. Then, the following statements hold:*

1. $\mathsf{S}_{\mathsf{owf}}$ *is an* odiO*-sampler (resp.* oiO*-sampler), and*
2. *if* Obf *is a* $(\{\mathsf{S}_{\mathsf{owf}}\})$-odiO*-obfuscator (resp.* $(\{\mathsf{S}_{\mathsf{owf}}\})$-oiO*-obfuscator) then* $\mathsf{F}_\lambda \in \mathcal{F}$ *is a OWF.*

Corollary 6.4. *For* $\mathsf{type} \in \{\mathsf{odiO}, \mathsf{oiO}\}$*, there exists (unconditionally) a* type*-sampler* S *such that* $\mathsf{S} \notin \mathcal{S}_{\mathsf{type}}$ *where* $\mathcal{S}_{\mathsf{type}}$ *as defined in Definition 4.3.*

The above corollary follows by combining Corollary 6.2 and Theorem 6.3, i.e., either $\mathsf{S}_{\mathsf{owf}} \notin \mathcal{S}_{\mathsf{type}}$ or $\widehat{\mathsf{S}} \notin \mathcal{S}_{\mathsf{type}}$ (for $\mathsf{type} \in \{\mathsf{odiO}, \mathsf{odiO}\}$) where $\mathsf{S}_{\mathsf{owf}}$ and $\widehat{\mathsf{S}}$ defined in Fig. 5 and Corollary 6.2, respectively.

6.2 Impossibility of Obfuscating Semantically and sel-IND-CPRA-key Secure SKE into sel-IND-CPA Secure PKE Schemes

We now demonstrate that it is inherently impossible to convert a semantically secure and sel-IND-CPRA-key SKEs into sel-IND-CPA PKEs by simply obfuscating the SKE's encryption algorithm, as described in our oiO-based Construction 2. We prove this by leveraging a similar technique to that of [8]: We construct a SKE Π^* that satisfies semantic and sel-IND-CPRA-key security that, when obfuscated into a PKE (as described in Sect. 5.2), the latter results to be completely insecure. By leveraging the ensemble $\mathcal{C}$ of Theorem 6.1, a PRF $\overline{\Pi} = (\overline{\mathsf{Gen}}, \bar{\mathsf{F}})$, and a semantically and sel-IND-CPRA-key secure SKE scheme $\widetilde{\Pi} = (\widetilde{\mathsf{KGen}}, \widetilde{\mathsf{Enc}}, \widetilde{\mathsf{Dec}})$, we build the contrived SKE Π^* as follows:

$$\mathsf{Enc}^*(\mathsf{k}^*, (\ell, v); r) = (\widetilde{\mathsf{Enc}}(\widetilde{\mathsf{k}}, (\ell, v); r), C^*_{\mathsf{s},(\widehat{\mathsf{k}},a,b,\mathsf{y},e)}(\ell, v, r), \bar{\mathsf{F}}(\mathsf{y}, (\ell, v, r)) \oplus \widetilde{\mathsf{k}}) \quad (1)$$

where $\mathsf{k}^* = (\widehat{\mathsf{k}}, \widetilde{\mathsf{k}}, \mathsf{s}, a, b, \mathsf{y}, e)$.

Π^* is a semantically and sel-IND-CPRA-key secure SKE for the following reasons. First, as described in Sect. 6.1, oracle access to the circuit $C^*_{\mathsf{s},(\widehat{k},a,b,\mathsf{y},e)} \in \mathcal{C}$ is computationally indistinguishable from having oracle access to a circuit $\widetilde{C}_{\mathsf{k}}$ that always returns encryptions of 0. Hence, this implies that $C^*_{\mathsf{s},(\widehat{k},a,b,\mathsf{y},e)}$ does not leak the message (ℓ, v) and that an adversary cannot leak any information about $(\widehat{\mathsf{k}}, a, b, \mathsf{y}, e)$. Second, conditioned to the above observation, the semantic security of Π^* easily follows from the semantic security of $\widetilde{\Pi}$ and the security of $\overline{\Pi}$. Third, as for the sel-IND-CPRA-key security of Π^*, it follows from sel-IND-CPRA-key security of $\widetilde{\Pi}$, the security of $\overline{\Pi}$, and the fact that $\mathcal{C}$ satisfies input-indistinguishability (see Theorem 6.1).

On the other hand, when Enc^* is obfuscated (as in Construction 2), an adversary can exploit the partial reversibility of $\mathcal{C}$ (Theorem 6.1) to extract y and, in turn, the key $\widetilde{\mathsf{k}}$ that is used to encrypt the message $m = (\ell, v)$. Below, we report the formal result.

Theorem 6.5. *If OWFs exist then the following statements hold:*

1. *there exist a SKE Π^* such that Π^* is semantically secure, sel-IND-CPRA-key, and*
2. *the PKE scheme $\Pi = (\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ (output by applying to Π^* the transformation defined in Construction 2) is not sel-IND-CPA (Theorem 5.4).*

We stress that the above result improves the impossibility result of Barak et al. [8] since ours apply to the smaller class of SKEs (i.e., SKEs with stronger notions of security) that satisfy sel-IND-CPRA-key security.

Also, all our odiO-based transformations remain plausible as the impossibility result do not seem to extend. We provide more details in the full version of this work.

Acknowledgments. The authors would like to thank the anonymous reviewers for useful feedback. The research described in this paper received funding from: the Concordium Blockhain Research Center, Aarhus University, Denmark; the Carlsberg Foundation under the Semper Ardens Research Project CF18-112 (BCM); the European Research Council (ERC) under the European Unions's Horizon 2020 research and innovation programme under grant agreement No 803096 (SPEC).

References

1. Ananth, P., Boneh, D., Garg, S., Sahai, A., Zhandry, M.: Differing-inputs obfuscation and applications. IACR Cryptol. ePrint Arch. **2013**, 689 (2013)

2. Ananth, P., Jain, A., Lin, H., Matt, C., Sahai, A.: Indistinguishability obfuscation without multilinear maps: new paradigms via low degree weak pseudorandomness and security amplification. In: Boldyreva, A., Micciancio, D. (eds.) CRYPTO 2019. LNCS, vol. 11694, pp. 284–332. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-26954-8_10
3. Ananth, P., Jain, A.: Indistinguishability obfuscation from compact functional encryption. In: Gennaro, R., Robshaw, M. (eds.) CRYPTO 2015. LNCS, vol. 9215, pp. 308–326. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-47989-6_15
4. Ananth, P., Jain, A., Sahai, A.: Indistinguishability obfuscation from functional encryption for simple functions. Cryptology ePrint Archive (2015)
5. Barak, B., Bitansky, N., Canetti, R., Kalai, Y.T., Paneth, O., Sahai, A.: Obfuscation for evasive functions. In: Lindell, Y. (ed.) TCC 2014. LNCS, vol. 8349, pp. 26–51. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-642-54242-8_2
6. Barak, B., Garg, S., Kalai, Y.T., Paneth, O., Sahai, A.: Protecting obfuscation against algebraic attacks. In: Nguyen, P.Q., Oswald, E. (eds.) EUROCRYPT 2014. LNCS, vol. 8441, pp. 221–238. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-642-55220-5_13
7. Barak, B., et al.: On the (Im)Possibility of obfuscating programs. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 1–18. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-44647-8_1
8. Barak, B., et al.: On the (im) possibility of obfuscating programs. J. ACM (JACM) **59**(2), 1–48 (2012)
9. Bellare, M., Hoang, V.T., Keelveedhi, S.: Instantiating random oracles via UCEs. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013. LNCS, vol. 8043, pp. 398–415. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40084-1_23
10. Bellare, M., Stepanovs, I., Tessaro, S.: Poly-many hardcore bits for any one-way function and a framework for differing-inputs obfuscation. In: Sarkar, P., Iwata, T. (eds.) ASIACRYPT 2014. LNCS, vol. 8874, pp. 102–121. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-45608-8_6
11. Bellare, M., Stepanovs, I., Waters, B.: New negative results on differing-inputs obfuscation. In: Fischlin, M., Coron, J.-S. (eds.) EUROCRYPT 2016. LNCS, vol. 9666, pp. 792–821. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49896-5_28
12. Bitansky, N., Canetti, R.: On strong simulation and composable point obfuscation. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 520–537. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-14623-7_28
13. Bitansky, N., et al.: The impossibility of obfuscation with auxiliary input or a universal simulator. In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014. LNCS, vol. 8617, pp. 71–89. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-44381-1_5
14. Bitansky, N., Canetti, R., Kalai, Y.T., Paneth, O.: On virtual grey box obfuscation for general circuits. Algorithmica **79**(4), 1014–1051 (2017)
15. Bitansky, N., Vaikuntanathan, V.: Indistinguishability obfuscation from functional encryption. In: 2015 IEEE 56th Annual Symposium on Foundations of Computer Science (FOCS), pp. 171–190. IEEE Computer Society (2015)
16. Bitansky, N., Vaikuntanathan, V.: Indistinguishability obfuscation from functional encryption. J. ACM (JACM) **65**(6), 1–37 (2018)
17. Boyle, E., Chung, K.-M., Pass, R.: On extractability obfuscation. In: Lindell, Y. (ed.) TCC 2014. LNCS, vol. 8349, pp. 52–73. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-642-54242-8_3

18. Boyle, E., Pass, R.: Limits of extractability assumptions with distributional auxiliary input. In: Iwata, T., Cheon, J.H. (eds.) ASIACRYPT 2015. LNCS, vol. 9453, pp. 236–261. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-48800-3_10
19. Brakerski, Z., Döttling, N., Garg, S., Malavolta, G.: Candidate iO from homomorphic encryption schemes. In: Canteaut, A., Ishai, Y. (eds.) EUROCRYPT 2020. LNCS, vol. 12105, pp. 79–109. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-45721-1_4
20. Brakerski, Z., Rothblum, G.N.: Virtual black-box obfuscation for all circuits via generic graded encoding. In: Lindell, Y. (ed.) TCC 2014. LNCS, vol. 8349, pp. 1–25. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-642-54242-8_1
21. Campanelli, M., Francati, D., Orlandi, C.: Structure-preserving compilers from new notions of obfuscations. Cryptology ePrint Archive, Paper 2022/732 (2022). https://eprint.iacr.org/2022/732
22. Canetti, R., Kalai, Y.T., Paneth, O.: On obfuscation with random oracles. In: Dodis, Y., Nielsen, J.B. (eds.) TCC 2015. LNCS, vol. 9015, pp. 456–467. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46497-7_18
23. Canetti, R., Rothblum, G.N., Varia, M.: Obfuscation of hyperplane membership. In: Micciancio, D. (ed.) TCC 2010. LNCS, vol. 5978, pp. 72–89. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-11799-2_5
24. Chase, M., Lysyanskaya, A.: On signatures of knowledge. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 78–96. Springer, Heidelberg (2006). https://doi.org/10.1007/11818175_5
25. Diffie, W., Hellman, M.E.: New directions in cryptography. IEEE Trans. Inf. Theory **22**(6), 644–654 (1976)
26. Garg, S., Gentry, C., Halevi, S., Raykova, M., Sahai, A., Waters, B.: Candidate indistinguishability obfuscation and functional encryption for all circuits. In: 2013 IEEE 54th Annual Symposium on Foundations of Computer Science (FOCS), pp. 40–49. IEEE Computer Society (2013)
27. Garg, S., Gentry, C., Halevi, S., Wichs, D.: On the implausibility of differing-inputs obfuscation and extractable witness encryption with auxiliary input. Algorithmica **79**(4), 1353–1373 (2017)
28. Garg, S., Mahmoody, M., Mohammed, A.: When does functional encryption imply obfuscation? In: Kalai, Y., Reyzin, L. (eds.) TCC 2017. LNCS, vol. 10677, pp. 82–115. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-70500-2_4
29. Gay, R., Pass, R.: Indistinguishability obfuscation from circular security. In: Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing, pp. 736–749 (2021)
30. Goldwasser, S., Kalai, Y.T.: On the impossibility of obfuscation with auxiliary input. In: 46th Annual IEEE Symposium on Foundations of Computer Science (FOCS2005), pp. 553–562. IEEE (2005)
31. Goldwasser, S., Kalai, Y.T.: A note on the impossibility of obfuscation with auxiliary input. IACR Cryptol. ePrint Arch. **2013**, 665 (2013)
32. Goldwasser, S., Rothblum, G.N.: On best-possible obfuscation. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, pp. 194–213. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-70936-7_11
33. Goyal, R., Koppula, V., Waters, B.: Lockable obfuscation. In: 2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS), pp. 612–621. IEEE (2017)

34. Ishai, Y., Pandey, O., Sahai, A.: Public-coin differing-inputs obfuscation and its applications. In: Dodis, Y., Nielsen, J.B. (eds.) TCC 2015. LNCS, vol. 9015, pp. 668–697. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46497-7_26
35. Jain, A., Lin, H., Sahai, A.: Indistinguishability obfuscation from well-founded assumptions. In: Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing, pp. 60–73 (2021)
36. Lin, H., Pass, R., Seth, K., Telang, S.: Indistinguishability obfuscation with non-trivial efficiency. In: Cheng, C.-M., Chung, K.-M., Persiano, G., Yang, B.-Y. (eds.) PKC 2016. LNCS, vol. 9615, pp. 447–462. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49387-8_17
37. Lynn, B., Prabhakaran, M., Sahai, A.: Positive results and techniques for obfuscation. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 20–39. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-24676-3_2
38. Mahmoody, M., Mohammed, A., Nematihaji, S.: On the impossibility of virtual black-box obfuscation in idealized models. In: Kushilevitz, E., Malkin, T. (eds.) TCC 2016. LNCS, vol. 9562, pp. 18–48. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49096-9_2
39. Pass, R., Seth, K., Telang, S.: Indistinguishability obfuscation from semantically-secure multilinear encodings. In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014. LNCS, vol. 8616, pp. 500–517. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-44371-2_28
40. Pass, R., Shelat, A.: Impossibility of VBB obfuscation with ideal constant-degree graded encodings. In: Kushilevitz, E., Malkin, T. (eds.) TCC 2016. LNCS, vol. 9562, pp. 3–17. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49096-9_1
41. Sahai, A., Waters, B.: How to use indistinguishability obfuscation: deniable encryption, and more. In: Proceedings of the Forty-Sixth Annual ACM Symposium on Theory of Computing, pp. 475–484 (2014)
42. Shacham, H., Waters, B.: Compact proofs of retrievability. J. Cryptol. **26**(3), 442–483 (2013)
43. Wee, H.: On obfuscating point functions. In: Proceedings of the Thirty-seventh Annual ACM Symposium on Theory of Computing, pp. 523–532 (2005)
44. Wee, H., Wichs, D.: Candidate obfuscation via oblivious LWE sampling. In: Canteaut, A., Standaert, F.-X. (eds.) EUROCRYPT 2021. LNCS, vol. 12698, pp. 127–156. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-77883-5_5
45. Wichs, D., Zirdelis, G.: Obfuscating compute-and-compare programs under LWE. In: 2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS), pp. 600–611. IEEE (2017)