Structure-Preserving Compilers from New Notions of Obfuscations

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

[1] Protocol Labs, matteo@protocol.ai
[2] Aarhus University, dfrancati@cs.au.dk, orlandi@cs.au.dk

Abstract. The dream of software obfuscation is to take programs, *as they are*, and then 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 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).

Table of Contents

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 [BGI+12]. It follows that one important application of VBB is to *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 [DH76] 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 obfucators that allow *structure preserving* transformation of cryptographic primitives i.e., obfuscators that allow to compile cryptographic primitives while *preserving* parts of the original primitive. For instance, like in the Diffie and Hellman example, compile a SKE into a PKE in a *function-preserving* way (e.g., the decryption algorithm of the PKE is the same as the SKE), or at least in a *format-preserving* way (e.g., the PKE ciphertext is of the same format as the SKE one). We see this as an interesting design approach to transformation of primitives, worth of study of its own. *Structure-preserving* compilers—as we dub those preserving either function or format—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, a structure-preserving transformation maintains some of the scheme's original efficiency guarantees, either preserving the running time of the (possibly heavily optimized) original function or its communication complexity.

Nice transformations from *weaker* obfuscation? The seminal work in [BGI+01, BGI+12] has shown that the "dream version" of obfuscation, VBB, is in general impossible. 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 [SW14]. 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. In fact, most iO based constructions are quite involved and only carefully designed programs can be successfully obfuscated with iO. It is therefore natural to ask the following question:

Can we obtain 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 [BGI+12] only guarantees indistinguishability of obfuscations between pair of programs that have the exact same input/output behaviour. *Differing-input obfuscation* (diO) [BGI+12, ABG+13, BCP14] 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

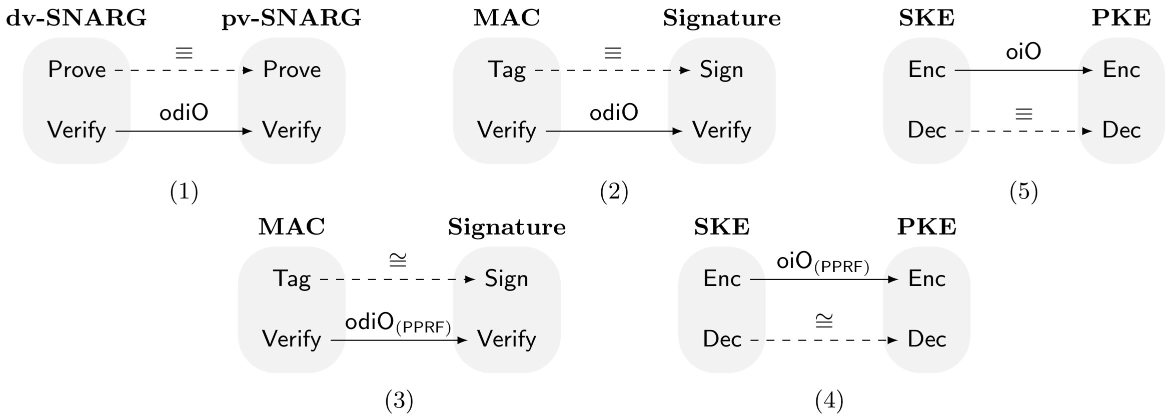

Fig. 1: The transformations (1)-(5) of this work: function-preserving on top row; format-preserving on bottom row. By odiO/oiO we denote an algorithm obtained through direct obfuscation of the one on the left; by $\equiv$ one that is completely unchanged; by $\cong$ one with minor changes but still able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

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 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. Since we do not have any candidate obfuscator for diO, we are then unable to provide any plausible candidate obfuscator for odiO and oiO (as for diO, one might use current candidates of iO obfuscator and "hope for the best"). 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).

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

1. A *function-preserving* transformation from selectively sound succinct designated verifier non-interactive argument systems (dv-SNARG) into publicly verifiable ones (pv-SNARG) (Section 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;[3]
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 (Section 5.2);
3. A *format-preserving* transformation that leverages puncturable PRFs to convert selectively existentially unforgeable MACs into selectively existentially unforgeable digital signatures (Section 5.3). In

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

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 (Section 5.4). 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 (Section 5.5). 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.

We highlight 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, note that all the transformations that use puncturable PRFs are (only) *format-preserving*.[4]

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. [BGI+01, BGI+12] extends to either odiO or oiO (or both). We provide two different negative results by adapting the techniques of [BGI+01, BGI+12] 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 (Section 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). We elaborate further on this later on.

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 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 [CL06] 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

[4] 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.

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 $\mathsf{Verify}(\mathsf{sk}, \dots)$;[5] 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:

- 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 $\hat{x}$). 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.[6] 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 $\mathsf{dv\text{-}NIZK}$ and $\mathsf{pv\text{-}NIZK}$. In contrast to soundness—which achieves only selective security—here we are able to preserve adaptive security. Again, the transfomation 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 Section 5.4, we show how to compile IV-based

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

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

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.[7] 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 intuitively 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).

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 [DH76]: 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

[7] 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.

would eventually need to reduce the security of our transformations (pv-SNARG, signature, PKE) to the security of the original secret-key primitive (dv-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 dv-SNARG into a pv-SNARG by publishing an obfuscation of the circuit C_0 which implements the dv-SNARG verification algorithm using an hardcoded 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 odiO/oiO. The seminal work of [BGI+01, BGI+12] 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, [BGI+01, BGI+12] 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 [BGI+01, BGI+12] can be extended to the setting of odiO and oiO. In particular, we show that there (unconditionally) exist samplers that are odiO/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 odiO/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 [BGI+12, 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 [BGI+12] 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 Section 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 [BGI+12] 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 [SW13]?

2 Related Work

Barak et al. [BGI+01, BGI+12] 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 [CRV10, WZ17, GKW17, Wee05] . Goldwasser and Kalai [GK05, GK13] and Bitansky et al. [BCC+14] 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, [BCC+14] demonstrated that the availability of auxiliary information is equivalent to VBB with universal simulation. Goldwasser and Rothblum [GR07] 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 [MMN16, LPS04, BGK+14, BR14, CKP15, PS16] studied the (in)feasibility of VBB in different idealized models.

To avoid the VBB paradigm (and its impossibility results), [BGI+12] 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 [SW14]. The first work that proposed a candidate iO construction is by Garg et al. [GGH+13] that built iO via multilinear maps. Subsequent works [GMM17, BV18, PST14, AJS15, BV15, AJ15, LPST16, AJL+19, BDGM20, GP21] 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. [JLS21] and Wee and Wichs [WW21]. [JLS21] 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, [WW21] proposed the first construction based solely on lattices and LWE. Their construction relies on a new falsifiable LWE assumption.

As for diO, [ABG+13, BCP14, BCP14, BST14] proposed different formalization of diO (for both circuits and Turing machines) and showed different applications. On the negative side, [BP15, BSW16, GGHW17] showed that, in the presence of (some) auxiliary information (e.g., samplers), a general diO obfuscator may not exist. Notably, Bellare et al. [BSW16] 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. [GGHW17] 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 [BP15, BSW16, GGHW17] 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. [IPS15] 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 include virtual gray-box obfuscation (VGB) [BC10, BCKP17]. 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 [BBC+14]. 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. To make the paper self-contained, our notation and all the standard definitions used in the paper can be found in Appendix A.

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) [BGI+01] 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) [BGI+12, ABG+13, BCP14] 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*

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

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*

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

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 [BSW16, GGHW17] (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. [BSW16] 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 [BSW16], if SNARKs exist the negative result can be extended to diO for circuits. Garg et al. [GGHW17] 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 [BSW16, GGHW17] 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) [BGI+01], 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 [BGI+01]. 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* $|\mathsf{Obf}(1^\lambda, C)| \le p(|C|)$.

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*

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

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 [BCC+14, BGI+01, BGI+12]. Moreover, also several applications of VBB are impossible to achieve. As an example, Barak et al. [BGI+12, 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 [WZ17] (also known as lockable obfuscation [GKW17]) and point functions [Wee05].

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 $\mathsf{type} = \mathsf{odiO}$:

$$\mathbb{P}\Big[C_0(x) \neq C_1(x) \Big| x \leftarrow\!\!\$\ \mathsf{A}^{C_0(\cdot), C_1(\cdot)}(1^\lambda, 1^{|C_0|}, \alpha)\Big] \leq \mathsf{negl}(\lambda),$$

If $\mathsf{type} = \mathsf{oiO}$:

$$\Big|\mathbb{P}\Big[\mathsf{A}^{C_0(\cdot)}(1^\lambda, 1^{|C_0|}, \alpha) = 1\Big] - \mathbb{P}\Big[\mathsf{A}^{C_1(\cdot)}(1^\lambda, 1^{|C_1|}, \alpha) = 1\Big]\Big| \leq \mathsf{negl}(\lambda),$$

where $(C_0, C_1, \alpha) \leftarrow\!\!\$\ \mathsf{S}(1^\lambda)$.[8]

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 $|\mathsf{Obf}(1^\lambda, C)| \leq p(|C|)$.*

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

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

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.,*

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

A PPT algorithm Obf is a best-possible universal type-obfuscator *if Obf is a $(\mathcal{S}_{\mathsf{type}})$-$\mathsf{type}$-obfuscator (Definitions 4.2, 3.3 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.[9] 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.

---

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

[9] 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 [BGI+01, BCC+14] (and odiO, oiO as we will show in Section 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 **(oiO $\Rightarrow$ odiO $\Rightarrow$ diO).** *For* 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.*

Proof. (Case odiO $\Rightarrow$ diO*).* Consider a diO-sampler $\mathsf{S} \in \mathcal{S}_{\mathsf{diO}}$. By definition, we have that for every PPT adversary A, that takes as input (C_0, C_1, α) (output by S), it is infeasible for A to find a differing-input x such that $C_0(x) \neq C_1(x)$ (Definition 3.2). As a consequence, it is also hard for A to find such a differing-input x when A has only oracle access to C_0 and C_1. Hence, S is also an odiO-sampler (Definition 4.1). Moreover, by definition of $\mathcal{S}_{\mathsf{diO}}$, $\mathsf{Obf}_{\mathsf{diO}}$ is a $(\{\mathsf{S}\})$-diO-obfuscator. Since the indistinguishability property of odiO is identical to that of diO, it follows that $\mathsf{Obf}_{\mathsf{diO}}$ is also a $(\{\mathsf{S}\})$-odiO-obfuscator. As a consequence, we conclude that $\mathsf{S} \in \mathcal{S}_{\mathsf{odiO}}$.

(Case oiO $\Rightarrow$ odiO*).* For $\mathsf{S} \in \mathcal{S}_{\mathsf{odiO}}$, we have that for every PPT adversary A, that takes as input α, it is infeasible to find differing-input x if A has only oracle access to C_0 and C_1 (where (C_0, C_1, α) output by S). As a consequence, these circuits (except with negligible probability) are identical when treated as oracles. Hence, S is also an oiO-sampler. Moreover, both odiO and oiO have the same indistinguishability property. This implies that $\mathsf{Obf}_{\mathsf{odiO}}$ is also a $(\{\mathsf{S}\})$-oiO-obfuscator and, in turn, this implies that $\mathsf{S} \in \mathcal{S}_{\mathsf{oiO}}$. □

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 appears in Appendix B.1.

Theorem 4.6 **(VBB $\Rightarrow$ oiO and VBB $\Rightarrow$ 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 $\mathsf{S} \notin \mathcal{S}_{\mathsf{odiO}}$.[10] The same applies between odiO and oiO, and between oiO and VBB (with respect to samplers as defined in Theorem 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. [BGI+01, Lemma 3.5, Corollary 3.8] have demonstrated that there (unconditionally) exists a distribution of circuits that cannot be VBB-obfuscated (see also Section 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 $(C, \bot)$ where C comes from the distribution of [BGI+01, Lemma 3.5]. Second, we have that any sampler S_1, that outputs $(C_0, C_1, \bot)$ such that $C_0 = C_1$, is an odiO-sampler (resp. oiO-sampler) that can be easily odiO-obfuscated (resp. oiO-obfuscated).[11] By combining these two observations, we conclude that if S_1 outputs $(C_0, C_1, \bot)$ where $C_0 = C_1$ and $(C_0, \bot) \leftarrow\!\!\$\ \mathsf{S}_0(1^\lambda)$, 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 transformations achieve (and use) different flavors of security whose definitions can be found in Appendix A. 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 (Section 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 (Section 5.2).

Format-Preserving Signatures from MACs: odiO is able to compile EUF MAC into a sel-EUF-CMA digital signature scheme, using puncturable PRF (Section 5.3).

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 (Section 5.4).

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 (Section 5.5).

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

$C^{\mathsf{Verify}}_{\mathsf{vrs}}(x,\pi)$	$\mathsf{S}_x(1^\lambda;r)$
return $b = \mathsf{Verify}^*(\mathsf{vrs},x,\pi)$	$(\mathsf{crs},\mathsf{vrs}) = \mathsf{Setup}^*(1^\lambda;r)$ Set $C_0 = C^{\mathsf{Verify}}_{\mathsf{vrs}}, C_1 = C^{\mathsf{Verify}}_{\mathsf{vrs},x}, \alpha = \mathsf{crs}$ **return** (C_0, C_1, α)
$C^{\mathsf{Verify}}_{\mathsf{vrs},x^*}(x,\pi)$	$\mathsf{S}_{\mathsf{Ext}^*}(1^\lambda;r)$
If $x = x^*$, **return** 0 **return** $b = \mathsf{Verify}^*(\mathsf{vrs},x,\pi)$	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, α)
$C^{\mathsf{Verify}}_{\mathsf{vrs},\mathsf{td},r}(x,\pi)$	
$\omega = \mathsf{Ext}^*_1(1^\lambda,\mathsf{td},x,\pi;r)$ **If** $\mathsf{Verify}^*(\mathsf{vrs},x,\pi) = 1$ and $(x,\omega)\in\mathcal{R}$, **return** 1 **return** 0	

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. $\gamma = \max\{|C^{\mathsf{Verify}}_{\mathsf{vrs}}|, |C^{\mathsf{Verify}}_{\mathsf{vrs},\mathsf{td},r}|\}$).

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 Figure 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 whose proof appears in Appendix B.2.

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 Figure 2.*

1. *If Π^* satisfies selective soundness (Definition A.2) 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 (Definitions A.2 and A.4).*

We extend the above result to the case of straight-line knowledge soundness. The proof appears in Appendix B.3.

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

1. *If Π^* satisfies straight-line knowledge soundness (Definition A.3) then the sampler $\mathsf{S}_{\mathsf{Ext}^*}$ of Figure 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 (Definitions A.3 and A.4).*

---

[10] 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.

[11] 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.

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

More on our transformations for arguments. To the best of our knowledge our work is the first to explicitly study how obfuscation can be used to transform designated verifiability into public verifiability.[12] One interesting feature of our transformation is that it fully preserves both the *communication complexity* and *the prover complexity* of the original designated-verifier scheme. Moreover, in certain cases it also preserves the *asymptotic* running time of the verifier. For example, if the verifier of the original dv-NIZK runs in asymptotic time $O(\mathsf{polylog}\,(|w|)\,\mathsf{poly}(\lambda))$—where w denotes the witness—so will the verifier in the compiled publicly verifiable scheme.[13] We believe these results can be of interest in at least two ways. First, they can leverage the efficiency (in terms of prover and proof size) of available designated-verifier schemes for which we cannot find a more efficient publicly verifiable counterpart. Second, they may provide a theoretical connection between designated and publicly verifiable SNARGs. For example, if both odiO and dv-SNARGs were known to be plausibly obtainable from assumption X, our transformation would show that pv-SNARGs can also be obtained from assumption X. To the best of our knowledge, not much is know on a separation between the two (see [CK21, Section 1.2] for a discussion).

We observe that constructions of non-adaptive zero-knowledge pv-SNARGs were already known through iO from the work in [SW14]. We now compare our results. First, we stress that our goals are different: our main priority is to obtain a pv-SNARG through a structure-preserving transformation from a weaker primitive (a dv-SNARG). The approach in [SW14] is not structure-preserving since their goal is to construct a zero-knowledge proof system "from scratch" through iO. Our constructions also differ with respect to some efficiency metrics. The verifier in [SW14] runs in $O(\mathsf{poly}\,(|x|, \lambda))$ while ours can potentially have worse asymptotics; their proof size is always polynomial in the security parameter and independent of other parameters. On the other hand, their construction has large parameters—it includes an obfuscation of a program verifying the whole relation—while our transformation preserves the size of the public parameters in the original dv-SNARG, which may be small.

5.2 From (q)-sEUF-sel-CMA MACs to (q)-sEUF-sel-CMA digital signatures

Construction 2 *Let $\Pi^* = (\mathsf{KGen}^*, \mathsf{Tag}^*, \mathsf{Verify}^*)$ and Obf be a MAC with message space $\mathcal{M}$ and an obfuscator, respectively. We compile Π^* into a digital signature scheme $\Pi = (\mathsf{KGen}, \mathsf{Sign}, \mathsf{Verify})$ with message space $\mathcal{M}$ as follows:*

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

$\mathsf{Sign}(\mathsf{sk}, m)$**:** *On input the secret key $\mathsf{sk} = \mathsf{k}^*$ and a message $m \in \mathcal{M}$, the randomized signing algorithm outputs $\sigma \leftarrow\!\!\$\, \mathsf{Tag}^*(\mathsf{k}^*, m)$.*

$\mathsf{Verify}(\mathsf{pk}, m, \sigma)$**:** *On input the public key $\mathsf{pk} = \widetilde{C}$, a message $m \in \mathcal{M}$, and a signature σ, the verification algorithm returns $b = \widetilde{C}(m, \sigma)$.*

Below we establish the following result whose proof appears in Appendix B.4.

[12] The work in [CK21] studies how much we can push succinct designated verifiability in proof schemes to obtain succinct and (somewhat) publicly verifiable schemes albeit both within a setting and through primitives very different from ours (e.g., requiring a committee sharing a secret).

[13] This holds if the public input x is absent or of size polynomial in the security parameter (e.g., in the case of the opening of a Merkle tree with given root). In the more general case, the resulting verifier will run in time $O(\mathsf{polylog}\,(|w|)\,\mathsf{poly}(\lambda) + \mathsf{poly}(|x|))$.

$C_{\mathsf{k}}^{\mathsf{Verify}}(m,\sigma)$	$\mathsf{S}_{\mathcal{Y}}(1^\lambda; r)$
return $b = \mathsf{Verify}^*(\mathsf{k}, m, \sigma)$	Let $r = (r_0, \dots, r_q)$
	Let $\mathcal{Y} = \{m_1, \dots, m_q\}$
$C_{\mathcal{X}}^{\mathsf{Verify}}(m,\sigma)$	$\mathsf{k} = \mathsf{KGen}^*(1^\lambda, 1^q; r_0)$
If $(m,\sigma) \notin \mathcal{X}$, **return** 0	$\forall i \in [q],\ \sigma_i = \mathsf{Tag}^*(\mathsf{k}, m_i; r_i)$
return 1	Set $\mathcal{X} = \{(m_i, \sigma_i)\}_{i \in [q]}$
	Set $C_0 = C_{\mathsf{k}}^{\mathsf{Verify}}, C_1 = C_{\mathcal{X}}^{\mathsf{Verify}}, \alpha = (\sigma_1, \dots, \sigma_q)$
	return (C_0, C_1, α)

Fig. 3: The circuits $C_{\mathsf{k}}^{\mathsf{Verify}}$, $C_{\mathcal{X}}^{\mathsf{Verify}}$, and the sampler $\mathsf{S}_{\mathcal{Y}}$. $C_{\mathsf{k}}^{\mathsf{Verify}}$ and $C_{\mathcal{X}}^{\mathsf{Verify}}$ are padded to match the size $\gamma = \max\{|C_{\mathsf{k}}^{\mathsf{Verify}}|, |C_{\mathcal{X}}^{\mathsf{Verify}}|\}$.

$C_{\mathsf{s}}^{\mathsf{Verify}}(m,\sigma)$	$\mathsf{S}_m(1^\lambda; r)$
$\mathsf{k} = \mathsf{KGen}_0^*(1^\lambda; \mathsf{F}_1^*(\mathsf{s}, m))$	$\mathsf{s} = \mathsf{Gen}_1^*(1^\lambda; r)$
return $b = \mathsf{Verify}_0^*(\mathsf{k}, m, \sigma)$	$\mathsf{s}' = \mathsf{Punct}_1^*(\mathsf{s}, m)$
	Set $C_0 = C_{\mathsf{s}}^{\mathsf{Verify}}, C_1 = C_{\mathsf{s}',m}^{\mathsf{Verify}}, \alpha = \mathsf{s}'$
$C_{\mathsf{s},m^*}^{\mathsf{Verify}}(m,\sigma)$	**return** (C_0, C_1, α)
If $m = m^*$, **return** 0	
$\mathsf{k} = \mathsf{KGen}_0^*(1^\lambda; \mathsf{F}_1^*(\mathsf{s}, m))$	
return $b = \mathsf{Verify}_0^*(\mathsf{k}, m, \sigma)$	

Fig. 4: The circuits $C_{\mathsf{s}}^{\mathsf{Verify}}$, $C_{\mathsf{s},m^*}^{\mathsf{Verify}}$, and the sampler S_m. $C_{\mathsf{s}}^{\mathsf{Verify}}$ and $C_{\mathsf{s},m^*}^{\mathsf{Verify}}$ are padded to match the size $\gamma = \max\{|C_{\mathsf{s}}^{\mathsf{Verify}}|, |C_{\mathsf{s},m^*}^{\mathsf{Verify}}|, |C_{\mathsf{s}',m^*,\mathsf{k}^*}^{\mathsf{Verify}}|\}$ where $C_{\mathsf{s}',m^*,\mathsf{k}^*}^{\mathsf{Verify}}$ is defined in Appendix B.5.

Theorem 5.4. *Let Π^* and Obf as defined in Construction 2. For every qin$\mathbb{N}$, every $\mathcal{Y} \subseteq \mathcal{M}$ such that $|\mathcal{Y}| = q$, consider the sampler $\mathsf{S}_{\mathcal{Y}}$ depicted in Figure 3.*

1. *If Π^* is (q)-sEUF-sel-CMA (Definition A.10) then for every $\mathcal{Y} \subseteq \mathcal{M}$ such that $|\mathcal{Y}| = q$, $\mathsf{S}_{\mathcal{Y}}$ is an* odiO*-sampler (Definition 4.1), and*
2. *for every $q \in \mathbb{N}$, if Obf is a $(\{\mathsf{S}_{\mathcal{Y}}\}_{\mathcal{Y} \subset \mathcal{M} : |\mathcal{Y}| = q})$-*odiO*-obfuscator (Definition 4.2) then the signature scheme Π of Construction 2 is (q)-sEUF-sel-CMA (Definition A.13).*

5.3 From EUF MACs to sel-EUF-CMA digital signatures using puncturable PRFs

Construction 3 *Let $\Pi_0^* = (\mathsf{KGen}_0^*, \mathsf{Tag}_0^*, \mathsf{Verify}_0^*)$, $\Pi_1^* = (\mathsf{Gen}_1^*, \mathsf{F}_1^*, \mathsf{Punct}_1^*)$ and Obf be a MAC with message space $\mathcal{M}$, a puncturable PRF, and an obfuscator, respectively. We combine Π_0^* and Π_1^* into a digital signature scheme $\Pi = (\mathsf{KGen}, \mathsf{Sign}, \mathsf{Verify})$ with message space $\mathcal{M}$ as follows:*

$\mathsf{KGen}(1^\lambda)$**:** *On input the security parameter 1^λ, the key generation algorithm computes $\mathsf{s} \leftarrow\!\!\$\ \mathsf{Gen}_1^*(1^\lambda)$ and outputs $\mathsf{pk} = \widetilde{C}$ and $\mathsf{sk} = \mathsf{s}$ where $\widetilde{C} \leftarrow\!\!\$\ \mathsf{Obf}(1^\lambda, C_{\mathsf{s}}^{\mathsf{Verify}})$ and $C_{\mathsf{s}}^{\mathsf{Verify}}$ is depicted in Figure 4.*

$\mathsf{Sign}(\mathsf{sk}, m)$**:** *On input the secret key $\mathsf{sk} = \mathsf{s}$ and a message $m \in \mathcal{M}$, the randomized signing algorithm outputs $\sigma \leftarrow\!\!\$\ \mathsf{Tag}^*(\mathsf{k}, m)$ where $\mathsf{k} = \mathsf{KGen}(1^\lambda; \mathsf{F}_1^*(\mathsf{s}, m))$.*

$\mathsf{Verify}(\mathsf{pk}, m, \sigma)$**:** *On input the public key $\mathsf{pk} = \widetilde{C}$, a message $m \in \mathcal{M}$, and a signature σ, the verification algorithm returns $b = \widetilde{C}(m, \sigma)$.*

Below we establish the following result whose proof appears in Appendix B.5.

Theorem 5.5. *Let Π_0^*, Π_1^*, and Obf as defined in Construction 3. For every $m \in \mathcal{M}$, consider the sampler S_m depicted in Figure 4.*

$C^{\mathsf{Enc}}_{\mathsf{s}_1,\mathsf{s}_2}(m, r)$	$\mathsf{S}_m(1^\lambda; r)$
$\mathsf{iv} = \mathsf{F}^*_1(\mathsf{s}_1, r)$	Let $r = (r_0, r_1, r_2)$
$\mathsf{k} = \mathsf{KGen}^*_0(1^\lambda; \mathsf{F}^*_2(\mathsf{s}_2, \mathsf{iv}))$	$\mathsf{s}_1 = \mathsf{Gen}^*_1(1^\lambda; r_0),\ \mathsf{s}_2 = \mathsf{Gen}^*_2(1^\lambda; r_1)$
return $\mathsf{Enc}^*_0(\mathsf{k}, m; \mathsf{iv})$	$\mathsf{iv} = \mathsf{F}^*_1(\mathsf{s}_1, r_2)$
	$\mathsf{k} = \mathsf{KGen}^*_0(1^\lambda; \mathsf{F}^*_2(\mathsf{s}_2, \mathsf{iv}))$
$C^{\mathsf{Enc}}_{\mathsf{s}_1,\mathsf{s}_2,r^*}(m, r)$	$c = \mathsf{Enc}^*_0(\mathsf{k}, m; \mathsf{iv})$
If $r = r^*$, **return** 0	$\mathsf{s}'_1 = \mathsf{Punct}^*_1(\mathsf{s}_1, r_2),\ \mathsf{s}'_2 = \mathsf{Punct}^*_2(\mathsf{s}_2, \mathsf{iv})$
$\mathsf{iv} = \mathsf{F}^*_1(\mathsf{s}_1, r)$	Set $C_0 = C^{\mathsf{Enc}}_{\mathsf{s}_1,\mathsf{s}_2}, C_1 = C^{\mathsf{Enc}}_{\mathsf{s}'_1,\mathsf{s}'_2,r_2}, \alpha = c$
$\mathsf{k} = \mathsf{KGen}^*_0(1^\lambda; \mathsf{F}^*_2(\mathsf{s}_2, \mathsf{iv}))$	**return** (C_0, C_1, α)
return $\mathsf{Enc}^*_0(\mathsf{k}, m; \mathsf{iv})$	

Fig. 5: The circuits $C^{\mathsf{Enc}}_{\mathsf{s}_1,\mathsf{s}_2}$, $C^{\mathsf{Enc}}_{\mathsf{s}_1,\mathsf{s}_2,r^*}$ and the sampler S_m. $C^{\mathsf{Enc}}_{\mathsf{s}_1,\mathsf{s}_2}$ and $C^{\mathsf{Enc}}_{\mathsf{s}_1,\mathsf{s}_2,r^*}$ (output by S_m) are padded to match the size $\gamma = \max\{|C^{\mathsf{Enc}}_{\mathsf{s}_1,\mathsf{s}_2}|, |C^{\mathsf{Enc}}_{\mathsf{s}'_1,\mathsf{s}'_2,r^*}|, |C^{\mathsf{Enc}}_{\mathsf{s}'_1,\mathsf{s}_2,r^*}|\}$ where $C^{\mathsf{Enc}}_{\mathsf{s}'_1,\mathsf{s}_2,r^*}$ is output by the sampler $\widetilde{\mathsf{S}}_m$ as defined in Appendix B.6.

1. *If Π^*_0 is EUF (Definition A.11) and Π^*_1 is secure (Definition A.8) then, for every $m \in \mathcal{M}$, S_m is an* odiO*-sampler (Definition 4.1), and*
2. *if* Obf *is a* $(\{\mathsf{S}_m\}_{m\in\mathcal{M}})$-odiO*-obfuscator (Definition 4.2) then the signature scheme Π of Construction 3 is sel-EUF-CMA (Definition A.14).*

5.4 From semantically secure IV-based SKEs to sel-IND-CPA PKEs using puncturable PRFs

Here, we compile IV-based SKEs into PKEs. IV-based SKEs (e.g., AES-CBC-mode) are symmetric key encryption schemes such that Enc outputs ciphertexts 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 the message.[14]

Construction 4 *Let $\Pi^*_0 = (\mathsf{KGen}^*_0, \mathsf{Enc}^*_0, \mathsf{Dec}^*_0)$, $\Pi^*_1 = (\mathsf{Gen}^*_1, \mathsf{F}^*_1, \mathsf{Punct}^*_1)$, $\Pi^*_2 = (\mathsf{Gen}^*_2, \mathsf{F}^*_2, \mathsf{Punct}^*_2)$, and* Obf *be an* IV*-based SKE with message space $\mathcal{M}$, two puncturable PRFs, and an obfuscator, respectively. We combine Π^*_0, Π^*_1, and Π^*_2 into a PKE scheme Π = (KGen, Enc, Dec) with message space $\mathcal{M}$ as follows:*

$\mathsf{KGen}(1^\lambda)$**:** *On input the security parameter 1^λ, the key generation algorithm computes $\mathsf{s}_1 \leftarrow_\$ \mathsf{Gen}^*_1(1^\lambda)$, $\mathsf{s}_2 \leftarrow_\$ \mathsf{Gen}^*_2(1^\lambda)$, and outputs $\mathsf{pk} = \widetilde{C}$ and $\mathsf{sk} = \mathsf{s}_2$ where $\widetilde{C} \leftarrow_\$ \mathsf{Obf}(1^\lambda, C^{\mathsf{Enc}}_{\mathsf{s}_1,\mathsf{s}_2})$ and $C^{\mathsf{Enc}}_{\mathsf{s}_1,\mathsf{s}_2}$ is depicted in Figure 5.*

$\mathsf{Enc}(\mathsf{pk}, m; r)$**:** *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 $(\mathsf{iv}, c) = \widetilde{C}(m, r)$.*

$\mathsf{Dec}(\mathsf{sk}, c)$**:** *On input the secret key $\mathsf{sk} = \mathsf{s}_2$ and a ciphertext (iv, c), the deterministic decryption algorithm returns $m = \mathsf{Dec}(\mathsf{k}, (\mathsf{iv}, c))$ where $\mathsf{k} = \mathsf{KGen}^*_0(1^\lambda; \mathsf{F}^*_2(\mathsf{s}_2, \mathsf{iv}))$.*

Below we establish the following result whose proof appears in Appendix B.6.

Theorem 5.6. *Let Π^*_0, Π^*_1, Π^*_2, and* Obf *as defined in Construction 4. For every $m \in \mathcal{M}$, consider the sampler S_m depicted in Figure 5.*

1. *If Π^*_0 is semantically secure (Definition A.18) and Π^*_1, Π^*_2 are secure (Definition A.8) then, for every $m \in \mathcal{M}$, S_m is an* odiO*-sampler (Definition 4.1), and*
2. *if* Obf *is a* $(\{\mathsf{S}_m\}_{m\in\mathcal{M}})$-odiO*-obfuscator (Definition 4.2) then the PKE scheme Π of Construction 4 is sel-IND-CPA (Definition A.22).*

$C^{\mathsf{Enc}}_{\mathsf{k}}(m,r)$	$\mathsf{S}_m(1^\lambda;r)$
return $c = \mathsf{Enc}^*(\mathsf{k},m;r)$	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,α)

Fig. 6: 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 $\gamma=\max\{|C^{\mathsf{Enc}}_{\mathsf{k}_0}|,|C^{\mathsf{Enc}}_{\mathsf{k}_1}|\}$

5.5 From semantically and sel-IND-CPRA-key SKEs to sel-IND-CPA PKEs

Construction 5 *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 Figure 6.*

$\mathsf{Enc}(\mathsf{pk},m;r)$**:** *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 whose proof appears in Appendix B.7.

Theorem 5.7. *Let Π^* and Obf as defined in Construction 5. For every $m\in\mathcal{M}$, consider the sampler S_m depicted in Figure 6.*

1. *If Π^* is sel-IND-CPRA-key (Definition A.19) then, for every $m\in\mathcal{M}$, S_m is an oiO-sampler (Definition 4.1), and*
2. *If Π^* is semantically secure (Definition A.16) and Obf is a $(\{\mathsf{S}_m\}_{m\in\mathcal{M}})$-$\mathsf{oiO}$-obfuscator (Definition 4.2) then the PKE scheme Π of Construction 5 is sel-IND-CPA (Definition A.22).*

6 Extending the impossibility results of Barak et al. [BGI+01, BGI+12] to the setting of odiO and oiO

In Section 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 (Section 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. [BGI+01, BGI+12]) extend to either odiO or oiO (or both). Unfortunately, this turned out to be true: As we show in Section 6.1, for $\mathsf{type}\in\{\mathsf{odiO},\mathsf{oiO}\}$, there exist a type-sampler that cannot be type-obfuscated (unconditionally).

In addition, Barak et al. [BGI+12, 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 (Section 5.5) since the contrived SKE Π of [BGI+12] is not sel-IND-CPRA-key.* Following the same spirit, we study whether a similar

[14] IV-based SKEs are related to nonce-based SKEs [Rog04]. The main difference is that in IV-based SKE the initialization vectors iv are random whereas in nonce-based SKE iv is replaced with a nonce that not necessarily needs to be randomly chosen (e.g., the nonce could be a counter).

$C^0_{\mathsf{k},a,b}(x, r)$

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. 7: 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).

argument applies to our format- and function-preserving transformations, described in Construction 4 and Construction 5. In this case, we have a negative answer but only for the oiO-based function-preserving transformation (Construction 5): 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 5. On the other hand, it remains unclear how we can prove a similar impossibility result for our odiO-based format-preserving transformation (Construction 4) from SKEs to PKEs (through puncturable PRFs). See Section 5.4 and Remark 6.6 for more details.

We stress that both our impossibility results leverage similar techniques to that of Barak et al. [BGI+01, BGI+12] that we describe in the next sections.

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 hardcoded 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 Section 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 5) into a sel-IND-CPA PKE (Section 6.2). We build such an ensemble $\mathcal{C}$ (depicted in Figure 7) by using a similar technique to that of [BGI+01, BGI+12] (for more details, we refer the reader to [BGI+01, BGI+12]).

In a nutshell, $C^*_{\mathsf{s},(\mathsf{k},a,b,\mathsf{y},e)}$ (depicted in Figure 7) 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 $v = (x, i, c_1, c_2, \odot, d_1, \dots, d_\lambda)$, $C^*_{\mathsf{s},(\mathsf{k},a,b,\mathsf{y},e)}$ 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 $\ell = 3$, $C^3_{\mathsf{k},a,b,\mathsf{y},e}(d_1, \dots, d_\lambda, \mathsf{F}_1(\mathsf{s}, (\ell, v, r)))$ 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 [BGI+12, BGI+01], 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 $\widetilde{C}_{\mathsf{k}}$ (see Figure 10 and proof of Theorem 6.1) 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 $(c_1, \dots, c_\lambda)$ of the a's bits (see Item 2).
- Use $(c_1, \dots, c_\lambda)$ to compute $(d_1, \dots, d_\lambda)$ 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 $(d_1, \dots, d_\lambda)$ (see Item 4).

The properties of the ensemble $\mathcal{C}$ are formalized in Theorem 6.1 whose proof appears in Appendix B.8. 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. [BGI+01, BGI+12]) permits to have multiple randomnesses for a fixed pair (ℓ, v). This is allows us to prove a new property (not achieved by [BGI+01, BGI+12]) named *input-indistinguishability* that, in turn, is fundamental to prove the impossibility (Section 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. [BGI+01, 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 Figure 7 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 (Definition A.17) and Π_1 is secure (Definition A.6) then for every PPT adversary D, we have*

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

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 (Definition A.17) and IND-CPA-key (Definition A.18), and Π_1 is secure (Definition A.6), then for every $\ell, v \in \{0,1\}^*$, every PPT adversary D, we have*

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

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 $d \in \{0,1\}$.

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 $\ell, v, r \in \{0,1\}^*$,*

$$\mathbb{P}\Big[(\mathsf{k}, a, b, \mathsf{y}, e) \leftarrow\$ \mathsf{Ext}(1^\lambda, \widetilde{C})\Big] = 1.$$

Theorem 6.1 and Corollary A.20 imply that there exists an odiO-sampler (resp. oiO-sampler) $\widehat{\mathsf{S}}$ that cannot be odiO-obfuscated (resp. oiO-obfuscated), if OWFs exist.

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

Fig. 8: The circuit $C^{\mathsf{owf}}_{r,b}$ and the sampler $\mathsf{S}_{\mathsf{owf}}$.

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.*

Proof. If a OWF exists then there exist a IND-CCA1 and IND-CPA-key SKE scheme Π_0 (Corollary A.20) and a secure PRF scheme $\Pi_1 = (\mathsf{Gen}_1, \mathsf{F}_1)$. Consider $C^*_{\mathsf{s},(\mathsf{k},a,b,\mathsf{y},e)}$ the circuit (depicted in Figure 7) defined with respect to Π_0 and Π_1. Let $\widehat{\mathsf{S}}$ be the sampler that, on input the security parameter 1^λ, it outputs $(C_0 = C^*_{\mathsf{s},(\mathsf{k},a,b,\mathsf{y},0)}, C_1 = C^*_{\mathsf{s},(\mathsf{k},a,b,\mathsf{y},1)}, \bot)$ where $(a, b) \leftarrow\!\!\$\ \{0,1\}^{2\lambda}$, $\mathsf{k} \leftarrow\!\!\$\ \mathsf{KGen}_0(1^\lambda)$, $\mathsf{s} \leftarrow\!\!\$\ \mathsf{Gen}_1(1^\lambda)$, and $\mathsf{y} \leftarrow\!\!\$\ \mathsf{Gen}_1(1^\lambda)$. Since Π_1 is a secure PRF scheme and Π_0 is IND-CCA1 and IND-CPA-key secure, then C_0 and C_1 (output by $\widehat{\mathsf{S}}$) satisfy the oracle-differing-input property of Theorem 6.1. This implies that $\widehat{\mathsf{S}}$ is both an odiO-sampler and oiO-sampler (recall that any odiO-sampler is also an oiO-sampler (Theorem 4.5)).

Moreover, the partial reversibility property of Theorem 6.1 implies that $\widehat{\mathsf{S}}$ cannot be odiO-obfuscated (resp. oiO-obfusated). This is because there always exists a distinguisher D, that on input an obfuscated circuit $\widetilde{C} \leftarrow\!\!\$\ \mathsf{Obf}(1^\lambda, C_d)$, executes $(\mathsf{k}, a, b, \mathsf{y}, e) \leftarrow\!\!\$\ \mathsf{Ext}(1^\lambda, \widetilde{C})$ and outputs e (observe that $e = d$). Hence, we conclude that $\widehat{\mathsf{S}} \notin \mathcal{S}_{\mathsf{odiO}}$ (resp. $\widehat{\mathsf{S}} \notin \mathcal{S}_{\mathsf{oiO}}$). □

Similarly to VBB, both odiO and oiO imply the existence of OWFs (Theorem 6.3, proof in Appendix B.9). 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 Figure 8. 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* $(\{\mathsf{S}_{\mathsf{owf}}\})$*-*$\mathsf{odiO}$*-obfuscator (resp.* $(\{\mathsf{S}_{\mathsf{owf}}\})$*-*$\mathsf{oiO}$*-obfuscator) then* $\mathsf{F}_\lambda \in \mathcal{F}$ *is a OWF (Definition A.5).*

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.*

Proof. By combining Corollary 6.2 and Theorem 6.3, we obtain that 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 Figure 8 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 5. We prove this by leveraging a similar technique to that of [BGI+12]: We construct a SKE Π^* that satisfies semantic and sel-IND-CPRA-key security that, when obfuscated into a PKE (as described in Section 5.5), the latter results to be completely insecure. By leveraging the ensemble $\mathcal{C}$ of Theorem 6.1, a PRF $\overline{\Pi} = (\overline{\mathsf{Gen}}, \overline{\mathsf{F}})$, 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^* (see Appendix B.10) which is defined as follows:

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

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:

1. As described in Section 6.1 (see also proof of Theorem 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 $\widetilde{C}_{\mathsf{k}}$ (see Figure 10) 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 $(\widehat{\mathsf{k}}, a, b, \mathsf{y}, e)$.
2. Conditioned to the above observation, the semantic security of Π^* easily follows from the semantic security of $\widetilde{\Pi}$ and the security of $\overline{\Pi}$.
3. 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 5), 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 whose proof appears in Appendix B.10.

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

1. *there exist a SKE Π^* such that Π^* is semantically secure (Definition A.16), sel-IND-CPRA-key (Definition A.19), and*
2. *the PKE scheme $\Pi = (\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ (output by applying to Π^* the transformation defined in Construction 5) is not sel-IND-CPA (Theorem 5.7).*

We stress that the above result improves the impossibility result of Barak et al. [BGI+12] since ours apply to the smaller class of SKEs (i.e., SKEs with stronger notions of security) that satisfy sel-IND-CPRA-key security (Definition A.19).

Remark 6.6 (On Construction 4). We highlight that the technique used to build the contrived SKE scheme Π^* of Equation (1) is not enough to contradict the security of our odiO-based construction from IV-based SKEs to PKEs (through puncturable PRFs). Indeed, consider the following contrived IV-based SKE Π^* built starting from an IV-based SKE $\widetilde{\Pi}$:

$$\mathsf{Enc}^*(\mathsf{k}^*, (\ell, v); \mathsf{iv}) = (\mathsf{iv}, (c', C^*_{\mathsf{s},(\widehat{\mathsf{k}},a,b,\mathsf{y},e)}(\ell, v, \mathsf{iv}), \bar{\mathsf{F}}(\mathsf{y}, (\ell, v, \mathsf{iv})) \oplus \widetilde{\mathsf{k}})), \tag{2}$$

where $\mathsf{k}^* = (\widehat{\mathsf{k}}, \widetilde{\mathsf{k}}, \mathsf{s}, a, b, \mathsf{y}, e)$ and $\widetilde{\mathsf{Enc}}(\widetilde{\mathsf{k}}, (\ell, v); \mathsf{iv}) = (\mathsf{iv}, c')$. The PKE scheme Π (output by the compilation of the contrived Π^* into a PKE Π as described in Construction 4) has public keys of the form $\mathsf{pk} = \widetilde{C}$ where the obfuscated circuit $\widetilde{C}$ internally generates a new symmetric encryption key for each randomness r (see Figure 5). Although, an adversary A can still exploit the partial reversibility property of $\mathcal{C}$ to leak the symmetric key $\widetilde{\mathsf{k}}$ (part of k^*, see Equation (2)) generated through a particular randomness r, A will not be able to leak the one used to encrypt the challenge ciphertext c^* (of the sel-IND-CPA experiment of the PKE Π). This is because c^* is computed using a randomly chosen randomness r^* (not revealed to A). Hence, in order to exploit the partial reversibility property to leak the symmetric key $\widetilde{\mathsf{k}}$ (used to encrypt part of challenge ciphertext c^*), A needs first to guess the randomness r^*. This happens with negligible probability.

Remark 6.7 (On MACs and non-interactive argument systems). It is worth mentioning that [BGI+12, Theorem 4.10] also shows an impossibility result for MACs, i.e., there exists a contrived MAC whose Tag algorithm cannot be obfuscated. This impossibility result does not apply to our two transformations (described in Sections 5.2 and 5.3) since odiO transforms a MAC scheme into a digital signature by obfuscating the MAC's verification algorithm Verify (see Construction 2 and Construction 3). Indeed, note that the contrived MAC scheme described in [BGI+12] heavily relies on the fact that Tag outputs long strings (in order to leak the MAC's symmetric key). However, Verify's output is a single bit and this is a main obstacle while trying to extend their result to our transformations. Intuitively because we would need to build a contrived Verify algorithm that leaks information when obfuscated. However, since the Verify's output is a single bit, this would break the unforgeability of the MAC. The same argument applies to non-interactive argument systems.

References

ABG$^+$13. Prabhanjan Ananth, Dan Boneh, Sanjam Garg, Amit Sahai, and Mark Zhandry. Differing-inputs obfuscation and applications. *IACR Cryptol. ePrint Arch.*, 2013:689, 2013.

AJ15. Prabhanjan Ananth and Abhishek Jain. Indistinguishability obfuscation from compact functional encryption. In *Annual Cryptology Conference*, pages 308–326. Springer, 2015.

AJL$^+$19. Prabhanjan Ananth, Aayush Jain, Huijia Lin, Christian Matt, and Amit Sahai. Indistinguishability obfuscation without multilinear maps: new paradigms via low degree weak pseudorandomness and security amplification. In *Annual International Cryptology Conference*, pages 284–332. Springer, 2019.

AJS15. Prabhanjan Ananth, Abhishek Jain, and Amit Sahai. Indistinguishability obfuscation from functional encryption for simple functions. *Cryptology ePrint Archive*, 2015.

BBC$^+$14. Boaz Barak, Nir Bitansky, Ran Canetti, Yael Tauman Kalai, Omer Paneth, and Amit Sahai. Obfuscation for evasive functions. In *Theory of Cryptography Conference*, pages 26–51. Springer, 2014.

BC10. Nir Bitansky and Ran Canetti. On strong simulation and composable point obfuscation. In *Annual Cryptology Conference*, pages 520–537. Springer, 2010.

BCC$^+$14. Nir Bitansky, Ran Canetti, Henry Cohn, Shafi Goldwasser, Yael Tauman Kalai, Omer Paneth, and Alon Rosen. The impossibility of obfuscation with auxiliary input or a universal simulator. In *Annual Cryptology Conference*, pages 71–89. Springer, 2014.

BCKP17. Nir Bitansky, Ran Canetti, Yael Tauman Kalai, and Omer Paneth. On virtual grey box obfuscation for general circuits. *Algorithmica*, 79(4):1014–1051, 2017.

BCP14. Elette Boyle, Kai-Min Chung, and Rafael Pass. On extractability obfuscation. In *Theory of cryptography conference*, pages 52–73. Springer, 2014.

BDGM20. Zvika Brakerski, Nico Döttling, Sanjam Garg, and Giulio Malavolta. Candidate io from homomorphic encryption schemes. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 79–109. Springer, 2020.

BGI$^+$01. Boaz Barak, Oded Goldreich, Rusell Impagliazzo, Steven Rudich, Amit Sahai, Salil Vadhan, and Ke Yang. On the (im) possibility of obfuscating programs. In *Annual international cryptology conference*, pages 1–18. Springer, 2001.

BGI$^+$12. Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil Vadhan, and Ke Yang. On the (im) possibility of obfuscating programs. *Journal of the ACM (JACM)*, 59(2):1–48, 2012.

BGK$^+$14. Boaz Barak, Sanjam Garg, Yael Tauman Kalai, Omer Paneth, and Amit Sahai. Protecting obfuscation against algebraic attacks. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 221–238. Springer, 2014.

BP15. Elette Boyle and Rafael Pass. Limits of extractability assumptions with distributional auxiliary input. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 236–261. Springer, 2015.

BR14. Zvika Brakerski and Guy N Rothblum. Virtual black-box obfuscation for all circuits via generic graded encoding. In *Theory of Cryptography Conference*, pages 1–25. Springer, 2014.

BST14. Mihir Bellare, Igors Stepanovs, and Stefano Tessaro. Poly-many hardcore bits for any one-way function and a framework for differing-inputs obfuscation. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 102–121. Springer, 2014.

BSW16. Mihir Bellare, Igors Stepanovs, and Brent Waters. New negative results on differing-inputs obfuscation. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 792–821. Springer, 2016.

BV15. Nir Bitansky and Vinod Vaikuntanathan. Indistinguishability obfuscation from functional encryption. In *2015 IEEE 56th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 171–190. IEEE Computer Society, 2015.

BV18. Nir Bitansky and Vinod Vaikuntanathan. Indistinguishability obfuscation from functional encryption. *Journal of the ACM (JACM)*, 65(6):1–37, 2018.

CFF$^+$21. Matteo Campanelli, Antonio Faonio, Dario Fiore, Anaïs Querol, and Hadrián Rodríguez. Lunar: a toolbox for more efficient universal and updatable zksnarks and commit-and-prove extensions. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 3–33. Springer, 2021.

CK21. Matteo Campanelli and Hamidreza Khoshakhlagh. Succinct publicly-certifiable proofs. In *International Conference on Cryptology in India*, pages 607–631. Springer, 2021.

CKP15. Ran Canetti, Yael Tauman Kalai, and Omer Paneth. On obfuscation with random oracles. In *Theory of Cryptography Conference*, pages 456–467. Springer, 2015.

CL06. Melissa Chase and Anna Lysyanskaya. On signatures of knowledge. In *Annual International Cryptology Conference*, pages 78–96. Springer, 2006.

CRV10. Ran Canetti, Guy N Rothblum, and Mayank Varia. Obfuscation of hyperplane membership. In *Theory of Cryptography Conference*, pages 72–89. Springer, 2010.

DH76. Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Trans. Inf. Theory*, 22(6):644–654, 1976.

Fis05. Marc Fischlin. Communication-efficient non-interactive proofs of knowledge with online extractors. In *Annual International Cryptology Conference*, pages 152–168. Springer, 2005.

GGH+13. Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *2013 IEEE 54th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 40–49. IEEE Computer Society, 2013.

GGHW17. Sanjam Garg, Craig Gentry, Shai Halevi, and Daniel Wichs. On the implausibility of differing-inputs obfuscation and extractable witness encryption with auxiliary input. *Algorithmica*, 79(4):1353–1373, 2017.

GK05. Shafi Goldwasser and Yael Tauman Kalai. On the impossibility of obfuscation with auxiliary input. In *46th Annual IEEE Symposium on Foundations of Computer Science (FOCS'05)*, pages 553–562. IEEE, 2005.

GK13. Shafi Goldwasser and Yael Tauman Kalai. A note on the impossibility of obfuscation with auxiliary input. *IACR Cryptol. ePrint Arch.*, 2013:665, 2013.

GKW17. Rishab Goyal, Venkata Koppula, and Brent Waters. Lockable obfuscation. In *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 612–621. IEEE, 2017.

GMM17. Sanjam Garg, Mohammad Mahmoody, and Ameer Mohammed. When does functional encryption imply obfuscation? In *Theory of Cryptography Conference*, pages 82–115. Springer, 2017.

GP21. Romain Gay and Rafael Pass. Indistinguishability obfuscation from circular security. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pages 736–749, 2021.

GR07. Shafi Goldwasser and Guy N Rothblum. On best-possible obfuscation. In *Theory of Cryptography Conference*, pages 194–213. Springer, 2007.

HKW15. Susan Hohenberger, Venkata Koppula, and Brent Waters. Adaptively secure puncturable pseudorandom functions in the standard model. In *International conference on the theory and application of cryptology and information security*, pages 79–102. Springer, 2015.

IPS15. Yuval Ishai, Omkant Pandey, and Amit Sahai. Public-coin differing-inputs obfuscation and its applications. In *Theory of Cryptography Conference*, pages 668–697. Springer, 2015.

JLS21. Aayush Jain, Huijia Lin, and Amit Sahai. Indistinguishability obfuscation from well-founded assumptions. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pages 60–73, 2021.

LPS04. Benjamin Lynn, Manoj Prabhakaran, and Amit Sahai. Positive results and techniques for obfuscation. In *International conference on the theory and applications of cryptographic techniques*, pages 20–39. Springer, 2004.

LPST16. Huijia Lin, Rafael Pass, Karn Seth, and Sidharth Telang. Indistinguishability obfuscation with non-trivial efficiency. In *Public-Key Cryptography–PKC 2016*, pages 447–462. Springer, 2016.

MMN16. Mohammad Mahmoody, Ameer Mohammed, and Soheil Nematihaji. On the impossibility of virtual black-box obfuscation in idealized models. In *Theory of Cryptography Conference*, pages 18–48. Springer, 2016.

PS16. Rafael Pass and Abhi Shelat. Impossibility of vbb obfuscation with ideal constant-degree graded encodings. In *Theory of Cryptography Conference*, pages 3–17. Springer, 2016.

PST14. Rafael Pass, Karn Seth, and Sidharth Telang. Indistinguishability obfuscation from semantically-secure multilinear encodings. In *Annual Cryptology Conference*, pages 500–517. Springer, 2014.

Rog04. Phillip Rogaway. Nonce-based symmetric encryption. In *International workshop on fast software encryption*, pages 348–358. Springer, 2004.

SW13. Hovav Shacham and Brent Waters. Compact proofs of retrievability. *Journal of cryptology*, 26(3):442–483, 2013.

SW14. Amit Sahai and Brent Waters. How to use indistinguishability obfuscation: deniable encryption, and more. In *Proceedings of the forty-sixth annual ACM symposium on Theory of computing*, pages 475–484, 2014.

Wee05. Hoeteck Wee. On obfuscating point functions. In *Proceedings of the thirty-seventh annual ACM symposium on Theory of computing*, pages 523–532, 2005.

WW21. Hoeteck Wee and Daniel Wichs. Candidate obfuscation via oblivious lwe sampling. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 127–156. Springer, 2021.

WZ17. Daniel Wichs and Giorgos Zirdelis. Obfuscating compute-and-compare programs under lwe. In *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 600–611. IEEE, 2017.

A Further Preliminaries

A.1 Notation

We use the notation $[n] \stackrel{\text{def}}{=} \{1, \ldots, n\}$. Capital boldface letters (such as $\mathbf{X}$) are used to denote random variables, small letters (such as x) to denote concrete values, calligraphic letters (such as $\mathcal{X}$) to denote sets, and serif letters (such as A) to denote algorithms. All of our algorithms are modeled as (possibly interactive) Turing machines. For a string $x \in \{0,1\}^*$, we let $|x|$ be its length; if $\mathcal{X}$ is a set, $|\mathcal{X}|$ represents the cardinality of $\mathcal{X}$. When x is chosen randomly in $\mathcal{X}$, we write $x \leftarrow\!\$\, \mathcal{X}$. If A is an algorithm, we write $y \leftarrow\!\$\, \mathsf{A}(x)$ to denote a run of A on input x and output y; if A is randomized, y is a random variable and $\mathsf{A}(x;r)$ denotes a run of A on input x and (uniform) randomness r. An algorithm A is *probabilistic polynomial-time* (PPT) if A is randomized and for any input $x, r \in \{0,1\}^*$ the computation of $\mathsf{A}(x;r)$ terminates in a polynomial number of steps (in the input size).

Negligible functions. We denote by $\lambda \in \mathbb{N}$ the security parameter and we implicitly assume that every algorithm takes as input the security parameter (written in unary). A function $\nu : \mathbb{N} \to [0,1]$ is called *negligible* in the security parameter λ if it vanishes faster than the inverse of any polynomial in λ, i.e. $\nu(\lambda) \in O(1/p(\lambda))$ for all positive polynomials $p(\lambda)$. We sometimes write $\mathsf{negl}(\lambda)$ (resp., $\mathsf{poly}(\lambda)$) to denote an unspecified negligible function (resp., polynomial function) in the security parameter.

Computational indistinguishability. We say that $\mathbf{X}$ and $\mathbf{Y}$ are *computationally* indistinguishable, denoted $\mathbf{X} \approx_c \mathbf{Y}$, if for all PPT distinguishers D we have that $\left|\mathbb{P}\left[\mathsf{D}(1^\lambda, \mathbf{X}) = 1\right] - \mathbb{P}\left[\mathsf{D}(1^\lambda, \mathbf{Y}) = 1\right]\right| \leq \mathsf{negl}(\lambda)$.

A.2 Non-Interactive Argument systems

Let $\mathcal{R}$ be a decidable binary relation composed of pairs (x, ω) where x and ω are called statement and witness, respectively. Also, let $\mathcal{L}$ be the language composed of all statements for which there exists a witness ω in $\mathcal{R}$, i.e., $\mathcal{L} = \{x\}_{(x,\omega)\in\mathcal{R}}$. A non-interactive argument system Π for a relation $\mathcal{R}$ is composed of the following polynomial-time algorithms:

$\mathsf{Setup}(1^\lambda, \mathcal{R})$**:** The randomized setup algorithm takes as input the security parameter 1^λ and a relation $\mathcal{R}$. It outputs a common reference string crs and a verification key vrs.

$\mathsf{Prove}(\mathsf{crs}, x, \omega)$**:** The randomized prover algorithm takes as input the common reference string crs, a statement x, and a witness ω. It outputs a proof π.

$\mathsf{Verify}(\mathsf{vrs}, x, \pi)$**:** The deterministic verification algorithm takes as input the verification key vrs, a statement x, and a proof π. It outputs a decision bit b.

We require a non-interactive argument system to be complete, i.e., honest proofs correctly verify. As for security, we consider two different definitions with respect to DV setting: *selective soundness* and *straight-line knowledge soundness*. The former says that it must be infeasible to find a proof that correctly verifies with respect to a statement $x \notin \mathcal{L}$ where x is chosen before the execution of Setup. On other hand, the latter says that there exists a universal extractor Ext that, on input a trapdoor td, is able to extract a witness ω (such that $(x, \omega) \in \mathcal{R}$) from any pair (x, π) that correctly verifies, i.e., $\mathsf{Verify}(\mathsf{vrs}, x, \pi) = 1$. Both definitions are for the designated verifier (DV) setting, i.e., vrs is kept secret and the adversary has oracle access to $\mathsf{Verify}(\mathsf{vrs}, \cdot, \cdot)$

Definition A.1 (Completeness). *A non-interactive proof system Π for a relation $\mathcal{R}$ is complete if $\forall \lambda \in \mathbb{N}$, $\forall (x, \omega) \in \mathcal{R}$ we have:*

$$\mathbb{P}\left[\mathsf{Verify}(\mathsf{vrs}, x, \mathsf{Prove}(\mathsf{crs}, x, \omega)) = 1 \middle| (\mathsf{crs}, \mathsf{vrs}) \leftarrow\!\$\, \mathsf{Setup}(1^\lambda, \mathcal{R})\right] \geq 1 - \mathsf{negl}(\lambda).$$

Definition A.2 (Selective Soundness). *A non-interactive argument system Π for a relation $\mathcal{R}$ satisfies selective soundness if, for every $x \notin \mathcal{L}$, for every PPT adversary A, we have:*

$$\mathbb{P}\left[\mathsf{Verify}(\mathsf{vrs}, x, \pi) = 1 \middle| \begin{array}{rl} (\mathsf{crs}, \mathsf{vrs}) & \leftarrow\!\$\, \mathsf{Setup}(1^\lambda, \mathcal{R}) \\ \pi & \leftarrow\!\$\, \mathsf{A}^{\mathsf{Verify}(\mathsf{vrs},\cdot,\cdot)}(1^\lambda, \mathsf{crs}) \end{array}\right] \leq \mathsf{negl}(\lambda).$$

The following definition models the ability of an extractor to be able to output a prover immediately, without having to further invoke the adversary. These extractors are often called *straight-line* and have been shown to be interesting for compiling both interactive and idealized proof schemes into concrete non-interactive ones [Fis05, CFF+21].

Definition A.3 (Straight-line Knowledge Soundness). *A non-interactive argument system Π for a relation $\mathcal{R}$ satisfies straight-line knowledge soundness if there exists a PPT algorithm $\mathsf{Ext} = (\mathsf{Ext}_0, \mathsf{Ext}_1)$ such that:*

Indistinguishability. *For every PPT adversary D, we have:*

$$\Big|\mathbb{P}\Big[\mathsf{D}(1^\lambda, \mathsf{crs}, \mathsf{vrs}) = 1 \Big| (\mathsf{crs}, \mathsf{vrs}) \leftarrow\!\!\$\ \mathsf{Setup}(1^\lambda, \mathcal{R})\Big] - \mathbb{P}\Big[\mathsf{D}(1^\lambda, \mathsf{crs}, \mathsf{vrs}) = 1 \Big| (\mathsf{crs}, \mathsf{vrs}, \mathsf{td}) \leftarrow\!\!\$\ \mathsf{Ext}_0(1^\lambda, \mathcal{R})\Big]\Big| \leq \mathsf{negl}(\lambda).$$

Extractability. *For every PPT adversary A, we have:*

$$\mathbb{P}\left[\begin{array}{c} (x,\omega) \notin \mathcal{R} \\ \wedge \\ \mathsf{Verify}(\mathsf{vrs}, x, \pi) = 1 \end{array} \middle| \begin{array}{cl} (\mathsf{crs}, \mathsf{vrs}, \mathsf{td}) & \leftarrow\!\!\$\ \mathsf{Ext}_0(1^\lambda, \mathcal{R}) \\ (x, \pi) & \leftarrow\!\!\$\ \mathsf{A}^{\mathsf{Verify}(\mathsf{vrs},\cdot,\cdot)}(1^\lambda, \mathsf{crs}) \\ \omega & \leftarrow\!\!\$\ \mathsf{Ext}_1(1^\lambda, \mathsf{td}, x, \pi) \end{array}\right] \leq \mathsf{negl}(\lambda).$$

Definitions A.2 and A.3 are for designated verifier (DV) non-interactive argument systems. We extend them to the publicly verifiable (PV) case.

Definition A.4 (Public Verifiability). *A non-interactive argument system Π for a relation $\mathcal{R}$ is publicly verifiable (PV) if selective soundness (resp. Straight-line Knowledge Soundness) holds even if the verification key vrs is given to the adversary A.*

A.3 One-way and (Puncturable) Pseudorandom Functions

One-way functions. Let $\ell_{in}(\cdot)$, $\ell_{out}(\cdot)$, and $\mathcal{F} = \{\mathsf{F}_\lambda : \{0,1\}^{\ell_{in}(\lambda)} \to \{0,1\}^{\ell_{out}(\lambda)}\}_{\lambda\in\mathbb{N}}$ be two polynomials and an ensemble of functions, respectively. We say a function $\mathsf{F}_\lambda \in \mathcal{F}$ is a one-way function (OWF) if it is computationally infeasible to find $x' \in \{0,1\}^{\ell_{in}(\lambda)}$ such that $\mathsf{F}_\lambda(x) = \mathsf{F}_\lambda(x')$ where $x \leftarrow\!\!\$\ \{0,1\}^{\ell_{in}(\lambda)}$.

Definition A.5. *We say $\mathsf{F}_\lambda \in \mathcal{F}$ is a OWF if for every PPT adversary A, we have:*

$$\mathbb{P}\Big[\mathsf{F}_\lambda(\mathsf{A}(1^\lambda, \mathsf{F}_\lambda(x))) = \mathsf{F}_\lambda(x) \Big| x \leftarrow\!\!\$\ \{0,1\}^{\ell_{in}(\lambda)}\Big] \leq \mathsf{negl}(\lambda).$$

Pseudorandom functions. A pseudorandom function (PRF) scheme $\Pi = (\mathsf{Gen}, \mathsf{F})$ with input space $\{0,1\}^{\ell_{in}}$ and output space $\{0,1\}^{\ell_{out}}$ is composed of the following polynomial-time algorithms:

$\mathsf{KGen}(1^\lambda)$**:** The randomized key generation algorithm takes as input the security parameter 1^λ and outputs a key s.

$\mathsf{F}(\mathsf{s}, x)$**:** The deterministic function evaluation algorithm takes as input a key s and an input $x \in \{0,1\}^{\ell_{in}}$, it outputs a value $y \in \{0,1\}^{\ell_{out}}$.

A PRF Π is considered secure (i.e., pseudorandom) if its output distribution is indistinguishable to the one of a truly random function.

Definition A.6 (Security of PRF). *A PRF Π is secure if for every PPT adversary D, we have:*

$$\Big|\mathbb{P}\Big[\mathsf{D}^{\mathsf{F}(\mathsf{s},\cdot)}(1^\lambda) = 1\Big] - \mathbb{P}\Big[\mathsf{D}^{\mathsf{F}_{\mathsf{rnd}}(\cdot)}(1^\lambda) = 1\Big]\Big| \leq \mathsf{negl}(\lambda),$$

where $\mathsf{s} \leftarrow\!\!\$\ \mathsf{Gen}(1^\lambda)$ and $\mathsf{F}_{\mathsf{rnd}} : \{0,1\}^{\ell_{in}} \to \{0,1\}^{\ell_{out}}$ is a truly random function.

Puncturable pseudorandom functions [HKW15]. A puncturable PRF scheme $\Pi = (\mathsf{Gen}, \mathsf{F}, \mathsf{Punct})$ offers an additional polynomial-time algorithm Punct defined as follows:

$\mathsf{Punct}(\mathsf{s}, x)$**:** The deterministic puncturing algorithm takes as input a key s and an input $x \in \{0,1\}^{\ell_{in}}$, it outputs a punctured key s'.

We require a puncturable PRF to be *correct under puncturing* and *pseudorandom at punctured inputs.*

Definition A.7 (Correctness of puncturable PRF). *A puncturable PRF Π is correct if $\forall \lambda \in \mathbb{N}$, $\forall x, x' \in \{0,1\}^{\ell_{in}}$ such that $x \neq x'$, we have*

$$\mathbb{P}\big[\mathsf{F}(\mathsf{s}, x') = \mathsf{F}(\mathsf{s}', x') | \mathsf{s} \leftarrow\$ \mathsf{Gen}(1^\lambda), \mathsf{s}' = \mathsf{Punct}(\mathsf{s}, x)\big] = 1.$$

Definition A.8 (Security of puncturable PRF). *A PRF Π is secure if for every $x \in \{0,1\}^{\ell_{in}}$, every PPT adversary D, we have:*

$$\big|\mathbb{P}\big[\mathsf{D}(1^\lambda, \mathsf{s}', \mathsf{F}(\mathsf{s}, x)) = 1\big] - \mathbb{P}\big[\mathsf{D}(1^\lambda, \mathsf{s}', y) = 1\big]\big| \leq \mathsf{negl}(\lambda),$$

where $\mathsf{s} \leftarrow\$ \mathsf{KGen}(1^\lambda)$, $\mathsf{s}' = \mathsf{Punct}(\mathsf{s}, x)$, and $y \leftarrow\$ \{0,1\}^{\ell_{out}}$.

A.4 Message Authentication Codes

A message authentication code (MAC) Π with message space $\mathcal{M}$ is composed of the following polynomial-time algorithms:

$\mathsf{KGen}(1^\lambda)$**:** The randomized key generation algorithm takes as input the security parameter 1^λ and outputs a key k. Optionally, KGen takes as input an additional parameter 1^q and k's size can depend on 1^q.

$\mathsf{Tag}(\mathsf{k}, m)$**:** The randomized tagging algorithm takes as input a key sk and a message m. It outputs a tag σ.

$\mathsf{Verify}(\mathsf{k}, m, \sigma)$**:** The deterministic verification algorithm takes as input a key k, a message $m \in \mathcal{M}$, and a tag σ. It outputs a decision bit b.

We consider MACs that are correct and strong existentially unforgeable under selective chosen message attacks ((q)-sEUF-sel-CMA), i.e., fresh valid tags are unforgeable if the adversary can ask a fixed number q of tags (for arbitrary messages) in a selective fashion (note that this is weaker than the standard sEUF-CMA security in which the adversary has adaptive and unbounded access to tagging oracle Tag).

Definition A.9 (Correctness of MACs). *A MAC scheme Π is correct if $\forall \lambda \in \mathbb{N}$, $\forall m \in \mathcal{M}$, we have that:*

$$\mathbb{P}\big[\mathsf{Verify}(\mathsf{k}, m, \mathsf{Tag}(\mathsf{k}, m)) = 1 | \mathsf{k} \leftarrow\$ \mathsf{KGen}(1^\lambda)\big] \geq 1 - \mathsf{negl}(\lambda).$$

Definition A.10 ((q)-sEUF-sel-CMA security of MACs). *A MAC scheme Π with message space $\mathcal{M}$ is strong existentially unforgeable under selective chosen message attacks in the (q)-bounded setting ((q)-sEUF-sel-CMA) if for every $(m_1, \ldots, m_q) \in \mathcal{M}^q$, every PPT adversary A, we have that:*

$$\mathbb{P}\left[\begin{array}{c|l} \forall i \in [q], (m, \sigma) \neq (m_i, \sigma_i) & \mathsf{k} \leftarrow\$ \mathsf{KGen}(1^\lambda, 1^q) \\ \wedge & \forall i \in [q], \sigma_i \leftarrow\$ \mathsf{Tag}(\mathsf{k}, m_i) \\ \mathsf{Verify}(\mathsf{k}, m, \sigma) = 1 & (m, \sigma) \leftarrow\$ \mathsf{A}(1^\lambda, \sigma_1, \ldots, \sigma_q) \end{array}\right] \leq \mathsf{negl}(\lambda).$$

In addition, we also consider a weaker definition of security, named *existential unforgeability* (EUF). In this definition, the adversary does not have oracle access to Tag.

Definition A.11 (EUF security of MACs). *A MAC scheme Π with message space $\mathcal{M}$ is existentially unforgeable (EUF) if for every $m \in \mathcal{M}$, every PPT adversary A, we have that:*

$$\mathbb{P}\big[\mathsf{Verify}(\mathsf{k}, m, \sigma) = 1 | \mathsf{k} \leftarrow\$ \mathsf{KGen}(1^\lambda), \sigma \leftarrow\$ \mathsf{A}(1^\lambda, m)\big] \leq \mathsf{negl}(\lambda).$$

A.5 Digital Signatures

A signature scheme Π with message space $\mathcal{M}$ is composed of the following polynomial-time algorithms:

$\mathsf{KGen}(1^\lambda)$**:** The randomized key generation algorithm takes as input the security parameter 1^λ and outputs a signing key sk and a public key pk. Optionally, KGen takes as input an additional parameter 1^q and pk's size can depend on 1^q.

$\mathsf{Sign}(\mathsf{sk}, m)$**:** The randomized signing algorithm takes as input a signing key sk and a message m. It outputs a signature σ.

$\mathsf{Verify}(\mathsf{pk}, m, \sigma)$**:** The deterministic verification algorithm takes as input the public key pk, a message $m \in \mathcal{M}$, and a signature σ. It outputs a decision bit b.

Similarly to MACs, we consider correctness and strong existential unforgeability under selective chosen message attacks in the (q)-bounded setting ((q)-sEUF-sel-CMA). In addition, we consider the notion of selective existential unforgeability under (adaptive) chosen message attacks (sel-EUF-CMA), i.e., the adversary must commit on a target message m before getting adaptive access to the oracle Sign.

Definition A.12 (Correctness of signatures). *A digital signature scheme Π is correct if $\forall \lambda \in \mathbb{N}$, $\forall m \in \mathcal{M}$, we have that:*

$$\mathbb{P}\big[\mathsf{Verify}(\mathsf{pk}, m, \mathsf{Sign}(\mathsf{sk}, m)) = 1 \big| (\mathsf{sk}, \mathsf{pk}) \leftarrow\!\!\$\ \mathsf{KGen}(1^\lambda)\big] = 1.$$

Definition A.13 ((q)-sEUF-sel-CMA security of signatures). *A signature scheme Π with message space $\mathcal{M}$ is strong existentially unforgeable under selective chosen message attacks in the (q)-bounded setting ((q)-sEUF-sel-CMA) if for every $(m_1, \ldots, m_q) \in \mathcal{M}^q$, every PPT adversary A, we have that:*

$$\mathbb{P}\left[\begin{array}{c} \forall i \in [q], (m, \sigma) \neq (m_i, \sigma_i) \\ \wedge \\ \mathsf{Verify}(\mathsf{pk}, m, \sigma) = 1 \end{array} \middle| \begin{array}{l} (\mathsf{sk}, \mathsf{pk}) \leftarrow\!\!\$\ \mathsf{KGen}(1^\lambda, 1^q) \\ \forall i \in [q], \sigma_i \leftarrow\!\!\$\ \mathsf{Sign}(\mathsf{sk}, m_i) \\ (m, \sigma) \leftarrow\!\!\$\ \mathsf{A}(1^\lambda, \mathsf{pk}, \sigma_1, \ldots, \sigma_q) \end{array}\right] \leq \mathsf{negl}(\lambda).$$

Definition A.14 (sel-EUF-CMA security of signatures). *A signature scheme Π with message space $\mathcal{M}$ is selectively existentially unforgeable under chosen message attacks (sel-EUF-CMA) if for every $m \in \mathcal{M}$, every PPT adversary A, we have that:*

$$\mathbb{P}\left[m \notin \mathcal{Q}_{\mathsf{Sign}} \wedge \mathsf{Verify}(\mathsf{pk}, m, \sigma) = 1 \middle| \begin{array}{l} (\mathsf{sk}, \mathsf{pk}) \leftarrow\!\!\$\ \mathsf{KGen}(1^\lambda) \\ \sigma \leftarrow\!\!\$\ \mathsf{A}^{\mathsf{Sign}(\mathsf{sk}, \cdot)}(1^\lambda, \mathsf{pk}, m) \end{array}\right] \leq \mathsf{negl}(\lambda),$$

where $\mathcal{Q}_{\mathsf{Sign}}$ is the set of messages submitted to the oracle Sign*.*

A.6 Symmetric Key Encryption

A symmetric encryption (SKE) scheme with message space $\mathcal{M}$ is composed of the following polynomial-time algorithms:

$\mathsf{KGen}(1^\lambda)$**:** The randomized key generator takes as input the security parameter 1^λ and outputs a symmetric key k.

$\mathsf{Enc}(\mathsf{k}, m)$**:** The randomized encryption algorithm takes as input a symmetric key k and a message $m \in \mathcal{M}$, it outputs a ciphertext c.

$\mathsf{Dec}(\mathsf{k}, c)$**:** The deterministic decryption algorithm takes as input a symmetric key k and a ciphertext c, it outputs a message m.

A SKE is correct if honest ciphertexts correctly decrypt.

Definition A.15 (Correctness of SKE). *A SKE Π with message space $\mathcal{M}$ is correct if $\forall \lambda \in \mathbb{N}$, $\forall m \in \mathcal{M}$, we have*

$$\mathbb{P}\big[\mathsf{Dec}(\mathsf{k}, \mathsf{Enc}(\mathsf{k}, m)) = m \big| \mathsf{k} \leftarrow\!\!\$\ \mathsf{KGen}(1^\lambda)\big] = 1.$$

$\mathbf{G}^{\mathsf{SKEsec}}_{\Pi,\mathsf{A},m_0,m_1}(\lambda)$

$\mathsf{k} \leftarrow\$ \mathsf{KGen}(1^\lambda)$
$b \leftarrow\$ \{0,1\}$
$c \leftarrow\$ \mathsf{Enc}(\mathsf{k}, m_b)$
$b' \leftarrow\$ \mathsf{A}(1^\lambda, c)$
If $b' = b$, **return** 1
return 0

$\mathbf{G}^{\mathsf{SKEcprakey}}_{\Pi,\mathsf{A},m}(\lambda)$

$\mathsf{k}_0 \leftarrow\$ \mathsf{KGen}(1^\lambda), \mathsf{k}_1 \leftarrow\$ \mathsf{KGen}(1^\lambda)$
$b \leftarrow\$ \{0,1\}$
$c \leftarrow\$ \mathsf{Enc}(\mathsf{k}_b, m)$
$b' \leftarrow\$ \mathsf{A}^{\mathsf{Enc}(\mathsf{k}_0,\cdot;\cdot),\mathsf{Enc}(\mathsf{k}_1,\cdot;\cdot)}(1^\lambda, c)$
If $b' = b$, **return** 1
return 0

$\mathbf{G}^{\mathsf{PKEcpa}}_{\Pi,\mathsf{A},m_0,m_1}(\lambda)$

$(\mathsf{sk}, \mathsf{pk}) \leftarrow\$ \mathsf{KGen}(1^\lambda)$
$b \leftarrow\$ \{0,1\}$
$c \leftarrow\$ \mathsf{Enc}(\mathsf{pk}, m_b)$
$b' \leftarrow\$ \mathsf{A}(1^\lambda, \mathsf{pk}, c)$
If $b' = b$, **return** 1
return 0

$\mathbf{G}^{\mathsf{SKEcca1}}_{\Pi,\mathsf{A}}(\lambda)$

$\mathsf{k} \leftarrow\$ \mathsf{KGen}(1^\lambda)$
$m_0, m_1 \leftarrow\$ \mathsf{A}_0^{\mathsf{Enc}(\mathsf{k},\cdot),\mathsf{Dec}(\mathsf{k},\cdot)}(1^\lambda)$
$b \leftarrow\$ \{0,1\}$
$c \leftarrow\$ \mathsf{Enc}(\mathsf{k}, m_b)$
$b' \leftarrow\$ \mathsf{A}_1^{\mathsf{Enc}(\mathsf{k},\cdot)}(1^\lambda, c)$
If $b' = b$, **return** 1
return 0

$\mathbf{G}^{\mathsf{SKEcpakey}}_{\Pi,\mathsf{A}}(\lambda)$

$\mathsf{k}_0 \leftarrow\$ \mathsf{KGen}(1^\lambda), \mathsf{k}_1 \leftarrow\$ \mathsf{KGen}(1^\lambda)$
$m \leftarrow\$ \mathsf{A}_0^{\mathsf{Enc}(\mathsf{k}_0,\cdot),\mathsf{Enc}(\mathsf{k}_1,\cdot)}(1^\lambda)$
$b \leftarrow\$ \{0,1\}$
$c \leftarrow\$ \mathsf{Enc}(\mathsf{k}_b, m)$
$b' \leftarrow\$ \mathsf{A}_1^{\mathsf{Enc}(\mathsf{k}_0,\cdot),\mathsf{Enc}(\mathsf{k}_1,\cdot)}(1^\lambda, c)$
If $b' = b$, **return** 1
return 0

Fig. 9: Game defining semantic, IND-CCA1, IND-CPA-key, sel-IND-CPRA-key of SKE and sel-IND-CPA of PKE. The top three games (that define semantic, sel-IND-CPRA-key, and sel-IND-CPA security) are parametrized by two (or one) messages since they only cover selective security, while the bottom games cover also the adaptive case where the adversary is allowed to choose the messages after seeing the public key.

We now define different flavors of security in both selective and adaptive setting.

First, we consider the standard *semantic security* and *security under chosen ciphertext attacks (IND-CCA1)*. In the IND-CCA1 experiment, the adversary has oracle access to Enc and Dec where Dec is available only before the selection of the messages m_0 and m_1.

Definition A.16 (Semantic security of SKE). *We say that a SKE Π with message space $\mathcal{M}$ is* semantically secure *if for every $m_0, m_1 \in \mathcal{M}$, every PPT adversaries A, we have:*

$$\left|\mathbb{P}\left[\mathbf{G}^{\mathsf{SKEsec}}_{\Pi,\mathsf{A},m_0,m_1}(\lambda) = 1\right] - \frac{1}{2}\right| \leq \mathsf{negl}(\lambda),$$

where the experiment $\mathbf{G}^{\mathsf{SKEsec}}_{\Pi,\mathsf{A},m_0,m_1}(\lambda)$ is depicted in Figure 9.

Definition A.17 (IND-CCA1 security of SKE). *We say that a SKE Π with message space $\mathcal{M}$ is* secure under chosen ciphertext attacks (IND-CCA1) *if for every PPT adversaries $\mathsf{A} = (\mathsf{A}_0, \mathsf{A}_1)$, we have:*

$$\left|\mathbb{P}\left[\mathbf{G}^{\mathsf{SKEcca1}}_{\Pi,\mathsf{A}}(\lambda) = 1\right] - \frac{1}{2}\right| \leq \mathsf{negl}(\lambda),$$

where the experiment $\mathbf{G}^{\mathsf{SKEcca1}}_{\Pi,\mathsf{A}}(\lambda)$ is depicted in Figure 9.

Second, we consider SKEs that are *key indistinguishable*, i.e., a computationally bounded adversary A cannot determine which key (between $\mathsf{k}_0 \leftarrow\$ \mathsf{KGen}(\lambda)$ and $\mathsf{k}_1 \leftarrow\$ \mathsf{KGen}(\lambda)$) has been used to encrypt an adversarially chosen message m. We define key indistinguishability with respect to two different models: (*i*) *adaptive message and chosen plaintext attacks (IND-CPA-key)* and (*ii*) *selective message and chosen plaintext randomness attacks (sel-IND-CPRA-key)*. The IND-CPA-key experiment allows an adversary, with oracle access to $\mathsf{Enc}(\mathsf{k}_0,\cdot)$ and $\mathsf{Enc}(\mathsf{k}_1,\cdot)$, to adaptively choose the message m. On the other hand, in the sel-IND-CPRA-key experiment, the adversary is required to commit on the message m before getting oracle access to $\mathsf{Enc}(\mathsf{k}_0,\cdot;\cdot)$ and $\mathsf{Enc}(\mathsf{k}_1,\cdot;\cdot)$ where the latter oracles accept adversarially chosen plaintexts and randomnesses.

Definition A.18 (IND-CPA-key security of SKE). *We say that a SKE Π with message space $\mathcal{M}$ is* key indistinguishable under chosen plaintext attacks (IND-CPA-key) *if for every PPT adversaries $\mathsf{A} = (\mathsf{A}_0, \mathsf{A}_1)$, we have:*

$$\left|\mathbb{P}\left[\mathbf{G}^{\mathsf{SKEcpakey}}_{\Pi,\mathsf{A}}(\lambda) = 1\right] - \frac{1}{2}\right| \leq \mathsf{negl}(\lambda),$$

where the experiment $\mathbf{G}^{\mathsf{SKEcpakey}}_{\Pi,\mathsf{A}}(\lambda)$ is depicted in Figure 9.

Definition A.19 (sel-IND-CPRA-key of SKE). *We say that a SKE Π with message space $\mathcal{M}$ is* selectively key indistinguishable under chosen plaintext randomness attacks (sel-IND-CPRA-key) *if for every $m \in \mathcal{M}$, every PPT adversaries A, we have:*

$$\left|\mathbb{P}\left[\mathbf{G}^{\mathsf{SKEcprakey}}_{\Pi,\mathsf{A},m}(\lambda) = 1\right] - \frac{1}{2}\right| \leq \mathsf{negl}(\lambda),$$

where the experiment $\mathbf{G}^{\mathsf{SKEcprakey}}_{\Pi,\mathsf{A},m}(\lambda)$ is depicted in Figure 9.

Through the paper, we leverage the above definitions of security to prove two main results. In Section 5.5, we show that oiO is (potentially) able to compile any semantically secure and sel-IND-CPRA-key SKE Π into a sel-IND-CPA PKE Π' (see Appendix A.7 and definition A.22). On the other hand, in Section 3, we make use of the adaptive IND-CCA1 and IND-CPA-key definitions to prove some *unconditional* impossibility results for both odiO and oiO.

Lastly, we stress that all the above definitions follow from OWFs. Indeed, the well known $\mathsf{Enc}(\mathsf{k}, m; r) = (\mathsf{F}(\mathsf{k}, r) \oplus m, r)$ (where F is a PRF) satifies both IND-CCA1 (and in turn semantic security) and IND-CPA-key. Moreover, any sel-IND-CPA-key SKE scheme $\Pi = (\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ can be transformed into a sel-IND-CPRA-key SKE scheme $\Pi^* = (\mathsf{KGen}^*, \mathsf{Enc}^*, \mathsf{Dec})$ by simply setting $\mathsf{Enc}^*(\mathsf{k}^*, m; r) = \mathsf{Enc}(\mathsf{k}, m; \mathsf{F}(\mathsf{s}, (m, r)))$ where $\mathsf{k}^* = (\mathsf{k}, \mathsf{s}) \leftarrow\!\!\$\ \mathsf{KGen}^*(1^\lambda)$.[15]

Corollary A.20. *If OWFs exist then there exists a SKE scheme Π that satifies Definitions A.15 to A.19.*

A.7 Public Key Encryption

A public key encryption (PKE) scheme with message space $\mathcal{M}$ is composed of the following polynomial-time algorithms:

$\mathsf{KGen}(1^\lambda)$**:** The randomized key generator takes as input the security parameter 1^λ and outputs a secret key sk and a public key pk.

$\mathsf{Enc}(\mathsf{pk}, m)$**:** The randomized encryption algorithm takes as input a public key pk and a message $m \in \mathcal{M}$, it outputs a ciphertext c.

$\mathsf{Dec}(\mathsf{sk}, c)$**:** The deterministic decryption algorithm takes as input a secret key sk and a ciphertext c, it outputs a message m.

We consider PKEs that are correct and selectively secure under chosen plaintext attacks (sel-IND-CPA), i.e., the messages m_0, m_1 are chosen before executing KGen.

Definition A.21 (Correctness of PKE). *A PKE Π with message space $\mathcal{M}$ correct if $\forall \lambda \in \mathbb{N}$, $\forall m \in \mathcal{M}$, we have*

$$\mathbb{P}\left[\mathsf{Dec}(\mathsf{sk}, \mathsf{Enc}(\mathsf{pk}, m)) = m | (\mathsf{pk}, \mathsf{sk}) \leftarrow\!\!\$\ \mathsf{KGen}(1^\lambda)\right] = 1.$$

Definition A.22 (sel-IND-CPA security of PKE). *We say that a PKE Π is* selectively secure under chosen plaintext attacks (sel-IND-CPA) *if for every $m_0, m_1 \in \mathcal{M}$, every PPT adversary A:*

$$\left|\mathbb{P}\left[\mathbf{G}^{\mathsf{PKEcpa}}_{\Pi,\mathsf{A},m_0,m_1}(\lambda) = 1\right] - \frac{1}{2}\right| \leq \mathsf{negl}(\lambda),$$

where game $\mathbf{G}^{\mathsf{PKEcpa}}_{\Pi,\mathsf{A},m_0,m_1}(\lambda)$ is depicted in Figure 9.

Sometimes, we will consider the (standard) adaptive version of the above definition of security, i.e., *security under chosen plaintext attacks (IND-CPA)*.

[15] We stress that the same transformation achieves adaptive security, i.e., any IND-CPA-key SKE scheme can be transformed into a IND-CPRA-key scheme (the adaptive flavor of Definition A.19).

B Supporting Proofs

B.1 Proof of Theorem 4.6

VBB ⇒ oiO. We start by proving the following lemma.

Lemma B.1. *If there exists a PPT* Obf *obfuscator such that* Obf *is a* $(\{\mathsf{S}_0, \mathsf{S}_1\})$-VBB-*obfuscator (Definition 3.5) then* Obf *is a* $(\{\mathsf{S}\})$-oiO-*obfuscator (Definition 4.2).*

Proof. By contradiction, assume that Obf is not a $(\{\mathsf{S}\})$-oiO-obfuscator, i.e., there exist a PPT adversary D such that

$$\left|\mathbb{P}\left[\mathsf{D}(1^\lambda, \mathsf{Obf}(1^\lambda, C_0), \alpha) = 1\right] - \mathbb{P}\left[\mathsf{D}(1^\lambda, \mathsf{Obf}(1^\lambda, C_1), \alpha) = 1\right]\right| \geq \epsilon, \tag{3}$$

where $(C_0, C_1, \alpha) \leftarrow\!\!\$\ \mathsf{S}(1^\lambda)$ and ϵ non-negligible. By definition of $\mathsf{S}_0, \mathsf{S}_1$, the following condition holds:

$$\forall r \in \{0,1\}^*, C_0 = C_0' \wedge C_1 = C_1' \wedge \alpha = \alpha', b$$

where $(C_0, C_1, \alpha) = \mathsf{S}(1^\lambda; r)$,$(C_0', \alpha') = \mathsf{S}_0(1^\lambda; r)$, and $(C_1', \alpha') = \mathsf{S}_1(1^\lambda; r)$. Hence, we can rewrite Equation (3) as follows:

$$\left|\mathbb{P}\left[\mathsf{D}(1^\lambda, \mathsf{Obf}(1^\lambda, C_0'), \alpha') = 1\right] - \mathbb{P}\left[\mathsf{D}(1^\lambda, \mathsf{Obf}(1^\lambda, C_1'), \alpha') = 1\right]\right| \geq \epsilon, \tag{4}$$

where $r \leftarrow\!\!\$\ \{0,1\}^*$, $(C_0', \alpha') = \mathsf{S}_0(1^\lambda; r)$, and $(C_1', \alpha') = \mathsf{S}_1(1^\lambda; r)$. By leveraging the fact that Obf is a $(\{\mathsf{S}_0, \mathsf{S}_1\})$-VBB-obfuscator, we conclude that there exists a PPT simulator Sim such that for every $\mathsf{S}_b \in \{\mathsf{S}_0, \mathsf{S}_1\}$:

$$\left|\mathbb{P}\left[\mathsf{D}(1^\lambda, \mathsf{Obf}(1^\lambda, C_b'), \alpha') = 1\right] - \mathbb{P}\left[\mathsf{Sim}^{C_b'(\cdot)}(1^\lambda, 1^{|C_b'|}, \alpha') = 1\right]\right| \leq \mathsf{negl}(\lambda), \tag{5}$$

where $r \leftarrow\!\!\$\ \{0,1\}^*$ and $(C_b, \alpha') \leftarrow\!\!\$\ \mathsf{S}_b(1^\lambda)$. By combining Equations (3) to (5) we conclude that

$$\left|\mathbb{P}\left[\mathsf{Sim}^{C_0'(\cdot)}(1^\lambda, 1^{|C_0'|}, \alpha') = 1\right] - \mathbb{P}\left[\mathsf{Sim}^{C_1'(\cdot)}(1^\lambda, 1^{|C_1'|}, \alpha') = 1\right]\right| \geq \epsilon + \mathsf{negl}(\lambda), \tag{6}$$

where $r \leftarrow\!\!\$\ \{0,1\}^*$, $(C_0', \alpha') = \mathsf{S}_0(1^\lambda; r)$, and $(C_1', \alpha') = \mathsf{S}_1(1^\lambda; r)$. Since ϵ is non-negligible, Equation (6) contradicts the fact that S is an oiO-sampler. This concludes the proof. ☐

We now use Lemma B.1 to prove Theorem 4.6. By contradiction, suppose $\mathsf{S} \notin \mathcal{S}_{\mathsf{oiO}}$. This implies that, for every ensemble of oiO-samplers $\mathcal{S}$ such that $\mathsf{S} \in \mathcal{S}$, it does not exists a PPT Obf that is a $(\mathcal{S})$-oiO-obfuscator. By leveraging Lemma B.1, we can also conclude that it must not exists a PPT Obf′ that is a $(\{\mathsf{S}_0, \mathsf{S}_1\})$-VBB-obfuscator. As a consequence, it must be that either $\mathsf{S}_0 \notin \mathcal{S}_{\mathsf{VBB}}$ or $\mathsf{S}_1 \notin \mathcal{S}_{\mathsf{VBB}}$. This concludes the proof.

VBB ⇒ odiO. This case follows by combining the above argument and the fact that oiO ⇒ odiO (Theorem 4.5).

B.2 Proof of Theorem 5.1

(Part one) S_x is an odiO-sampler. By contradiction, suppose there exists $x^* \notin \mathcal{L}$ such that S_{x^*} is not an odiO-sampler, i.e., there exists a PPT adversary A such that

$$\mathbb{P}\left[C_0(v) \neq C_1(v) \middle| v \leftarrow\!\!\$\ \mathsf{A}^{C_0(\cdot), C_1(\cdot)}(1^\lambda, 1^{|C_0|}, \alpha)\right] \geq \epsilon,$$

where $(C_0, C_1, \alpha) \leftarrow\!\!\$\ \mathsf{S}_{x^*}(1^\lambda)$ and ϵ non-negligible. We build an adversary A′ that breaks the selective soundness of Π^* with respect to the statement $x^* \notin \mathcal{L}$. The adversary A′ proceeds as follows:

1. Receive crs^* from the challenger.

2. Let $C_0^* = C^{\mathsf{Verify}}_{\mathsf{vrs}^*}$,$C_1 = C^{\mathsf{Verify}}_{\mathsf{vrs}^*,x^*}$ and $\alpha = \mathsf{crs}^*$ (note that both C_0^* and C_1^* are unknown to A' since vrs^* is kept secret by the challenger).
3. Send α and 1^γ (where γ as defined in Figure 2) to A and answer to the incoming queries as follows:
   (a) On input (x, π) for C_0^*, A' forwards (x, π) to the oracle $\mathsf{Verify}^*(\mathsf{vrs}^*, \cdot, \cdot)$ and returns the answer.
   (b) On input (x, π) for C_1^*, if $x = x^*$, return 0. Otherwise, forward (x, π) to the oracle $\mathsf{Verify}^*(\mathsf{vrs}^*, \cdot, \cdot)$ and return the answer.
4. Finally, A' receives $v = (\widehat{x}, \widehat{\pi})$ from A. It forwards $\widehat{\pi}$ to the challenger.

It is easy to see that A' perfectly simulates the view of A. This because crs^* and vrs^* (generated by the challenger) have the exact same distribution to the one generated by S_{x^*}. Moreover, we have that A' perfectly simulates both circuits C_0^* and C_1^*. Observe that $v = (\widehat{x}, \widehat{\pi})$ is a differing-input for C_0^* and C_1^* only if $\widehat{x} = x^*$ and $\mathsf{Verify}^*(\mathsf{vrs}^*, x^*, \widehat{\pi}) = 1$. Hence, A' breaks selective soundness property of Π^* with the same non-negligible advantage ϵ of A. This concludes the proof.

(Part two) Π satisfies (publicly verifiable) selective soundness. Let $x^* \notin \mathcal{L}$. Consider the following hybrid experiments:

$\mathsf{Hyb}_0^{x^*}(\lambda)$**:** This is the standard selective soundness experiment (with respect to the statement $x^* \notin \mathcal{L}$) for publicly verifiable argument systems (Definition A.2). Recall that in the publicly verifiable setting, both crs^* and vrs^* are given in input to the adversary (Definition A.4).

$\mathsf{Hyb}_1^{x^*}(\lambda)$**:** Same as $\mathsf{Hyb}_0^{x^*}$, except that the challenger obfuscates the circuit $C^{\mathsf{Verify}}_{\mathsf{vrs}^*,x^*}$ of Figure 2 (instead of $C^{\mathsf{Verify}}_{\mathsf{vrs}^*}$). Formally, the challenger computes $\mathsf{vrs} \leftarrow\!\!\$\ \mathsf{Obf}(1^\lambda, C^{\mathsf{Verify}}_{\mathsf{vrs}^*,x^*})$ where $(\mathsf{crs}^*, \mathsf{vrs}^*) \leftarrow\!\!\$\ \mathsf{Setup}^*(1^\lambda, \mathcal{R})$.

Lemma B.2. *For every $x^* \notin \mathcal{L}$, $\mathsf{Hyb}_0^{x^*}(1^\lambda) \approx_c \mathsf{Hyb}_1^{x^*}(1^\lambda)$.*

Proof. By contradiction, assume there exists $x^* \notin \mathcal{L}$ such that $\mathsf{Hyb}_0^{x^*}$ and $\mathsf{Hyb}_1^{x^*}$ are not computationally indistinguishable, i.e., there exists a PPT distinguisher D that has a non-negligible advantage in distinguishing between $\mathsf{Hyb}_0^{x^*}$ and $\mathsf{Hyb}_1^{x^*}$. We build a distinguisher D' that breaks the indistinguishability property of Obf for the odiO-sampler S_{x^*}. The distinguisher D' proceeds as follows:

1. Receive in input an obfuscated circuit $\widetilde{C}$ and α. Recall that $\widetilde{C} \leftarrow\!\!\$\ \mathsf{Obf}(1^\lambda, C_b)$ and $\alpha = \mathsf{crs}^*$ where $b \leftarrow\!\!\$\ \{0,1\}$ is the unknown challenge bit and $(C_0, C_1, \alpha) \leftarrow\!\!\$\ \mathsf{S}_{x^*}(1^\lambda)$.
2. Send $\mathsf{crs} = \mathsf{crs}^*$ and $\mathsf{vrs} = \widetilde{C}$ to D.
3. Return whatever D outputs.

It is easy to see that, if $b = 0$ then D' simulates $\mathsf{Hyb}_0^{x^*}$. On the other hand, if $b = 1$ then D' simulates $\mathsf{Hyb}_1^{x^*}$. Hence, D' retains the same non negligible advantage of D. ◻

Observe that, for every $x^* \notin \mathcal{L}$, A has advantage 0 against the selective soundness experiment of $\mathsf{Hyb}_1^{x^*}$. This because, for every $\pi \in \{0,1\}^*$, $\mathsf{Verify}(\mathsf{vrs}, x^*, \pi)$ returns 0 since $\mathsf{vrs} \leftarrow\!\!\$\ \mathsf{Obf}(1^\lambda, C^{\mathsf{Verify}}_{\mathsf{vrs}^*,x^*})$ (see the definition of $C^{\mathsf{Verify}}_{\mathsf{vrs}^*,x^*}$ depicted in Figure 2). This concludes the proof.

B.3 Proof of Theorem 5.2

(Part one) $\mathsf{S}_{\mathsf{Ext}^*}$ is an odiO-sampler. By contradiction, suppose $\mathsf{S}_{\mathsf{Ext}^*}$ is not an odiO-sampler, i.e., there exists a PPT adversary A such that:

$$\mathbb{P}\Big[C_0(v) \neq C_1(v) \Big| v \leftarrow\!\!\$\ \mathsf{A}^{C_0(\cdot),C_1(\cdot)}(1^\lambda, 1^{|C_0|}, \alpha)\Big] \geq \epsilon,$$

where $(C_0, C_1, \alpha) \leftarrow\!\!\$\ \mathsf{S}_{\mathsf{Ext}^*}(1^\lambda)$ and ϵ non-negligible. We build an adversary A' that breaks the extractability property of Π^*. The adversary A' proceeds as follows:

1. Receive crs^* from the challenger.
2. Let $C_0^* = C^{\mathsf{Verify}}_{\mathsf{vrs}^*}$,$C_1 = C^{\mathsf{Verify}}_{\mathsf{vrs}^*,\mathsf{td}^*,r_1^*}$ and $\alpha = \mathsf{crs}^*$ for random $r_1^* \leftarrow\!\!\$\ \{0,1\}^*$ (note that both C_0^* and C_1^* are unknown to A' since both vrs^* and td^* is kept secret by the challenger).
3. Send α and 1^γ (where γ as defined in Figure 2) to A and answer the incoming queries as follows:

(a) On input (x, π) for the circuit C_i^* for $i \in \{0,1\}$, A' forwards (x, π) to the oracle $\mathsf{Verify}^*(\mathsf{vrs}^*, \cdot, \cdot)$ and returns the answer.

4. Receive $v = (\widehat{x}, \widehat{\pi})$ from A.
5. Sample a random bit $b \leftarrow\!\!\$\, \{0,1\}$. If $b = 0$, A' returns $(x', \pi') \leftarrow\!\!\$\, \mathcal{Q}_{C_1^*}$ where $\mathcal{Q}_{C_1^*}$ are the queries submitted by A to the oracle C_1^*. Otherwise, if $b = 1$, A' returns $(x', \pi') = (\widehat{x}, \widehat{\pi})$.

First, note that (C_0^*, C_1^*, α) comes from a distribution that is identical to that of $\mathsf{S}_{\mathsf{Ext}^*}$; this is because vrs^* and td^* are generated by executing Ext_0^* and r_1^* is sampled at random from $\{0,1\}^*$ (as done by $\mathsf{S}_{\mathsf{Ext}^*}$).

In addition, observe that

1. If A' correctly simulates A's view with respect to (C_0^*, C_1^*, α) then $(\widehat{x}, \widehat{\pi})$ (output by A) contradicts the Straight-line Knowledge Soundness property of Π^*.
2. On the other hand, if A' fails to correctly simulate A's view with respect to (C_0^*, C_1^*, α) then there exists $(x', \pi') \in \mathcal{Q}_{C_1^*}$ (submitted by A) that contradicts the Straight-line Knowledge Soundness property of Π^*.

Consider the following events defined with respect to $(\mathsf{crs}^*, \mathsf{vrs}^*, \mathsf{td}^*, r_1^*)$:

$$\begin{aligned}
&\mathbf{Sim} : \exists (x, \pi) \in \mathcal{Q}_{C_1^*}, \mathsf{Verify}^*(\mathsf{vrs}^*, x, \pi) = 1 \wedge (x, \omega) \notin \mathcal{R} \\
&\qquad \text{where } \omega = \mathsf{Ext}_1^*(1^\lambda, \mathsf{td}^*, x, \pi; r_1^*), \\
&\mathbf{Win} : \mathsf{Verify}^*(\mathsf{vrs}^*, x', \pi') = 1 \wedge (x', \omega') \notin \mathcal{R} \\
&\qquad \text{where } \omega' = \mathsf{Ext}_1^*(1^\lambda, \mathsf{td}^*, x', \pi'; r_1^*), \\
&\mathbf{Bit} : b = 1.
\end{aligned}$$

We can bound the advantage of A' as follows:

$$\begin{aligned}
\mathbb{P}[\mathbf{Win}] &= \mathbb{P}[\mathbf{Win}|\mathbf{Sim}, \mathbf{Bit}] \cdot \mathbb{P}[\mathbf{Sim}] \cdot \mathbb{P}[\mathbf{Bit}] \\
&\quad + \mathbb{P}[\mathbf{Win}|\mathbf{Sim}, \neg\mathbf{Bit}] \cdot \mathbb{P}[\mathbf{Sim}] \cdot \mathbb{P}[\neg\mathbf{Bit}] \\
&\quad + \mathbb{P}[\mathbf{Win}|\neg\mathbf{Sim}, \mathbf{Bit}] \cdot \mathbb{P}[\neg\mathbf{Sim}] \cdot \mathbb{P}[\mathbf{Bit}] \\
&\quad + \mathbb{P}[\mathbf{Win}|\neg\mathbf{Sim}, \neg\mathbf{Bit}] \cdot \mathbb{P}[\neg\mathbf{Sim}] \cdot \mathbb{P}[\neg\mathbf{Bit}] \\
&\geq \mathbb{P}[\mathbf{Win}|\neg\mathbf{Sim}, \mathbf{Bit}] \cdot \mathbb{P}[\neg\mathbf{Sim}] \cdot \mathbb{P}[\mathbf{Bit}] \\
&\quad + \mathbb{P}[\mathbf{Win}|\mathbf{Sim}, \neg\mathbf{Bit}] \cdot \mathbb{P}[\mathbf{Sim}] \cdot \mathbb{P}[\neg\mathbf{Bit}] \\
&= \mathbb{P}[\mathbf{Win}|\neg\mathbf{Sim}, \mathbf{Bit}] \cdot \frac{1 - p_{\mathbf{Sim}}}{2} + \mathbb{P}[\mathbf{Win}|\mathbf{Sim}, \neg\mathbf{Bit}] \cdot \frac{p_{\mathbf{Sim}}}{2},
\end{aligned} \tag{7}$$

for $p_{\mathbf{Sim}} = \mathbb{P}[\mathbf{Sim}]$ and $\mathbb{P}[\mathbf{Bit}] = \mathbb{P}[\neg\mathbf{Bit}] = 1/2$. A differing-input $v = (x, \pi)$ for $C_0^* = C_{\mathsf{vrs}^*}^{\mathsf{Verify}}$ and $C_1^* = C_{\mathsf{vrs}^*, \mathsf{td}^*, r_1^*}^{\mathsf{Verify}}$ needs to satisfy the following condition

$$\mathsf{Verify}^*(\mathsf{vrs}^*, x, \pi) = 1 \wedge (x, \omega) \notin \mathcal{R}.$$

We consider two cases:

– When $\neg\mathbf{Bit}$ happens, A' outputs $(x', \pi') \leftarrow\!\!\$\, \mathcal{Q}_{C_1^*}$. Moreover, when $\mathbf{Sim}$ happens, we are guaranteed that there exists $(x, \pi) \in \mathcal{Q}_{C_1^*}$ such that $\mathsf{Verify}^*(\mathsf{vrs}^*, x, \pi) = 1 \wedge (x, \omega) \notin \mathcal{R}$. Hence, we conclude that $\mathbb{P}[\mathbf{Win}|\mathbf{Sim}, \neg\mathbf{Bit}] = 1/|\mathcal{Q}_{C_1^*}|$.
– When $\mathbf{Bit}$ happens, A' outputs $(x', \pi') = (\widehat{x}, \widehat{\pi})$ where $(\widehat{x}, \widehat{\pi})$ is the final output of A. Observe that, conditioned to the event $\neg\mathbf{Sim}$, A' correctly simulates the view of A. As a consequence, A outputs a valid differing-input $v = (\widehat{x}, \widehat{\pi})$ with non-neglibile probability. Hence, we have that $\mathbb{P}[\mathbf{Win}|\neg\mathbf{Sim}, \mathbf{Bit}] \geq \epsilon$.

By combining Equation (7) and the above conditions we conclude that

$$\mathbb{P}[\mathbf{Win}] \geq \epsilon \cdot \frac{1 - p_{\mathbf{Sim}}}{2} + \frac{1}{|\mathcal{Q}_{C_1^*}|} \cdot \frac{p_{\mathbf{Sim}}}{2} \notin \mathsf{negl}(\lambda).$$

This concludes the proof.

(Part two) Π satisfies (publicly verifiable) straight-line knowledge soundness. Let $\mathsf{Ext}^* = (\mathsf{Ext}^*_0, \mathsf{Ext}^*_1)$ be the extractor of the designated verifier non-interactive proof system Π^*. Consider the following extractor $\mathsf{Ext} = (\mathsf{Ext}_0, \mathsf{Ext}_1)$ for Π:

$\mathsf{Ext}_0(1^\lambda, \mathcal{R})$**:** On input the security parameter 1^λ and a relation $\mathcal{R}$, the algorithm outputs the common reference string $\mathsf{crs} = \mathsf{crs}^*$, the verification key $\mathsf{vrs} \leftarrow\!\!\$\ \mathsf{Obf}(1^\lambda, C^{\mathsf{Verify}}_{\mathsf{vrs}^*,\mathsf{td}^*,r_1^*})$, and the trapdoor $\mathsf{td} = (\mathsf{td}^*, r_1^*)$ where $(\mathsf{crs}^*, \mathsf{vrs}^*, \mathsf{td}^*) \leftarrow\!\!\$\ \mathsf{Ext}^*_0(1^\lambda, \mathcal{R})$ and $r_1^* \leftarrow\!\!\$\ \{0,1\}^*$.

$\mathsf{Ext}_1(1^\lambda, \mathsf{td}, x, \pi)$**:** On input the security parameter 1^λ, a trapdoor $\mathsf{td} = (\mathsf{td}^*, r_1^*)$, a statement x, and a proof π, the algorithm returns $\omega = \mathsf{Ext}^*_1(1^\lambda, \mathsf{td}^*, x, \pi; r_1^*)$.

We prove the following lemmas.

Lemma B.3. *For every PPT adversary* D*, we have:*

$$\Big|\mathbb{P}\Big[\mathsf{D}(1^\lambda, \mathsf{crs}, \mathsf{vrs}) = 1 \Big| (\mathsf{crs}, \mathsf{vrs}, \mathsf{td}) \leftarrow\!\!\$\ \mathsf{Ext}_0(1^\lambda, \mathcal{R})\Big] - \tag{8}$$

$$\mathbb{P}\left[\mathsf{D}(1^\lambda, \mathsf{crs}^*, \mathsf{vrs}) = 1 \middle| \begin{array}{l}(\mathsf{crs}^*, \mathsf{vrs}^*, \mathsf{td}^*) \leftarrow\!\!\$\ \mathsf{Ext}^*_0(1^\lambda, \mathcal{R})\\ \mathsf{vrs} \leftarrow\!\!\$\ \mathsf{Obf}(1^\lambda, C^{\mathsf{Verify}}_{\mathsf{vrs}^*})\end{array}\right]\Bigg| \leq \mathsf{negl}(\lambda). \tag{9}$$

Proof. By contradiction, assume there exists a PPT adversary D that distinguishes the above two distributions with non-neglibile advantage. We build a distinguisher D' that breaks the indistinguishability property of Obf with respect to the odiO-sampler $\mathsf{S}_{\mathsf{Ext}^*}$. The distinguisher D' proceeds as follows:

1. Receive in input an obfuscated circuit $\widetilde{C}$ and α. Recall $\widetilde{C} \leftarrow\!\!\$\ \mathsf{Obf}(1^\lambda, C_b)$ where $b \leftarrow\!\!\$\ \{0,1\}$ is the unknown challenge bit and $(C_0, C_1, \alpha) \leftarrow\!\!\$\ \mathsf{S}_{\mathsf{Ext}^*}(1^\lambda)$.
2. Send $\mathsf{crs} = \alpha$ and $\mathsf{vrs} = \widetilde{C}$ to D.
3. Return the output of D.

Let $C^*_0 = C^{\mathsf{Verify}}_{\mathsf{vrs}^*}$, $C^*_1 = C^{\mathsf{Verify}}_{\mathsf{vrs}^*,\mathsf{td}^*,r_1^*}$, and $\alpha = \mathsf{crs}^*$. If $b = 0$, D's view is distributed as in Equation (9); on the hand, if $b = 1$, D's view is distributed as Equation (8). Hence, D' has the same non-negligible advantage of D. This concludes the proof. □

Lemma B.4. *For every PPT adversary* D*, we have:*

$$\left|\mathbb{P}\left[\mathsf{D}(1^\lambda, \mathsf{crs}^*, \mathsf{vrs}) = 1 \middle| \begin{array}{l}(\mathsf{crs}^*, \mathsf{vrs}^*, \mathsf{td}^*) \leftarrow\!\!\$\ \mathsf{Ext}^*_0(1^\lambda, \mathcal{R})\\ \mathsf{vrs} \leftarrow\!\!\$\ \mathsf{Obf}(1^\lambda, C^{\mathsf{Verify}}_{\mathsf{vrs}^*})\end{array}\right] - \right. \tag{10}$$

$$\mathbb{P}\Big[\mathsf{D}(1^\lambda, \mathsf{crs}, \mathsf{vrs}) = 1 \Big| (\mathsf{crs}, \mathsf{vrs}) \leftarrow\!\!\$\ \mathsf{Setup}(1^\lambda, \mathcal{R})\Big]\Big| \leq \mathsf{negl}(\lambda). \tag{11}$$

Proof. By contradiction, assume there exists a PPT adversary D that distinguishes the above two distributions with non-neglibile advantage. We build an adversary D' that breaks the indistinguishability property of Π^* (Definition A.3). D' proceeds as follows:

1. Receive $(\mathsf{crs}^*, \mathsf{vrs}^*)$ from the challenger.
2. Send $(\mathsf{crs}^*, \mathsf{vrs})$ to D where $\mathsf{vrs} \leftarrow\!\!\$\ \mathsf{Obf}(1^\lambda, C^{\mathsf{Verify}}_{\mathsf{vrs}^*})$.
3. Return the output of D.

Observe that if the challenger generates $(\mathsf{crs}^*, \mathsf{vrs}^*)$ by executing Ext^*_0 then D simulates the distribution of Equation (10); on the other hand, if $(\mathsf{crs}^*, \mathsf{vrs}^*)$ are generated by executing Setup^* then D' simulates the distribution of Equation (11). Hence, D' has the same advantage of D. This concludes the proof. □

We now prove that Π satisfies (publicly verifiable) straight-line knowledge soundness Definitions A.3 and A.4.

Lemma B.5. *For every PPT adversary* A*, we have:*

$$\mathbb{P}\left[(x, \omega) \notin \mathcal{R} \wedge \mathsf{Verify}(\mathsf{vrs}, x, \pi) = 1 \middle| \begin{array}{rl}(\mathsf{crs}, \mathsf{vrs}, \mathsf{td}) & \leftarrow\!\!\$\ \mathsf{Ext}_0(1^\lambda, \mathcal{R})\\ (x, \pi) & \leftarrow\!\!\$\ \mathsf{A}(1^\lambda, \mathsf{crs}, \mathsf{vrs})\\ \omega & \leftarrow\!\!\$\ \mathsf{Ext}_1(1^\lambda, \mathsf{td}, x, \pi)\end{array}\right] = 0.$$

Proof. Observe that $\mathsf{Verify}(\mathsf{vrs}, x, \pi) = 1$ if $C^{\mathsf{Verify}}_{\mathsf{vrs}^*,\mathsf{td}^*,r_1^*}(x, \pi) = 1$ where $r_1^* \leftarrow\!\!\$ \{0,1\}^*$, $(\mathsf{crs}^*, \mathsf{vrs}^*, \mathsf{td}^*) \leftarrow\!\!\$ \mathsf{Ext}_0^*(1^\lambda, \mathcal{R})$, $\mathsf{td} = (\mathsf{td}^*, r_1^*)$, $\mathsf{vrs} \leftarrow\!\!\$ \mathsf{Obf}(1^\lambda, C^{\mathsf{Verify}}_{\mathsf{vrs}^*,\mathsf{td}^*,r_1^*})$. In turn, the circuit $C^{\mathsf{Verify}}_{\mathsf{vrs}^*,\mathsf{td}^*,r_1^*}(x, \pi)$ outputs 1 only if $\mathsf{Verify}^*(\mathsf{vrs}^*, x, \pi) = 1$ and $(x, \omega) \in \mathcal{R}$ where $\omega \leftarrow\!\!\$ \mathsf{Ext}_1^*(1^\lambda, \mathsf{td}^*, x, \pi; r_1^*)$ (recall that $\mathsf{Ext}_1^*(1^\lambda, \mathsf{td}^*, x, \pi; r_1^*)$ $= \mathsf{Ext}_1(1^\lambda, \mathsf{td}, x, \pi; r)$ for every $r \in \{0,1\}^*$). Hence, A's advantage is 0. This concludes the proof. □

By combining Lemmas B.3 and B.4 we conclude that Π satisfies the indistinguishability property of Definition A.3. Moreover, Lemma B.5 implies that Π satisfies the extraction property (Definitions A.3 and A.4).

B.4 Proof of Theorem 5.4

(Part one) $\mathsf{S}_\mathcal{Y}$ is an odiO-sampler. By contradiction, suppose there exists a $q \in \mathbb{N}$, $\mathcal{Y} \subset \mathcal{M}$ such that $|\mathcal{Y}| = q$ and $\mathsf{S}_\mathcal{Y}$ is not an odiO-sampler, i.e., there exists a PPT adversary A such that

$$\mathbb{P}\Big[C_0(v) \neq C_1(v) \Big| v \leftarrow\!\!\$ \mathsf{A}^{C_0(\cdot), C_1(\cdot)}(1^\lambda, 1^{|C_0|}, \alpha)\Big] \geq \epsilon,$$

where $(C_0, C_1, \alpha) \leftarrow\!\!\$ \mathsf{S}_\mathcal{Y}(1^\lambda)$ and ϵ non-negligible. We build an adversary A' that breaks the (q)-sEUF-sel-CMA security of Π^* with respect to the messages $(m_i)_{m_i \in \mathcal{Y}} \in \mathcal{M}^q$. The adversary A' proceeds as follows:

1. Receive $(\sigma_1^*, \dots, \sigma_q^*)$ from the challenger.
2. Let $C_0^* = C^{\mathsf{Verify}}_{\mathsf{k}^*}$,$C_1^* = C^{\mathsf{Verify}}_{\mathcal{X}^*}$ and $\alpha = (\sigma_1^*, \dots, \sigma_q^*)$ where $\mathcal{X}^* = \{(m_i, \sigma_i^*)\}_{i \in [q]}$ (note that C_0^* is unknown to A' since k^* is kept secret by the challenger).
3. Send α and 1^γ (where γ as defined in Figure 3) to A and answer the incoming queries as follows:
   (a) On input (m, σ) for the circuit C_i^* for $i \in \{0,1\}$, if $(m, \sigma) \in \mathcal{X}^*$ returns 1. Otherwise, return 0.
4. Receive $v = (\widehat{m}, \widehat{\sigma})$ from A.
5. Sample a random bit $b \leftarrow\!\!\$ \{0,1\}$. If $b = 0$, A' returns $(m', \sigma') \leftarrow\!\!\$ \mathcal{Q}_{C_1^*}$ where $\mathcal{Q}_{C_1^*}$ are the queries submitted by A to the oracle C_1^*. Otherwise, if $b = 1$, A' returns $(m', \sigma') = (\widehat{m}, \widehat{\sigma})$.

Note that (C_0^*, C_1^*, α) comes from a distribution that is identical to that of $\mathsf{S}_\mathcal{Y}$; this because $\mathsf{k}^* \leftarrow\!\!\$ \mathsf{KGen}(1^\lambda, 1^q)$ and $\sigma_i^* \leftarrow\!\!\$ \mathsf{Tag}^*(\mathsf{k}^*, m_i)$ for $i \in [q]$.

We now demonstrate the following two points:

1. If A' correctly simulates A's view with respect to (C_0^*, C_1^*, α) then $(\widehat{m}, \widehat{\sigma})$ (output by A) contradicts the (q)-sEUF-sel-CMA security of Π^*.
2. On the other hand, if A' fails to correctly simulate A's view with respect to (C_0^*, C_1^*, α) then there exists $(x', \pi') \in \mathcal{Q}_{C_1^*}$ (submitted by A) that breaks the (q)-sEUF-sel-CMA security of Π^*.

Consider the following events defined with respect to k^*:

$$\begin{aligned}
&\mathbf{Sim} : \exists (m, \sigma) \in \mathcal{Q}_{C_1^*}, \mathsf{Verify}^*(\mathsf{k}^*, m, \sigma) = 1 \wedge (m, \sigma) \notin \mathcal{X}^*, \\
&\mathbf{Win} : \mathsf{Verify}^*(\mathsf{k}^*, x', \sigma') = 1 \wedge (m', \sigma') \notin \mathcal{X}^*, \\
&\mathbf{Bit} : b = 1.
\end{aligned}$$

We can bound the advantage of A' as follows:

$$\begin{aligned}
\mathbb{P}[\mathbf{Win}] = &\ \mathbb{P}[\mathbf{Win}|\mathbf{Sim}, \mathbf{Bit}] \cdot \mathbb{P}[\mathbf{Sim}] \cdot \mathbb{P}[\mathbf{Bit}] \\
&+ \mathbb{P}[\mathbf{Win}|\mathbf{Sim}, \neg\mathbf{Bit}] \cdot \mathbb{P}[\mathbf{Sim}] \cdot \mathbb{P}[\neg\mathbf{Bit}] \\
&+ \mathbb{P}[\mathbf{Win}|\neg\mathbf{Sim}, \mathbf{Bit}] \cdot \mathbb{P}[\neg\mathbf{Sim}] \cdot \mathbb{P}[\mathbf{Bit}] \\
&+ \mathbb{P}[\mathbf{Win}|\neg\mathbf{Sim}, \neg\mathbf{Bit}] \cdot \mathbb{P}[\neg\mathbf{Sim}] \cdot \mathbb{P}[\neg\mathbf{Bit}] \\
\geq &\ \mathbb{P}[\mathbf{Win}|\neg\mathbf{Sim}, \mathbf{Bit}] \cdot \mathbb{P}[\neg\mathbf{Sim}] \cdot \mathbb{P}[\mathbf{Bit}] \\
&+ \mathbb{P}[\mathbf{Win}|\mathbf{Sim}, \neg\mathbf{Bit}] \cdot \mathbb{P}[\mathbf{Sim}] \cdot \mathbb{P}[\neg\mathbf{Bit}] \\
= &\ \mathbb{P}[\mathbf{Win}|\neg\mathbf{Sim}, \mathbf{Bit}] \cdot \frac{1 - p_{\mathbf{Sim}}}{2} + \mathbb{P}[\mathbf{Win}|\mathbf{Sim}, \neg\mathbf{Bit}] \cdot \frac{p_{\mathbf{Sim}}}{2},
\end{aligned} \tag{12}$$

for $p_{\mathbf{Sim}} = \mathbb{P}[\mathbf{Sim}]$ and $\mathbb{P}[\mathbf{Bit}] = \mathbb{P}[\neg\mathbf{Bit}] = 1/2$. A differing-input $v = (m, \sigma)$ for $C_0^* = C^{\mathsf{Verify}}_{\mathsf{k}^*}$ and $C_1^* = C^{\mathsf{Verify}}_{\mathcal{X}^*}$ needs to satisfy the condition $\mathsf{Verify}^*(\mathsf{k}^*, m, \sigma) = 1 \wedge (m, \sigma) \notin \mathcal{X}^*$. We consider two cases:

– When $\neg\mathbf{Bit}$ happens, A' outputs $(m', \sigma') \leftarrow\!\!\$\; \mathcal{Q}_{C_1^*}$. Moreover, when $\mathbf{Sim}$ happens, we are guaranteed that there exists $(m, \sigma) \in \mathcal{Q}_{C_1^*}$ such that $\mathsf{Verify}^*(\mathsf{k}^*, m, \sigma) = 1 \wedge (m, \sigma) \notin \mathcal{X}^*$. Hence, we conclude that $\mathbb{P}[\mathbf{Win}|\mathbf{Sim}, \neg\mathbf{Bit}] = 1/|\mathcal{Q}_{C_1^*}|$.
– When $\mathbf{Bit}$ happens, A' outputs $(m', \sigma') = (\widehat{m}, \widehat{\sigma})$ where $(\widehat{m}, \widehat{\sigma})$ is the final output of A. Observe that, conditioned to the event $\neg\mathbf{Sim}$, A' correctly simulates the view of A. As a consequence, A outputs a valid differing-input $v = (\widehat{m}, \widehat{\sigma})$ with non-neglibile probability. Hence, we have that $\mathbb{P}[\mathbf{Win}|\neg\mathbf{Sim}, \mathbf{Bit}] \geq \epsilon$.

By combining Equation (12) and the above conditions we conclude that

$$\mathbb{P}[\mathbf{Win}] \geq \epsilon \cdot \frac{1 - p_{\mathbf{Sim}}}{2} + \frac{1}{|\mathcal{Q}_{C_1^*}|} \cdot \frac{p_{\mathbf{Sim}}}{2} \notin \mathsf{negl}(\lambda).$$

This concludes the proof.

(Part two) Π is (q)-sEUF-sel-CMA secure. Let $q \in \mathbb{N}$ and $\mathcal{Y} \subset \mathcal{M}$ such that $|\mathcal{Y}| = q$. Consider the following hybrid experiments:

$\mathsf{Hyb}_0^{q,\mathcal{Y}}(\lambda)$**:** This is the standard (q)-sEUF-sel-CMA experiment for signatures with respect to messages $(m_i)_{m_i \in \mathcal{Y}}$ (Definition A.13).

$\mathsf{Hyb}_1^{q,\mathcal{Y}}(\lambda)$**:** Same as $\mathsf{Hyb}_0^{q,\mathcal{Y}}$, except that the challenger sets pk^* to the obfuscation of the circuit $C_{\mathcal{X}^*}^{\mathsf{Verify}}$ of Figure 3 (instead of $C_{\mathsf{k}^*}^{\mathsf{Verify}}$). Formally, the challenger computes $\mathsf{pk}^* \leftarrow\!\!\$\; \mathsf{Obf}(1^\lambda, C_{\mathcal{X}^*}^{\mathsf{Verify}})$ where $\mathcal{X}^* = \{(m_i, \sigma_i^*)\}_{i \in [q]}$, $\sigma_i^* \leftarrow\!\!\$\; \mathsf{Tag}(\mathsf{k}^*, m_i)$ for $i \in [q]$, and $\mathsf{k}^* \leftarrow\!\!\$\; \mathsf{KGen}^*(1^\lambda, 1^q)$.

Lemma B.6. *For every $q \in \mathbb{N}$, every $\mathcal{Y} \subseteq \mathcal{M}$ such that $|\mathcal{Y}| = q$, $\mathsf{Hyb}_0^{q,\mathcal{Y}}(1^\lambda) \approx_c \mathsf{Hyb}_1^{q,\mathcal{Y}}(1^\lambda)$.*

Proof. By contradiction, assume there exists $q \in \mathbb{N}$, $\mathcal{Y} \subset \mathcal{M}$ such that $|\mathcal{Y}| = q$ and $\mathsf{Hyb}_0^{q,\mathcal{Y}}(\lambda)$ and $\mathsf{Hyb}_1^{q,\mathcal{Y}}(\lambda)$ are not computationally indistinguishable, i.e., there exists a PPT distinguisher D that has a non-negligible advantage in distinguishing between $\mathsf{Hyb}_0^{q,\mathcal{Y}}(\lambda)$ and $\mathsf{Hyb}_1^{q,\mathcal{Y}}(\lambda)$. We build a distinguisher D' that breaks the indistinguishability property of Obf for the odiO-sampler $\mathsf{S}_{\mathcal{Y}}$. The distinguisher D' proceeds as follows:

1. Receive in input an obfuscated circuit $\widetilde{C}$ and α. Recall $\widetilde{C} \leftarrow\!\!\$\; \mathsf{Obf}(1^\lambda, C_b)$ and $\alpha = (\sigma_1, \ldots, \sigma_q)$ where $b \leftarrow\!\!\$\; \{0,1\}$ is the unknown challenge bit, $\sigma_i \leftarrow\!\!\$\; \mathsf{Tag}^*(\mathsf{k}^*, m_i)$ for $i \in [q]$, and $(C_0, C_1, \alpha) \leftarrow\!\!\$\; \mathsf{S}_{\mathcal{Y}}(1^\lambda)$.
2. Send $\mathsf{pk}^* = \widetilde{C}$ and $(\sigma_1, \ldots, \sigma_q)$ to D.
3. Return whatever D outputs.

It is easy to see that, if $b = 0$ then D' simulates $\mathsf{Hyb}_0^{q,\mathcal{Y}}(\lambda)$. On the other hand, if $b = 1$ then D' simulates $\mathsf{Hyb}_1^{q,\mathcal{Y}}(\lambda)$. Hence, D' retains the same non negligible advantage of D. □

Observe that, for every $q \in \mathsf{poly}(\lambda)$, every $\mathcal{Y} \subseteq \mathcal{M}$ such that $|\mathcal{Y}| = q$, A has advantage 0 in $\mathsf{Hyb}_1^{q,\mathcal{Y}}$. This because, for every (m, σ), $\mathsf{Verify}(\mathsf{pk}, m, \sigma)$ returns 0 if $(m, \sigma) \notin \mathcal{X}^*$ (see the definition of $C_{\mathcal{X}^*}^{\mathsf{Verify}}$ depicted in Figure 3). This concludes the proof.

B.5 Proof of Theorem 5.5

(Part one) S_m is an odiO-sampler. Let $m^* \in \mathcal{M}$. Consider the following hybrid experiments:

$\mathsf{Hyb}_0^{m^*}(\lambda)$**:** This is the experiment oracle-differing-input experiment with respect to sampler S_{m^*} (Definition 4.1).

$\mathsf{Hyb}_1^{m^*}(\lambda)$**:** Same as $\mathsf{Hyb}_0^{m^*}(\lambda)$, except that S_{m^*} is replaced with a sampler $\widehat{\mathsf{S}}_{m^*}$ that computes C_0 differently. Formally, $\widehat{\mathsf{S}}_{m^*}$ is defined as follows:

$C_{\mathsf{s},m^*,\mathsf{k}^*}^{\mathsf{Verify}}(m, \sigma)$	$\widehat{\mathsf{S}}_{m^*}(1^\lambda; r)$
If $m = m^*$, **return** $b = \mathsf{Verify}_0^*(\mathsf{k}^*, m^*, \sigma)$	Let $r = (r_0, r_1)$
$\mathsf{k} = \mathsf{KGen}_0^*(1^\lambda; \mathsf{F}_1^*(\mathsf{s}, m))$	$\mathsf{s} = \mathsf{Gen}_1^*(1^\lambda; r_0)$
return $b = \mathsf{Verify}_0^*(\mathsf{k}, m, \sigma)$	$\mathsf{s}' = \mathsf{Punct}_1^*(\mathsf{s}, m^*)$
	$\mathsf{k}^* = \mathsf{KGen}_0^*(1^\lambda; r_1)$
	Set $C_0 = C_{\mathsf{s}',m^*,\mathsf{k}^*}^{\mathsf{Verify}}$, $C_1 = C_{\mathsf{s}',m^*}^{\mathsf{Verify}}$, $\alpha = \mathsf{s}'$
	return (C_0, C_1, α)

where $C_1 = C^{\mathsf{Verify}}_{\mathsf{s}',m^*}$ is depicted in Figure 4. $C^{\mathsf{Verify}}_{\mathsf{s}',m^*,\mathsf{k}^*}$ and $C^{\mathsf{Verify}}_{\mathsf{s}',m^*}$ are padded to match the size γ as defined in Figure 4. Observe that the distribution of (C_1, α) output by S_{m^*} and $\widehat{\mathsf{S}}_{m^*}$ are identically distributed.

Lemma B.7. *For every $m^* \in \mathcal{M}$, $\mathsf{Hyb}_0^{m^*}(\lambda) \approx_c \mathsf{Hyb}_1^{m^*}(\lambda)$.*

Proof. The lemma follows by leveraging the security and correctness of the puncturable PRF scheme Π_1^*. □

Lemma B.8. *For every $m^* \in \mathcal{M}$, $\mathbb{P}\Big[\mathsf{Hyb}_1^{m^*}(\lambda) = 1\Big] \le \mathsf{negl}(\lambda)$.*

Proof. By contradiction, suppose there exists a message $m^* \in \mathcal{M}$ such that $\widehat{\mathsf{S}}_{m^*}$ is not an odiO-sampler, i.e., there exists a PPT adversary A such that

$$\mathbb{P}\Big[\mathsf{Hyb}_1^{m^*}(\lambda) = 1\Big] = \mathbb{P}\Big[C_0(v) \neq C_1(v) \Big| v \leftarrow\!\$\; \mathsf{A}^{C_0(\cdot), C_1(\cdot)}(1^\lambda, 1^{|C_0|}, \alpha)\Big] \ge \epsilon,$$

where $(C_0, C_1, \alpha) \leftarrow\!\$\; \widehat{\mathsf{S}}_{m^*}(1^\lambda)$ and ϵ non-negligible. We build an adversary A' that breaks the EUF security of Π_0^* with respect to the message $m^* \in \mathcal{M}$. The adversary A' proceeds as follows:

1. Receive m^* from the challenger.
2. Compute $\mathsf{s} \leftarrow\!\$\; \mathsf{Gen}_1^*(1^\lambda)$ and $\mathsf{s}' = \mathsf{Punct}_1^*(\mathsf{s}, m^*)$.
3. Let $C_0^* = C^{\mathsf{Verify}}_{\mathsf{s}',m^*,\mathsf{k}^*}$, $C_1 = C^{\mathsf{Verify}}_{\mathsf{s}',m^*}$ and $\alpha = \mathsf{s}'$ (note that C_0^* is unknown to A' since k^* is kept secret by the challenger).
4. Send α and 1^γ (where γ as defined in $\mathsf{Hyb}_1^{m^*}(\lambda)$) to A and answer the incoming queries as follows:
   (a) On input (m, σ) for the circuit C_i^* for $i \in \{0, 1\}$, if $m = m^*$ returns 0. Otherwise, return $\mathsf{Verify}_0^*(\mathsf{k}, m, \sigma)$ where $\mathsf{k} = \mathsf{KGen}_0^*(1^\lambda; \mathsf{F}_1^*(\mathsf{s}', m))$.
5. Receive $v = (\widehat{m}, \widehat{\sigma})$ from A.
6. Sample a random bit $b \leftarrow\!\$\; \{0, 1\}$. If $b = 0$, A' returns $(m', \sigma') \leftarrow\!\$\; \mathcal{Q}_{C_0^*}$ where $\mathcal{Q}_{C_0^*}$ are the queries submitted by A to the oracle C_0^*. Otherwise, if $b = 1$, A' returns $(m', \sigma') = (\widehat{m}, \widehat{\sigma})$.

Note that (C_0^*, C_1^*, α) comes from a distribution that is identical to that of $\widehat{\mathsf{S}}_{m^*}$; this because k^* (generated by the challenger) is generated by executing KGen_0^* on uniform random coins.

We now demonstrate the following two points:

1. If A' correctly simulates A's view with respect to (C_0^*, C_1^*, α) then $(\widehat{m}, \widehat{\sigma})$ (output by A) contradicts the EUF security of Π_0^*.
2. On the other hand, if A' fails to correctly simulate A's view with respect to (C_0^*, C_1^*, α) then there exists $(x', \pi') \in \mathcal{Q}_{C_0^*}$ (submitted by A) that contradicts the EUF security of Π_0^*.

Consider the following events defined with respect to k^*:

$$\begin{aligned}
&\mathbf{Sim} : \exists (m, \sigma) \in \mathcal{Q}_{C_0^*}, \mathsf{Verify}_0^*(\mathsf{k}^*, m, \sigma) = 1 \land m = m^*, \\
&\mathbf{Win} : \mathsf{Verify}_0^*(\mathsf{k}^*, m', \sigma') = 1 \land m = m^*, \\
&\mathbf{Bit} : b = 1.
\end{aligned}$$

We can bound the advantage of A' as follows:

$$\begin{aligned}
\mathbb{P}[\mathbf{Win}] &= \mathbb{P}[\mathbf{Win}|\mathbf{Sim}, \mathbf{Bit}] \cdot \mathbb{P}[\mathbf{Sim}] \cdot \mathbb{P}[\mathbf{Bit}] \\
&\quad + \mathbb{P}[\mathbf{Win}|\mathbf{Sim}, \neg\mathbf{Bit}] \cdot \mathbb{P}[\mathbf{Sim}] \cdot \mathbb{P}[\neg\mathbf{Bit}] \\
&\quad + \mathbb{P}[\mathbf{Win}|\neg\mathbf{Sim}, \mathbf{Bit}] \cdot \mathbb{P}[\neg\mathbf{Sim}] \cdot \mathbb{P}[\mathbf{Bit}] \\
&\quad + \mathbb{P}[\mathbf{Win}|\neg\mathbf{Sim}, \neg\mathbf{Bit}] \cdot \mathbb{P}[\neg\mathbf{Sim}] \cdot \mathbb{P}[\neg\mathbf{Bit}] \\
&\ge \mathbb{P}[\mathbf{Win}|\neg\mathbf{Sim}, \mathbf{Bit}] \cdot \mathbb{P}[\neg\mathbf{Sim}] \cdot \mathbb{P}[\mathbf{Bit}] \\
&\quad + \mathbb{P}[\mathbf{Win}|\mathbf{Sim}, \neg\mathbf{Bit}] \cdot \mathbb{P}[\mathbf{Sim}] \cdot \mathbb{P}[\neg\mathbf{Bit}] \\
&= \mathbb{P}[\mathbf{Win}|\neg\mathbf{Sim}, \mathbf{Bit}] \cdot \frac{1 - p_{\mathbf{Sim}}}{2} + \mathbb{P}[\mathbf{Win}|\mathbf{Sim}, \neg\mathbf{Bit}] \cdot \frac{p_{\mathbf{Sim}}}{2},
\end{aligned} \tag{13}$$

for $p_{\mathbf{Sim}} = \mathbb{P}[\mathbf{Sim}]$ and $\mathbb{P}[\mathbf{Bit}] = \mathbb{P}[\neg\mathbf{Bit}] = 1/2$. A differing-input $v = (m, \sigma)$ for $C_0^* = C^{\mathsf{Verify}}_{\mathsf{s}',m^*,\mathsf{k}^*}$ and $C_1^* = C^{\mathsf{Verify}}_{\mathsf{s}',m^*}$ needs to satisfy the condition $\mathsf{Verify}^*(\mathsf{k}^*, m, \sigma) = 1 \land m = m^*$. We consider two cases:

– When $\neg\mathbf{Bit}$ happens, A' outputs $(m', \sigma') \leftarrow\!\!\$\ \mathcal{Q}_{C_0^*}$. Moreover, when $\mathbf{Sim}$ happens, we are guaranteed that there exists $(m, \sigma) \in \mathcal{Q}_{C_0^*}$ such that $\mathsf{Verify}_0^*(\mathsf{k}^*, m, \sigma) = 1 \wedge m = m^*$. Hence, we conclude that $\mathbb{P}[\mathbf{Win}|\mathbf{Sim}, \neg\mathbf{Bit}] = 1/|\mathcal{Q}_{C_0^*}|$.
– When $\mathbf{Bit}$ happens, A' outputs $(m', \sigma') = (\widehat{m}, \widehat{\sigma})$ where $(\widehat{m}, \widehat{\sigma})$ is the final output of A. Observe that, conditioned to the event $\neg\mathbf{Sim}$, A' correctly simulates the view of A. As a consequence, A outputs a valid differing-input $v = (\widehat{m}, \widehat{\sigma})$ (i.e., $\mathsf{Verify}_0^*(\mathsf{k}^*, \widehat{m}, \widehat{\sigma}) = 1 \wedge \widehat{m} = m^*$) with non-neglibile probability. Hence, we have that $\mathbb{P}[\mathbf{Win}|\neg\mathbf{Sim}, \mathbf{Bit}] \geq \epsilon$.

By combining Equation (13) and the above conditions we conclude that

$$\mathbb{P}[\mathbf{Win}] \geq \epsilon \cdot \frac{1 - p_{\mathbf{Sim}}}{2} + \frac{1}{|\mathcal{Q}_{C_0^*}|} \cdot \frac{p_{\mathbf{Sim}}}{2} \notin \mathsf{negl}(\lambda)$$

This concludes the proof. □

By combining Lemmas B.7 and B.8, we conclude that S_{m^*} is an odiO-sampler.

(Part two) Π is sel-EUF-CMA secure. Let $m^* \in \mathcal{M}$. Consider the following hybrid experiments:

$\mathsf{Hyb}_0^{m^*}(\lambda)$**:** This is the standard sel-EUF-CMA experiment for signatures with respect to message m^* (Definition A.14).

$\mathsf{Hyb}_1^{m^*}(\lambda)$**:** Same as $\mathsf{Hyb}_0^{m^*}$, except that the challenger sets pk^* to the obfuscation of the circuit $C_{\mathsf{s}',m^*}^{\mathsf{Verify}}$ of Figure 4 (instead of $C_{\mathsf{s}}^{\mathsf{Verify}}$). Formally, the challenger computes $\mathsf{pk}^* \leftarrow\!\!\$\ \mathsf{Obf}(1^\lambda, C_{\mathsf{s}',m^*}^{\mathsf{Verify}})$ where $\mathsf{s} \leftarrow\!\!\$\ \mathsf{Gen}_1^*(1^\lambda)$ and $\mathsf{s}' = \mathsf{Punct}_1^*(\mathsf{s}, m^*)$.

Lemma B.9. *For every $m^* \in \mathcal{M}$, $\mathsf{Hyb}_0^{m^*}(1^\lambda) \approx_c \mathsf{Hyb}_1^{m^*}(1^\lambda)$.*

Proof. By contradiction, assume there exists a message $m^* \in \mathcal{M}$ such that $\mathsf{Hyb}_0^{m^*}(\lambda)$ and $\mathsf{Hyb}_1^{m^*}(\lambda)$ are not computationally indistinguishable, i.e., there exists a PPT distinguisher D that has a non-negligible advantage in distinguishing between $\mathsf{Hyb}_0^{m^*}(\lambda)$ and $\mathsf{Hyb}_1^{m^*}(\lambda)$. We build a distinguisher D' that breaks the indistinguishability property of Obf for the odiO-sampler S_{m^*}. The distinguisher D' proceeds as follows:

1. Receive in input an obfuscated circuit $\widetilde{C}$ and α. Recall $\widetilde{C} \leftarrow\!\!\$\ \mathsf{Obf}(1^\lambda, C_b)$ and $\alpha = \mathsf{s}'$ where $b \leftarrow\!\!\$\ \{0,1\}$ is the unknown challenge bit and $(C_0, C_1, \alpha) \leftarrow\!\!\$\ \mathsf{S}_{m^*}(1^\lambda)$.
2. Send $\mathsf{pk}^* = \widetilde{C}$ and m^* to D and answer the incoming queries as follows:
   (a) On input m for Sign, return $\mathsf{Tag}_0^*(\mathsf{k}, m)$ where $\mathsf{k} = \mathsf{KGen}_0^*(1^\lambda; \mathsf{F}_1^*(\mathsf{s}', m))$.
3. Return whatever D outputs.

In order to be valid, D cannot submit the message m^* to the oracle Sign. By leveraging this fact and the correctness of the puncturable PRF scheme Π_1^*, D's view is correctly simulated. In particular, if $b = 0$ then D' simulates $\mathsf{Hyb}_0^{m^*}(\lambda)$. On the other hand, if $b = 1$ then D' simulates $\mathsf{Hyb}_1^{m^*}(\lambda)$. Hence, D' retains the same non negligible advantage of D. □

Observe that, for every $m^* \in \mathcal{M}$, A has advantage 0 in $\mathsf{Hyb}_1^{m^*}(\lambda)$. This is because, for every σ, $\mathsf{Verify}(\mathsf{pk}, m^*, \sigma)$ returns 0 (see definition of $C_{\mathsf{s}',m^*}^{\mathsf{Verify}}$ depicted in Figure 4). This concludes the proof.

B.6 Proof of Theorem 5.6

(Part one) S_m is an odiO-sampler. Let $m^* \in \mathcal{M}$. Consider the following hybrid experiments:

$\mathsf{Hyb}_0^{m^*}(\lambda)$**:** This is the experiment oracle-differing-input experiment with respect to sampler S_{m^*} (Definition 4.1).

$\mathsf{Hyb}_1^{m^*}(\lambda)$**:** Same as $\mathsf{Hyb}_0^{m^*}(\lambda)$, except that S_{m^*} is replaced with a sampler $\widehat{\mathsf{S}}_{m^*}$ that computes C_0 differently. Formally, $\widehat{\mathsf{S}}_{m^*}$ is defined as follows:

$$\widehat{\mathsf{S}}_{m^*}(1^\lambda; r)$$

Let $r = (r_0, r_1, r_2, r_3)$

$\mathsf{s}_1 = \mathsf{Gen}_1^*(1^\lambda; r_0),\ \mathsf{s}_2 = \mathsf{Gen}_2^*(1^\lambda; r_1)$

$\mathsf{iv} = r_3$

$\mathsf{k} = \mathsf{KGen}_0^*(1^\lambda; \mathsf{F}_2^*(\mathsf{s}_2, \mathsf{iv}))$

$c = \mathsf{Enc}_0^*(\mathsf{k}, m; \mathsf{iv})$

$\mathsf{s}_1' = \mathsf{Punct}_1^*(\mathsf{s}_1, r_2),\ \mathsf{s}_2' = \mathsf{Punct}_2^*(\mathsf{s}_2, \mathsf{iv})$

Set $C_0 = C^{\mathsf{Enc}}_{\mathsf{s}_1',\mathsf{s}_2,r_2}, C_1 = C^{\mathsf{Enc}}_{\mathsf{s}_1',\mathsf{s}_2',r_2}, \alpha = c$

return (C_0, C_1, α)

where $C^{\mathsf{Enc}}_{\mathsf{s}_1',\mathsf{s}_2,r_2}$ and $C^{\mathsf{Enc}}_{\mathsf{s}_1',\mathsf{s}_2',r_2}$ are defined as in Figure 5. $C^{\mathsf{Enc}}_{\mathsf{s}_1',\mathsf{s}_2,r_2}$ and $C^{\mathsf{Enc}}_{\mathsf{s}_1',\mathsf{s}_2',r_2}$ are padded to match the size γ as defined in Figure 5.

$\mathsf{Hyb}_2^{m^*}(\lambda)$**:** Same as $\mathsf{Hyb}_1^{m^*}(\lambda)$, except that $\widehat{\mathsf{S}}_{m^*}$ is replaced with a sampler $\bar{\mathsf{S}}_{m^*}$ that computes C_0 differently. Formally, $\bar{\mathsf{S}}_{m^*}$ is defined as follows:

$$\bar{\mathsf{S}}_{m^*}(1^\lambda; r)$$

Let $r = (r_0, r_1, r_2, r_3, r_4)$

$\mathsf{s}_1 = \mathsf{Gen}_1^*(1^\lambda; r_0),\ \mathsf{s}_2 = \mathsf{Gen}_2^*(1^\lambda; r_1)$

$\mathsf{iv} = r_3$

$\mathsf{k} = \mathsf{KGen}_0^*(1^\lambda; r_4)$

$c = \mathsf{Enc}_0^*(\mathsf{k}, m; \mathsf{iv})$

$\mathsf{s}_1' = \mathsf{Punct}_1^*(\mathsf{s}_1, r_2),\ \mathsf{s}_2' = \mathsf{Punct}_2^*(\mathsf{s}_2, \mathsf{iv})$

Set $C_0 = C_1 = C^{\mathsf{Enc}}_{\mathsf{s}_1',\mathsf{s}_2',r_2}, \alpha = c$

return (C_0, C_1, α)

where $C^{\mathsf{Enc}}_{\mathsf{s}_1',\mathsf{s}_2',r_2}$ is depicted in Figure 5. $C^{\mathsf{Enc}}_{\mathsf{s}_1',\mathsf{s}_2',r_2}$ is padded to match the size γ as defined in Figure 5.

Lemma B.10. *For every* $m^* \in \mathcal{M}$, $\mathsf{Hyb}_0^{m^*}(\lambda) \approx_c \mathsf{Hyb}_1^{m^*}(\lambda)$.

Proof. The lemma follows by leveraging the security and correctness of the puncturable PRF scheme Π_1^* and the fact that r_2 is sampled at random, i.e., the adversary cannot guess the punctured point r_2 (that is also a differing-input) except with negligible probability. □

Lemma B.11. *For every* $m^* \in \mathcal{M}$, $\mathsf{Hyb}_1^{m^*}(\lambda) \approx_c \mathsf{Hyb}_2^{m^*}(\lambda)$.

Proof. The lemma follows by leveraging the security and correctness of the puncturable PRF scheme Π_2^* and the fact that r_2 is sampled at random, i.e., the adversary cannot guess the punctured point r_2 (that is also a differing-input) except with negligible probability. □

By combining Lemmas B.10 and B.11 and observing that in $\mathsf{Hyb}_2^{m^*}(\lambda)$ the sampler $\bar{\mathsf{S}}_{m^*}$ outputs two identical circuits, we conclude that S_{m^*} is an odiO-sampler.

(Part two) $\boldsymbol{\Pi}$ **is sel-IND-CPA secure.** Let $m_0^*, m_1^* \in \mathcal{M}$. Consider the following hybrid experiments:

$\mathsf{Hyb}_0^{m_0^*,m_1^*,b}(\lambda)$**:** This is the standard sel-IND-CPA experiment for PKE with respect to messages m_0^*, m_1^* and the challenge bit b (Definition A.22). In particular, the challenge ciphertext c^* is computed as $c^* = \widetilde{C}(m_b^*, r^*)$ where $r^* \leftarrow_\$ \{0,1\}^*$ and $\mathsf{pk} = \widetilde{C}$.

$\mathsf{Hyb}_1^{m_0^*,m_1^*,b}(\lambda)$**:** Same as $\mathsf{Hyb}_0^{m_0^*,m_1^*,b}$, except that the challenger sets pk^* to the obfuscation of the circuit $C^{\mathsf{Enc}}_{\mathsf{s}_1',\mathsf{s}_2',r^*}$ of Figure 5 (instead of $C^{\mathsf{Enc}}_{\mathsf{s}_1,\mathsf{s}_2}$) where $\mathsf{s}_1 \leftarrow_\$ \mathsf{Gen}_1^*(1^\lambda)$, $\mathsf{s}_2 \leftarrow_\$ \mathsf{Gen}_2^*(1^\lambda)$, $r^* \leftarrow_\$ \{0,1\}^*$, $\mathsf{s}_1' = \mathsf{Punct}_1^*(\mathsf{s}_1, r^*)$, $\mathsf{s}_2' = \mathsf{Punct}_2^*(\mathsf{s}_2, \mathsf{F}_1^*(\mathsf{s}_1, r^*))$. Recall that r^* is the randomness of the challenge ciphertext c^*. Moreover, the challenge ciphertext c^* is computed as $c^* = \mathsf{Enc}_0^*(\mathsf{k}, m_b^*, \mathsf{iv})$ where $\mathsf{iv} = \mathsf{F}_1^*(\mathsf{s}_1, r^*)$, $\mathsf{k} = \mathsf{KGen}_0^*(1^\lambda; \mathsf{F}_2^*(\mathsf{s}_2, \mathsf{iv}))$. Therefore, c^* is computed as in $\mathsf{Hyb}_0^{m_0^*,m_1^*,b}(\lambda)$.

$\mathsf{Hyb}_2^{m_0^*,m_1^*,b}(\lambda)$**:** Same as $\mathsf{Hyb}_1^{m_0^*,m_1^*,b}$, except that the challenger changes how it computes the challenge ciphertext c^*. First, the challenger computes pk^* as in $\mathsf{Hyb}_1^{m_0^*,m_1^*,b}(\lambda)$ and then it computes $c^* = \mathsf{Enc}_0^*(\mathsf{k}, m_b^*; \mathsf{iv})$ where $\mathsf{iv} \leftarrow\$ \{0,1\}^*$ and $\mathsf{k} = \mathsf{KGen}_0^*(1^\lambda; \mathsf{F}_2^*(\mathsf{s}_2, \mathsf{iv}))$.

$\mathsf{Hyb}_3^{m_0^*,m_1^*,b}(\lambda)$**:** Same as $\mathsf{Hyb}_2^{m_0^*,m_1^*,b}$, except that the challenger changes how it computes the challenge ciphertext c^*. First, the challenger computes pk^* as in $\mathsf{Hyb}_2^{m_0^*,m_1^*,b}(\lambda)$ and then it computes $c^* = \mathsf{Enc}_0^*(\mathsf{k}, m_b^*; \mathsf{iv})$ where $\mathsf{iv} \leftarrow\$ \{0,1\}^*$ and $\mathsf{k} \leftarrow\$ \mathsf{KGen}_0^*(1^\lambda)$.

Lemma B.12. *For every* $m_0^*, m_1^* \in \mathcal{M}$, $\mathsf{Hyb}_0^{m_0^*,m_1^*,b}(1^\lambda) \approx_c \mathsf{Hyb}_1^{m_0^*,m_1^*,b}(1^\lambda)$.

Proof. By contradiction, assume there exist $m_0^*, m_1^* \in \mathcal{M}$ such that $\mathsf{Hyb}_0^{m_0^*,m_1^*,b}(\lambda)$ and $\mathsf{Hyb}_1^{m_0^*,m_1^*,b}(\lambda)$ are not computationally indistinguishable, i.e., there exists a PPT distinguisher D that has a non-negligible advantage in distinguishing between $\mathsf{Hyb}_0^{m_0^*,m_1^*,b}(\lambda)$ and $\mathsf{Hyb}_1^{m_0^*,m_1^*,b}(\lambda)$. We build a distinguisher D' that breaks the indistinguishability property of Obf for the odiO-sampler $\mathsf{S}_{m_b^*}$. The distinguisher D' proceeds as follows:

1. Receive in input an obfuscated circuit $\widetilde{C}$ and α. Recall $\widetilde{C} \leftarrow\$ \mathsf{Obf}(1^\lambda, C_d)$ and $\alpha = c$ where $d \leftarrow\$ \{0,1\}$ is the unknown challenge bit and $(C_0, C_1, \alpha) \leftarrow\$ \mathsf{S}_{m_b^*}(1^\lambda)$.
2. Send $\mathsf{pk} = \widetilde{C}$ and c to D.
3. Return whatever D outputs.

If $d = 0$ then D' simulates $\mathsf{Hyb}_0^{m_0^*,m_1^*,b}(\lambda)$. On the other hand, if $b = 1$ then D' simulates $\mathsf{Hyb}_1^{m_0^*,m_1^*,b}(\lambda)$ Hence, D' retains the same non-negligible advantage of D. ◻

Lemma B.13. *For every* $m_0^*, m_1^* \in \mathcal{M}$, $\mathsf{Hyb}_1^{m_0^*,m_1^*,b}(1^\lambda) \approx_c \mathsf{Hyb}_2^{m_0^*,m_1^*,b}(1^\lambda)$.

Proof. The lemma follows by leveraging the security and correctness of the puncturable PRF scheme Π_1^*. ◻

Lemma B.14. *For every* $m_0^*, m_1^* \in \mathcal{M}$, $\mathsf{Hyb}_2^{m_0^*,m_1^*,b}(1^\lambda) \approx_c \mathsf{Hyb}_3^{m_0^*,m_1^*,b}(1^\lambda)$.

Proof. The lemma follows by leveraging the security and correctness of the puncturable PRF scheme Π_2^*. ◻

Lemma B.15. *For every* $m_0^*, m_1^* \in \mathcal{M}$, $\mathsf{Hyb}_3^{m_0^*,m_1^*,b}(1^\lambda) \approx_c \mathsf{Hyb}_3^{m_0^*,m_1^*,1-b}(1^\lambda)$.

Proof. By contradiction, assume there exist $m_0^*, m_1^* \in \mathcal{M}$ such that $\mathsf{Hyb}_3^{m_0^*,m_1^*,b}(\lambda)$ and $\mathsf{Hyb}_3^{m_0^*,m_1^*,1-b}(\lambda)$ are not computationally indistinguishable, i.e., there exists a PPT distinguisher D that has a non-negligible advantage in distinguishing between $\mathsf{Hyb}_3^{m_0^*,m_1^*,b}(\lambda)$ and $\mathsf{Hyb}_3^{m_0^*,m_1^*,1-b}(\lambda)$. We build an adversary A that breaks the semantic security of Π_0^* with respect to messages $m_0^*, m_1^* \in \mathcal{M}$. The distinguisher A proceeds as follows:

1. Receive $c^* = (\mathsf{iv}, c)$ from the challenger.
2. Compute $\mathsf{pk} \leftarrow\$ \mathsf{Obf}(1^\lambda, C'^{\mathsf{Enc}}_{\mathsf{s}_1',\mathsf{s}_2'})$ where $\mathsf{s}_1 \leftarrow\$ \mathsf{Gen}_1^*(1^\lambda)$, $\mathsf{s}_2 \leftarrow\$ \mathsf{Gen}_2^*(1^\lambda)$, $r^* \leftarrow\$ \{0,1\}^*$, $\mathsf{s}_1' = \mathsf{Punct}_1^*(\mathsf{s}_1, r^*)$, and $\mathsf{s}_2' = \mathsf{Punct}_2^*(\mathsf{s}_2, \mathsf{iv})$.
3. Send pk and c^* to D.
4. Return whatever D outputs.

Observe that A simulates $\mathsf{Hyb}_3^{m_0^*,m_1^*,b}(\lambda)$ where $b \in \{0,1\}$ is the challenge bit sampled by the challenger. Hence, A retains the same non-negligible advantage of D. ◻

By combining Lemmas B.12 to B.15, we conclude that Π is sel-IND-CPA.

B.7 Proof of Theorem 5.7

(Part one) S_m is an oiO-sampler. By contradiction, suppose there exists $m^* \in \mathcal{M}$ such that S_{m^*} is not an oiO-sampler, i.e., there exists a PPT adversary D such that

$$\left|\mathbb{P}\left[\mathsf{D}^{C_0(\cdot)}(1^\lambda, 1^{|C_0|}, \alpha) = 1\right] - \mathbb{P}\left[\mathsf{D}^{C_1(\cdot)}(1^\lambda, 1^{|C_0|}, \alpha) = 1\right]\right| \geq \epsilon, \tag{14}$$

where $(C_0, C_1, \alpha) \leftarrow\$ \mathsf{S}_m(1^\lambda)$ where ϵ non-negligible.

We build an adversary A that breaks the sel-IND-CPRA-key security of Π^* with respect to the message $m^* \in \mathcal{M}$. The adversary A proceeds as follows:

1. Receive c^* from the challenger.
2. Let $C_0^* = C^{\mathsf{Enc}}_{\mathsf{k}_0^*}$, $C_1 = C^{\mathsf{Enc}}_{\mathsf{k}_1^*}$, and $\alpha = c^*$ (note that both C_0^* and C_1^* are unknown to A since k_0^* and k_1^* are kept secret by the challenger).
3. Send α and 1^γ (where γ as defined in Figure 6) to D and answer to the incoming queries as follows:
   (a) On input (m, r), forward (m, r) to the oracle $\mathsf{Enc}^*(\mathsf{k}_0^*, \cdot; \cdot)$ and returns the answer.
4. Output whatever D outputs.

Note that k_0^*, k_1^*, and c^* (generated by the challenger) have the same distribution to the one generated by S_{m^*}. Moreover,

1. if $b = 0$ (the challenge bit sampled by the challenger), A simulates the left distribution of Equation (14). This because c^* is encrypted using the same key (i.e., k_0^*) hardcoded in the oracle C_0^* (see definition of S_{m^*} (Figure 6)) that, in turn, is simulated by A using the oracle $\mathsf{Enc}^*(\mathsf{k}_0^*, \cdot; \cdot)$.
2. On the other hand, if $b = 1$, A simulates the right distribution of Equation (14), i.e., c^* is encrypted using (a random) key k_1^* that is completely independent from the one of oracle C_1^* since the latter is simulated using the oracle $\mathsf{Enc}^*(\mathsf{k}_0^*, \cdot; \cdot)$.

Hence, A breaks the sel-CPRA-key-ind security of Π^* with the same non-negligible advantage of D. This concludes the proof.

(Part two) Π is sel-IND-CPA. Let $m_0^*, m_1^* \in \mathcal{M}$. Consider the following hybrid experiments:

$\mathsf{Hyb}_0^{m_0^*, m_1^*, b}(\lambda)$**:** This is the standard sel-CPA-sec experiment (with respect to the messages $m_0^*, m_1^* \in \mathcal{M}$) for PKE (Definition A.22) where the challenge bit is b.

$\mathsf{Hyb}_1^{m_0^*, m_1^*, b}(\lambda)$**:** Same as $\mathsf{Hyb}_0^{m_0^*, m_1^*, b}$, except that the challenger computes $\mathsf{k}_0^* \leftarrow\$ \mathsf{KGen}^*(\lambda)$, $\mathsf{k}_1^* \leftarrow\$ \mathsf{KGen}^*(\lambda)$, $\mathsf{pk} \leftarrow\$ \mathsf{Obf}(1^\lambda, C^{\mathsf{Enc}}_{\mathsf{k}_1^*})$ and set the challenge ciphertext to $c^* \leftarrow\$ \mathsf{Enc}^*(\mathsf{k}_0^*, m_b^*)$.

Lemma B.16. *For every $m_0^*, m_1^* \in \mathcal{M}$, $\mathsf{Hyb}_0^{m_0^*, m_1^*, b}(1^\lambda) \approx_c \mathsf{Hyb}_1^{m_0^*, m_1^*, b}(1^\lambda)$.*

Proof. By contradiction, assume there exist $m_0^*, m_1^* \in \mathcal{M}$ such that $\mathsf{Hyb}_0^{m_0^*, m_1^*, b}(\lambda)$ and $\mathsf{Hyb}_1^{m_0^*, m_1^*, b}(\lambda)$ are not computationally indistinguishable, i.e., there exists a PPT distinguisher D that has a non-negligible advantage in distinguishing between $\mathsf{Hyb}_0^{m_0^*, m_1^*, b}(\lambda)$ and $\mathsf{Hyb}_1^{m_0^*, m_1^*, b}(\lambda)$. We build a distinguisher D′ that breaks the indistinguishability property of Obf for the oiO-sampler $\mathsf{S}_{m_b^*}$. The distinguisher D′ proceeds as follows:

1. Receive in input an obfuscated circuit $\widetilde{C}$ and α. Recall $\widetilde{C} \leftarrow\$ \mathsf{Obf}(1^\lambda, C_d)$ and $\alpha = c$ where $d \leftarrow\$ \{0, 1\}$ is the unknown challenge bit and $(C_0, C_1, \alpha) \leftarrow\$ \mathsf{S}_{m_b^*}(1^\lambda)$.
2. Send $\mathsf{pk} = \widetilde{C}$ and c to D.
3. Return whatever D outputs.

If $d = 0$ then D′ simulates $\mathsf{Hyb}_0^{m_0^*, m_1^*, b}(\lambda)$. This because $\widetilde{C}$ encodes a random key k_0^* that is the same used to compute $c \leftarrow\$ \mathsf{Enc}^*(\mathsf{k}_0^*, m_b^*)$ (see definition of $\mathsf{S}_{m_b^*}$ depicted in Figure 6). On the other hand, if $b = 1$ then D′ simulates $\mathsf{Hyb}_1^{m_0^*, m_1^*, b}(\lambda)$ since $c \leftarrow\$ \mathsf{Enc}^*(\mathsf{k}_0^*, m_b^*)$ and the key k_0^* is completely independent to the one that is encoded into $\widetilde{C}$ since $\widetilde{C} \leftarrow\$ \mathsf{Obf}(1^\lambda, C^{\mathsf{Enc}}_{\mathsf{k}_1^*})$ for $\mathsf{k}_1^* \leftarrow\$ \mathsf{KGen}^*(1^\lambda)$. Hence, D′ retains the same non-negligible advantage of D. □

$\widetilde{C}^0_{\mathsf{k}}(x,r)$	$\widetilde{C}^1_{\mathsf{k}}(i,r)$	$\widetilde{C}^2_{\mathsf{k}}(c_1,c_2,\odot,r)$	$\widetilde{C}^3_{\mathsf{k}}(d_1,\ldots,d_\lambda,r)$
return $\mathsf{Enc}_0(\mathsf{k},0;r)$	**return** $\mathsf{Enc}_0(\mathsf{k},0;r)$	**return** $\mathsf{Enc}_0(\mathsf{k},0;r)$	**return** $\mathsf{Enc}_0(\mathsf{k},0;r)$

$C^{\mathsf{rnd}}_{(\mathsf{k},a,b,\mathsf{y},e)}(\ell,v,r)$

Let $v=(x,i,c_1,c_2,\odot,d_1,\ldots,d_\lambda)$
$r'=\mathsf{F}_{\mathsf{rnd}}(\ell,v,r)$
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** $C^3_{\mathsf{k},a,b,\mathsf{y},e}(d_1,\ldots,d_\lambda,r')$

$\widehat{C}_{(\mathsf{k},a,b)}(\ell,v,r)$

Let $v=(x,i,c_1,c_2,\odot,d_1,\ldots,d_\lambda)$
$r'=\mathsf{F}_{\mathsf{rnd}}(\ell,v,r)$
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** $\widetilde{C}^3_{\mathsf{k}}(d_1,\ldots,d_\lambda,r')$

$\widetilde{C}_{(\mathsf{k},a,b)}(\ell,v,r)$

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

$\widetilde{C}_{\mathsf{k}}(\ell,v,r)$

Let $v=(x,i,c_1,c_2,\odot,d_1,\ldots,d_\lambda)$
$r'=\mathsf{F}_{\mathsf{rnd}}(\ell,v,r)$
If $\ell=0$, **return** $\widetilde{C}^0_{\mathsf{k}}(x,r')$
If $\ell=1$, **return** $\widetilde{C}^1_{\mathsf{k}}(i,r')$
If $\ell=2$, **return** $\widetilde{C}^2_{\mathsf{k}}(c_1,c_2,\odot,r')$
If $\ell=3$, **return** $\widetilde{C}^3_{\mathsf{k}}(d_1,\ldots,d_\lambda,r')$

Fig. 10: The circuits $C^{\mathsf{rnd}}_{(\mathsf{k},a,b,\mathsf{y},e)}$, $\widehat{C}_{(\mathsf{k},a,b)}$, $\widetilde{C}_{(\mathsf{k},a,b)}$, and $\widetilde{C}_{\mathsf{k}}$ where $\mathsf{F}_{\mathsf{rnd}}(\cdot,\cdot,\cdot)$ denotes an arbitrary truly random function.

Lemma B.17. *For every* $m^*_0, m^*_1 \in \mathcal{M}$, $\mathsf{Hyb}_1^{m^*_0,m^*_1,b}(1^\lambda) \approx_c \mathsf{Hyb}_1^{m^*_0,m^*_1,1-b}(1^\lambda)$.

Proof. By contradiction, assume there exist $m^*_0, m^*_1 \in \mathcal{M}$ such that $\mathsf{Hyb}_1^{m^*_0,m^*_1,b}(\lambda)$ and $\mathsf{Hyb}_1^{m^*_0,m^*_1,1-b}(\lambda)$ are not computationally indistinguishable, i.e., there exists a PPT distinguisher D that has a non-negligible advantage in distinguishing between $\mathsf{Hyb}_1^{m^*_0,m^*_1,b}(\lambda)$ and $\mathsf{Hyb}_1^{m^*_0,m^*_1,1-b}(\lambda)$. We build an adversary A that breaks the semantic security of Π^* with respect to messages $m^*_0, m^*_1 \in \mathcal{M}$. The distinguisher A proceeds as follows:

1. Receive c^* from the challenger.
2. Compute $\mathsf{pk} \leftarrow\!\!\$\ \mathsf{Obf}(1^\lambda, C^{\mathsf{Enc}}_{\mathsf{k}})$ where $\mathsf{k} \leftarrow\!\!\$\ \mathsf{KGen}^*(1^\lambda)$.
3. Send pk and c^* to D.
4. Return whatever D outputs.

A correctly simulates D's view. Indeed, A simulates $\mathsf{Hyb}_1^{m^*_0,m^*_1,b}$ where $b \in \{0,1\}$ is the challenge bit sampled by the challenger. Hence, A retains the same non-negligible advantage of D. □

By combining Lemmas B.16 and B.17 we conclude that Π is sel-IND-CPA.

B.8 Proof of Theorem 6.1

(Part one) $\mathcal{C}$ satisfies oracle-differing-input. Let $p(\cdot)$ be a polynomial in the security parameter λ. Without loss of generality, we assume that A submits $p(\lambda)$ queries to the oracles $C^*_{\mathsf{s},(\mathsf{k},a,b,\mathsf{y},0)}$ and $C^*_{\mathsf{s},(\mathsf{k},a,b,\mathsf{y},1)}$. Consider the following hybrid experiments:

$\mathsf{Hyb}_0(\lambda)$ This is the oracle-differing-input experiment of Theorem 6.1.

$\mathsf{Hyb}_1(\lambda)$**:** Same as $\mathsf{Hyb}_0(\lambda)$, except that the oracle access to $C^*_{\mathsf{s},(\mathsf{k},a,b,\mathsf{y},0)}$ and $C^*_{\mathsf{s},(\mathsf{k},a,b,\mathsf{y},1)}$ are simulated as $C^{\mathsf{rnd}}_{(\mathsf{k},a,b,\mathsf{y},0)}$ and $C^{\mathsf{rnd}}_{(\mathsf{k},a,b,\mathsf{y},1)}$ (defined in Figure 10), respectively.

$\mathsf{Hyb}^0_2(\lambda)$**:** Same as $\mathsf{Hyb}_1(\lambda)$, except that $C^{\mathsf{rnd}}_{(\mathsf{k},a,b,\mathsf{y},0)}$ and $C^{\mathsf{rnd}}_{(\mathsf{k},a,b,\mathsf{y},1)}$ are both simulated as $\widehat{C}_{(\mathsf{k},a,b)}$ (Figure 10).

$\mathsf{Hyb}^j_2(\lambda)$**:** Same as $\mathsf{Hyb}^{j-1}_2(\lambda)$, except that the challenger changes how it answers to the last j-th queries. Formally, on input the j'-th query (ℓ, v, r), if $j' \geq p(\lambda) - j + 1$, the challenger returns $\widetilde{C}_{(\mathsf{k},a,b)}(\ell, v, r)$. Otherwise (i.e., $j' < p(\lambda) - j + 1$), it returns $\widehat{C}_{(\mathsf{k},a,b)}(\ell, v, r)$.

$\mathsf{Hyb}_3(\lambda)$**:** Same as $\mathsf{Hyb}^{p(\lambda)}_2(\lambda)$, except that the two oracle circuits simulated as $\widetilde{C}_{\mathsf{k}}$, instead of being simulated as $\widetilde{C}_{(\mathsf{k},a,b)}$ (see Figure 10).

Lemma B.18. $\mathsf{Hyb}_0(\lambda) \approx_c \mathsf{Hyb}_1(\lambda)$.

Proof. The lemma follows by the security of the PRF scheme Π_1. □

Lemma B.19. *For every* $j \in [p(\lambda)]$, $\mathsf{Hyb}^j_2(\lambda) \approx_c \mathsf{Hyb}^{j-1}_2(\lambda)$.

Proof. Let (ℓ, v, r) be the j-th query of A. We have the following cases:

1. If $\ell \in \{0, 3\}$ by definition of $\widehat{C}_{(\mathsf{k},a,b)}$ and $\widetilde{C}_{(\mathsf{k},a,b)}(\ell, v, r)$ we have that $\widehat{C}_{(\mathsf{k},a,b)}(\ell, v, r) = \widetilde{C}_{(\mathsf{k},a,b)}(\ell, v, r)$. Thus, the two hybrids $\mathsf{Hyb}^j_2(\lambda)$ and $\mathsf{Hyb}^{j-1}_2(\lambda)$ are identically distributed.
2. On the other hand, if $\ell \in \{1, 2\}$, we can show that $\mathsf{Hyb}^j_2(\lambda)$ and $\mathsf{Hyb}^{j-1}_2(\lambda)$ are computationally indistinguishable by leveraging the IND-CCA1 security of Π_0. This can be done by using an identical argument to that of Barak et al. [BGI+12, Claim 3.6.1].

This concludes the proof. □

Lemma B.20. $\mathsf{Hyb}^{p(\lambda)}_2(\lambda) \approx_c \mathsf{Hyb}_3(\lambda)$.

Proof. The only way to distinguish these two hybrids is to guess the trigger input $a \in \{0,1\}^\lambda$ that happens with negligible probability. □

Lemma B.21. $\mathsf{Hyb}_1(\lambda) \approx_c \mathsf{Hyb}^0_2(\lambda)$.

Proof. Suppose there exists a PPT D that distinguishes between $\mathsf{Hyb}_1(\lambda)$ and $\mathsf{Hyb}^0_2(\lambda)$. By definition of $C^{\mathsf{rnd}}_{(\mathsf{k},a,b,\mathsf{y},e)}$ (for $e \in \{0,1\}$) and $\widehat{C}_{(\mathsf{k},a,b)}$, this implies that D submits, with non-negligible probability ϵ, a query $(3, v^*, r^*)$ such that $\widehat{C}_{(\mathsf{k},a,b)}(3, v^*, r^*) \neq C^{\mathsf{rnd}}_{(\mathsf{k},a,b,\mathsf{y},e)}(3, v^*, r^*)$ for $e \in \{0,1\}$, i.e., $v^* = (x, i, c_1, c_2, \odot, d_1, \ldots, d_\lambda)$ and $\forall i \in [\lambda], \mathsf{Dec}_0(\mathsf{k}, d_i) = b_i$. By leveraging Lemmas B.20 and B.21, we have that $\mathsf{Hyb}^0_2(\lambda) \approx_c \mathsf{Hyb}_3(\lambda)$; hence, D must the same query $(3, v^*, r^*)$ during the experiment $\mathsf{Hyb}_3(\lambda)$ with the same non-negligible probability ϵ. However, in $\mathsf{Hyb}_3(\lambda)$ any distinguisher D has a negligible advantage in guessing b since it is sampled at random and $\mathsf{Hyb}_3(\lambda)$ is defined with respect to $\widetilde{C}_{\mathsf{k}}$ that does not depend on b. As a consequence, D can not submit such a query $(3, v^*, r^*)$, except with negligible probability. This concludes the proof. □

By combining Lemmas B.18 to B.21, we conclude that the ensemble $\mathcal{C}$ satisfies the oracle-differing-input.

(Part two) $\mathcal{C}$ satisfies input-indistinguishability. Let $p_0(\cdot), p_1(\cdot)$ be two polynomials in the security parameter λ. Without loss of generality, we assume that D submits $p_d(\lambda)$ queries to the oracles $C^*_{\mathsf{s}_d,(\mathsf{k}_d,a_d,b_d,\mathsf{y}_d,d)}$ for $d \in \{0,1\}$. Consider the following hybrid experiments:

$\mathsf{Hyb}^d_0(\lambda)$ This is the input-indistinguishability experiment of Theorem 6.1 where the challenge bit is d, i.e., the adversary receives in input m_d.

$\mathsf{Hyb}^d_1(\lambda)$**:** Same as $\mathsf{Hyb}^d_0(\lambda)$, except that the oracle access to $C^*_{\mathsf{s}_0,(\mathsf{k}_0,a_0,b_0,\mathsf{y}_0,0)}$ and $C^*_{\mathsf{s}_1,(\mathsf{k}_1,a_1,b_1,\mathsf{y}_1,1)}$ are simulates as $C^{\mathsf{rnd0}}_{(\mathsf{k}_0,a_0,b_0,\mathsf{y}_0,0)}$ and $C^{\mathsf{rnd1}}_{(\mathsf{k}_1,a_1,b_1,y_1,1)}$ (depicted in Figure 10), respectively. We stress that $C^{\mathsf{rnd0}}_{(\mathsf{k}_0,a_0,b_0,\mathsf{y}_0,0)}$ and $C^{\mathsf{rnd1}}_{(\mathsf{k}_1,a_1,b_1,\mathsf{y}_1,1)}$ are simulated using two independent truly random functions $\mathsf{F}_{\mathsf{rnd0}}(\cdot,\cdot,\cdot)$ and $\mathsf{F}_{\mathsf{rnd1}}(\cdot,\cdot,\cdot)$.

$\mathsf{Hyb}_2^{d,0}(\lambda)$**:** Same as $\mathsf{Hyb}_1^d(\lambda)$, except that the oracle access to $C^{\mathsf{rnd0}}_{(\mathsf{k}_0,a_0,b_0,\mathsf{y}_0,0)}$ is simulated as $\widehat{C}_{(\mathsf{k}_0,a_0,b_0)}$ (depicted in Figure 10).

$\mathsf{Hyb}_2^{d,j}(\lambda)$**:** Same as $\mathsf{Hyb}_2^{d,j-1}(\lambda)$, except that the challenger changes how it answers to the last j-th queries of $\widehat{C}_{(\mathsf{k}_0,a_0,b_0)}$. Formally, on input the j'-th query (ℓ, v, r) for $\widehat{C}_{(\mathsf{k}_0,a_0,b_0)}$, if $j' \geq p_0(\lambda) - j + 1$, the challenger returns $\widetilde{C}_{(\mathsf{k}_0,a_0,b_0)}(\ell, v, r)$. Otherwise (i.e., $j' < p_0(\lambda) - j + 1$), it returns $\widehat{C}_{(\mathsf{k}_0,a_0,b_0)}(\ell, v, r)$.

$\mathsf{Hyb}_3^{d}(\lambda)$**:** Same as $\mathsf{Hyb}_2^{d,p_0(\lambda)}(\lambda)$, except that the oracle access to $\widehat{C}_{(\mathsf{k}_0,a_0,b_0)}$ is simulated as $\widetilde{C}_{\mathsf{k}_0}$ (see Figure 10).

$\mathsf{Hyb}_4^{d,0}(\lambda)$**:** Same as $\mathsf{Hyb}_3^d(\lambda)$, except that the oracle access to $C^{\mathsf{rnd1}}_{(\mathsf{k}_1,a_1,b_1,\mathsf{y}_1,1)}$ is simulated as $\widehat{C}_{(\mathsf{k}_1,a_1,b_1)}$ (Figure 10).

$\mathsf{Hyb}_4^{d,j}(\lambda)$**:** Same as $\mathsf{Hyb}_4^{d,j-1}(\lambda)$, except that the challenger changes how it answers to the last j-th queries of $\widehat{C}_{(\mathsf{k}_1,a_1,b_1)}$. Formally, on input the j'-th query (ℓ, v, r) for $\widehat{C}_{(\mathsf{k}_1,a_1,b_1)}$, if $j' \geq p_1(\lambda) - j + 1$, the challenger returns $\widetilde{C}_{(\mathsf{k}_1,a_1,b_1)}(\ell, v, r)$. Otherwise (i.e., $j' < p_1(\lambda) - j + 1$), it returns $\widehat{C}_{(\mathsf{k}_1,a_1,b_1)}(\ell, v, r)$.

$\mathsf{Hyb}_5^{d}(\lambda)$**:** Same as $\mathsf{Hyb}_4^{d,p_1(\lambda)}(\lambda)$, except that the oracle access to $\widehat{C}_{(\mathsf{k}_1,a_1,b_1)}$ is simulated as $\widetilde{C}_{\mathsf{k}_1}$ (see Figure 10).

Lemma B.22. $\mathsf{Hyb}_0^d(\lambda) \approx_c \mathsf{Hyb}_1^d(\lambda)$.

Proof. The lemma follows by the security of the PRF scheme Π_1. ☐

Lemma B.23. *For every* $j \in [p_0(\lambda)]$, $\mathsf{Hyb}_2^{d,j}(\lambda) \approx_c \mathsf{Hyb}_2^{d,j-1}(\lambda)$.

Proof. The lemma follows by the security of the IND-CCA1 security of Π_0 and using a similar argument to that of Lemma B.19. ☐

Lemma B.24. $\mathsf{Hyb}_2^{d,p_0(\lambda)}(\lambda) \approx_c \mathsf{Hyb}_3^d(\lambda)$.

Proof. The proof is identical to that of Lemma B.20. The only way to distinguish these two hybrids is to guess the trigger input $a_0 \in \{0,1\}^\lambda$ that happens with negligible probability. ☐

Lemma B.25. $\mathsf{Hyb}_1^d(\lambda) \approx_c \mathsf{Hyb}_2^{d,0}(\lambda)$.

Proof. The lemma follows by using a similar argument to that of Lemma B.21. ☐

Lemma B.26. *For every* $j \in [p_1(\lambda)]$, $\mathsf{Hyb}_4^{d,j}(\lambda) \approx_c \mathsf{Hyb}_4^{d,j-1}(\lambda)$.

Proof. The lemma follows by the security of the IND-CCA1 security of Π_0 and using a similar argument to that of Lemma B.19. ☐

Lemma B.27. $\mathsf{Hyb}_4^{d,p_1(\lambda)}(\lambda) \approx_c \mathsf{Hyb}_5^d(\lambda)$.

Proof. The proof is identical to that of Lemma B.20. The only way to distinguish these two hybrids is to guess the trigger input $a_1 \in \{0,1\}^\lambda$ that happens with negligible probability. ☐

Lemma B.28. $\mathsf{Hyb}_3^d(\lambda) \approx_c \mathsf{Hyb}_4^{d,0}(\lambda)$.

Proof. The lemma follows by using a similar argument to that of Lemma B.21. ☐

Lemma B.29. $\mathsf{Hyb}_5^d(\lambda) \approx_c \mathsf{Hyb}_5^{1-d}(\lambda)$.

Proof. The lemma follows by leveraging the IND-CPA-key security of Π_0. ☐

By combining Lemmas B.22 to B.29, we conclude that the ensemble $\mathcal{C}$ satisfies input-indistinguishability.

(Part three) $\mathcal{C}$ satisfies partial reversability. The reversability property follows by using an identical argument to that discussed in [BGI+01, Lemma 3.5]. Consider the following PPT algorithm $\mathsf{Ext}(1^\lambda, \widetilde{C})$:

1. Let $v_i = (\bot, i, \bot, \ldots, \bot)$.
2. For every $i \in [\lambda]$, evaluate $c_i = \widetilde{C}(1, v_i, r_i)$ where $r_i \leftarrow\!\!\$\ \{0,1\}^*$.
3. Consider the gate representation of the circuit $C(\cdot,\cdot) = \widetilde{C}(0,\cdot,\cdot)$.
4. Let $(d_1, \ldots, d_\lambda)$ be the output of the gate-by-gate computation of $C(\cdot,\cdot)$ over the ciphertexts $(c_1, \ldots, c_\lambda)$ of a (this can be accomplished by leveraging the access to $\widetilde{C}(2,\cdot,\cdot)$ that correspond to the C^2_{k} of Figure 7 that, in turn, permits to perform arbitrary homomorphic computation over encrypted inputs). Observe that d_i will be the encryption of b_i since each c_i is an encryption of a_i and C returns b if evaluated over a.
5. Compute $(a, \mathsf{k}, e, y) = \widetilde{C}(3, v, r)$ where $v = (\bot, \bot, \bot, \bot, \bot, d_1, \ldots, d_\lambda)$ and $r \leftarrow\!\!\$\ \{0,1\}^*$.
6. Compute $b = \widetilde{C}(0, v', r')$ $v' = (a, \bot, \ldots, \bot)$ and $r' \leftarrow\!\!\$\ \{0,1\}^*$.
7. Output $(\mathsf{k}, a, b, \mathsf{y}, e)$.

By leveragng both the correctness of Π_0 (Definition A.15) and the definition of $\widetilde{C}$ (Theorem 6.1), it is easy to see that Ext always output the correct $(\mathsf{k}, a, b, \mathsf{y}, e)$.

B.9 Proof of Theorem 6.3

(Part one) $\mathsf{S}_{\mathsf{owf}}$ is an odiO-sampler. Let A be a PPT adversary. The only input $x \in \{0,1\}^\lambda$ on which $C_0 = C_{r,0}, C_1 = C_{r,1}$ (output by S_b) differ is $x = r$ where $r \leftarrow\!\!\$\ \{0,1\}^\lambda$. Since $\alpha = \bot$ and A has only oracle access to C_0 and C_1 we conclude that A cannot do better than guessing r, i.e.,

$$\mathbb{P}[C_{r,0}(x) \neq C_{r,1}(x)] = \mathbb{P}[x = r] = \frac{1}{2^\lambda},$$

where $(C_{r,0}, C_{r,1}, \bot) \leftarrow\!\!\$\ \mathsf{S}_{\mathsf{owf}}(1^\lambda)$ and $x \leftarrow\!\!\$\ \mathsf{A}^{C_{r,0}(\cdot), C_{r,1}(\cdot)}(1^\lambda, 1^{|C_{r,0}|}, \bot)$.[16]

(Part two) F_λ is a OWF. By contradiction, suppose F_λ is not a OWF, i.e., there exists a PPT adversary A such that

$$\mathbb{P}\Big[\mathsf{F}_\lambda(\mathsf{A}(1^\lambda, \mathsf{F}_\lambda(b, r_0, r_1))) = \mathsf{F}_\lambda(b, r_0, r_1) | (b, r_0, r_1) \leftarrow\!\!\$\ \{0,1\} \times \{0,1\}^\lambda \times \{0,1\}^{p(\lambda)}\Big] \geq \epsilon,$$

where ϵ non-negligible. We build an adversary D that breaks the indistinguishability property of Obf (Definition 4.2). D proceeds as follows:

1. Receive an obfuscated circuit $\widetilde{C}$.
2. Execute $\mathsf{A}(1^\lambda, \widetilde{C})$ and receive $(b, r_0, r_1) \in \{0,1\} \times \{0,1\}^\lambda \times \{0,1\}^{p(\lambda)}$.
3. Compute $C' = \mathsf{Obf}(C_{r_0,b}; r_1)$.
4. If $C' = \widetilde{C}$, return b.

Observe that A's view is perfectly simulated. Indeed, $\mathsf{S}_{\mathsf{owf}}$ chooses $r_0 \in \{0,1\}^\lambda$ at random and the obfuscated circuit $\widetilde{C}$ is computed using a fresh randomness $r_1 \leftarrow\!\!\$\ \{0,1\}^{p(\lambda)}$. Hence, the distribution $\widetilde{C}$ is the same of F_λ on random inputs $(b, r_0, r_1) \in \{0,1\} \times \{0,1\}^\lambda \times \{0,1\}^{p(\lambda)}$. This imply that D has the same non-negligible advantage ϵ in distinguishing between the obfuscations C_0 and C_1 output by $\mathsf{S}_{\mathsf{owf}}$. This concludes the proof.

B.10 Proof of Theorem 6.5

If OWFs exist then the following primitives exists:

1. A secure PRF scheme $\overline{\Pi} = (\overline{\mathsf{Gen}}, \bar{\mathsf{F}})$ with key space $\{0,1\}^\lambda$,
2. a SKE $\widehat{\Pi} = (\widehat{\mathsf{KGen}}, \widehat{\mathsf{Enc}}, \widehat{\mathsf{Dec}})$ with key space $\{0,1\}^\lambda$ that is IND-CCA1 and IND-CPA-key secure (Corollary A.20),

[16] Recall that $|C_{r,0}| = |C_{r,1}|$ by definition of sampler (Definition 3.1).

3. an ensemble of circuits $\mathcal{C} = \{C^*_{\mathsf{s},(\mathsf{k},a,b,\mathsf{y},e)}\}_{\mathsf{s},\mathsf{k},a,b,\mathsf{y}\in\{0,1\}^\lambda, e\in\{0,1\}}$ (defined with respect to $\widehat{\Pi}$ and $\overline{\Pi}$) that satisfies Theorem 6.1, and
4. a SKE $\widetilde{\Pi} = (\widetilde{\mathsf{KGen}}, \widetilde{\mathsf{Enc}}, \widetilde{\mathsf{Dec}})$ with key space $\{0,1\}^\lambda$ that is semantically and sel-IND-CPRA-key secure (Corollary A.20).

Consider the following SKE scheme $\Pi^* = (\mathsf{KGen}^*, \mathsf{Enc}^*, \mathsf{Dec}^*)$ with message space $\mathcal{M} = \{(\ell, v)\}_{\ell,v\in\{0,1\}^*}$:

$\mathsf{KGen}^*(1^\lambda)$: On input the security parameter 1^λ, the key generation algorithm computes $\widehat{\mathsf{k}} \leftarrow\!\!\$\ \widehat{\mathsf{KGen}}(1^\lambda)$, $\widetilde{\mathsf{k}} \leftarrow\!\!\$\ \widetilde{\mathsf{KGen}}(1^\lambda)$, $\mathsf{s} \leftarrow\!\!\$\ \overline{\mathsf{Gen}}(1^\lambda)$, $\mathsf{y} \leftarrow\!\!\$\ \overline{\mathsf{Gen}}(1^\lambda)$, $(a, b, e) \leftarrow\!\!\$\ \{0,1\}^{2\lambda+1}$, and returns $\mathsf{k}^* = (\widehat{\mathsf{k}}, \widetilde{\mathsf{k}}, \mathsf{s}, a, b, \mathsf{y}, e)$.

$\mathsf{Enc}^*(\mathsf{k}, m; r)$: On input the key $\mathsf{k}^* = (\widehat{\mathsf{k}}, \widetilde{\mathsf{k}}, \mathsf{s}, a, b, \mathsf{y}, e)$, a message $m = (\ell, v) \in \mathcal{M}$, and randomness $r \in \{0,1\}^*$, the encryption algorithm outputs $c = (c_0, c_1, c_2)$ where $c_0 = C^*_{\mathsf{s},(\widehat{\mathsf{k}},a,b,\mathsf{y},e)}(\ell, v, r)$, $c_1 = \widetilde{\mathsf{Enc}}(\widetilde{\mathsf{k}}, (\ell, v); r)$, and $c_2 = \bar{\mathsf{F}}(\mathsf{y}, (\ell, v, r)) \oplus \widetilde{\mathsf{k}}$.

$\mathsf{Dec}^*(\mathsf{k}, c)$: On input the key $\mathsf{k}^* = (\widehat{\mathsf{k}}, \widetilde{\mathsf{k}}, \mathsf{s}, a, b, \mathsf{y}, e)$ and a ciphertext $c = (c_0, c_1, c_2)$, the deterministic decryption algorithm returns $m = \widetilde{\mathsf{Dec}}(\widetilde{\mathsf{k}}, c_1)$.

First, we prove that Π^* is both semantically and sel-IND-CPRA-key secure. Second, we show that Π is not sel-IND-CPA where Π is the PKE scheme output by the application of Construction 5 to Π^*.

Π^* is semantically secure. Let $m_0^*, m_1^* \in \mathcal{M}$. Consider the following hybrid experiments:

$\mathsf{Hyb}_0^{m_0^*,m_1^*}(\lambda)$ This is the standard experiment of semantic security (Definition A.16) with respect to the messages $m_0^*, m_1^* \in \mathcal{M}$.

$\mathsf{Hyb}_1^{m_0^*,m_1^*}(\lambda)$**:** Same as $\mathsf{Hyb}_0^{m_0^*,m_1^*}(\lambda)$, except that the challenger replaces the execution of $C^*_{\mathsf{s},(\widehat{\mathsf{k}},a,b,\mathsf{y},e)}$ (that is done by the encryption algorithm Enc^*) with the execution of $\widetilde{C}_{\widehat{\mathsf{k}},a,b}$ (depicted in Figure 10).

$\mathsf{Hyb}_2^{m_0^*,m_1^*}(\lambda)$**:** Same as $\mathsf{Hyb}_1^{m_0^*,m_1^*}(\lambda)$, except that the challenger samples the challenge bit $b \leftarrow\!\!\$\ \{0,1\}$ and computes $c_2 = \bar{\mathsf{F}}_{\mathsf{rnd}}(\ell_b^*, v_b^*, r) \oplus \widetilde{\mathsf{k}}$ (instead of $\bar{\mathsf{F}}(\mathsf{y}, (\ell_b^*, v_b^*, r)) \oplus \widetilde{\mathsf{k}}$) where $\bar{\mathsf{F}}_{\mathsf{rnd}}(\cdot,\cdot,\cdot)$ is a truly random function, $m_b^* = (\ell_b^*, v_b^*)$, and $r \leftarrow\!\!\$\ \{0,1\}^*$.

$\mathsf{Hyb}_2^{m_0^*,m_1^*}(\lambda)$**:** Same as $\mathsf{Hyb}_1^{m_0^*,m_1^*}(\lambda)$, except that the challenger computes $c_1 = \widetilde{\mathsf{Enc}}(\widetilde{\mathsf{k}}, (0,0); r)$ (instead of $c_1 = \widetilde{\mathsf{Enc}}(\widetilde{\mathsf{k}}, (\ell_b^*, v_b^*); r)$).

Lemma B.30. *For every $m_0^*, m_1^* \in \mathcal{M}$, $\mathsf{Hyb}_0^{m_0^*,m_1^*}(\lambda) \approx_c \mathsf{Hyb}_1^{m_0^*,m_1^*}(\lambda)$.*

Proof. The lemma follows by leveraging the fact that $\overline{\Pi}$ is a secure PRF scheme and $\widehat{\Pi}$ is IND-CCA1 secure. The proof is similar to that of oracle-differing-input of Theorem 6.1. □

Lemma B.31. *For every $m_0^*, m_1^* \in \mathcal{M}$, $\mathsf{Hyb}_1^{m_0^*,m_1^*}(\lambda) \approx_c \mathsf{Hyb}_2^{m_0^*,m_1^*}(\lambda)$.*

Proof. The lemma follows by leveraging the fact that $\overline{\Pi}$ is a secure PRF scheme. □

Lemma B.32. *For every $m_0^*, m_1^* \in \mathcal{M}$, $\mathsf{Hyb}_1^{m_0^*,m_1^*}(\lambda) \approx_c \mathsf{Hyb}_2^{m_0^*,m_1^*}(\lambda)$.*

Proof. The lemma follows by the semantic security of $\widetilde{\Pi}$. □

By combining Lemmas B.30 to B.32, we obtain that Π^* is semantically secure.

Π^* is sel-IND-CPRA-key secure. Let $m^* \in \mathcal{M}$. Consider the following hybrid experiments:

$\mathsf{Hyb}_0^{d,m^*}(\lambda)$ This is the standard sel-IND-CPRA-key experiment (Definition A.19) with respect to the message $m^* \in \mathcal{M}$ and the challenge bit d.

$\mathsf{Hyb}_1^{d,m^*}(\lambda)$**:** Same as $\mathsf{Hyb}_0^{d,m^*}(\lambda)$, except that the challenger changes how it computes c_0^* of the challenge ciphertext $c^* = (c_0^*, c_1^*, c_2^*)$. Formally, the challenger computes $c_0^* = C^*_{\mathsf{s}_{1-d},(\widehat{\mathsf{k}}_0,a_0,b_0,\mathsf{y}_0,e_0)}(\ell^*, v^*, r)$ (instead of computing $c_0^* = C^*_{\mathsf{s}_d,(\widehat{\mathsf{k}}_d,a_d,b_d,\mathsf{y}_d,e_d)}(\ell^*, v^*, r)$) where $m^* = (\ell^*, v^*)$ and $r \leftarrow\!\!\$\ \{0,1\}^*$.

$\mathsf{Hyb}_2^{d,m^*}(\lambda)$**:** Same as $\mathsf{Hyb}_1^{d,m^*}(\lambda)$, except that the challenger changes how it answers to the oracle queries for $\mathsf{Enc}^*(\mathsf{k}_0^*, \cdot; \cdot)$ and $\mathsf{Enc}^*(\mathsf{k}_1^*, \cdot; \cdot)$. Formally, on input $(m = (\ell, v), r)$ for $\mathsf{Enc}^*(\mathsf{k}_i^*, \cdot; \cdot)$, the challenger computes $c_0 = \widetilde{C}_{\widehat{\mathsf{k}}_i,a_i,b_i}(\ell, v, r)$ (depicted in Figure 10).

$\mathsf{Hyb}_3^{d,m^*}(\lambda)$**:** Same as $\mathsf{Hyb}_2^{d,m^*}(\lambda)$, except that the challenger changes how $\mathsf{Enc}^*(\mathsf{k}_0^*, \cdot; \cdot)$ computes c_2. Formally, on input a message $m = (\ell, v)$ and a randomness r for $\mathsf{Enc}^*(\mathsf{k}_0, \cdot; \cdot)$, the challenger computes $c_2 = \bar{\mathsf{F}}_{\mathsf{rnd0}}(\ell, v, r) \oplus \widetilde{\mathsf{k}}_0$ (instead of $\bar{\mathsf{F}}(\mathsf{y}_0, (\ell, v, r)) \oplus \widetilde{\mathsf{k}}_0$) where $\bar{\mathsf{F}}_{\mathsf{rnd0}}(\cdot, \cdot, \cdot)$ is a truly random function.

$\mathsf{Hyb}_4^{d,m^*}(\lambda)$**:** Same as $\mathsf{Hyb}_3^{d,m^*}(\lambda)$, except that the challenger changes how $\mathsf{Enc}^*(\mathsf{k}_1^*, \cdot; \cdot)$ computes c_2. Formally, on input a message $m = (\ell, v)$ and a randomness r for $\mathsf{Enc}^*(\mathsf{k}_1, \cdot; \cdot)$, the challenger computes $c_2 = \bar{\mathsf{F}}_{\mathsf{rnd1}}(\ell, v, r) \oplus \widetilde{\mathsf{k}}_1$ (instead of $\bar{\mathsf{F}}(\mathsf{y}_1, (\ell, v, r)) \oplus \widetilde{\mathsf{k}}_1$) where $\bar{\mathsf{F}}_{\mathsf{rnd1}}(\cdot, \cdot, \cdot)$ is a truly random function.

Lemma B.33. *For every* $m^* \in \mathcal{M}$, $\mathsf{Hyb}_0^{d,m^*}(\lambda) \approx_c \mathsf{Hyb}_1^{d,m^*}(\lambda)$.

Proof. The lemma follows by leveraging the input-indistinguishability property of $\mathcal{C}$ (Theorem 6.1). □

Lemma B.34. *For every* $m^* \in \mathcal{M}$, $\mathsf{Hyb}_1^{d,m^*}(\lambda) \approx_c \mathsf{Hyb}_2^{d,m^*}(\lambda)$.

Proof. The lemma follows by leveraging the fact that $\overline{\Pi}$ is a secure PRF scheme and $\widehat{\Pi}$ is IND-CCA1 and IND-CPA-key secure. The proof uses a similar argument to that of input-indistinguishability of Theorem 6.1. □

Lemma B.35. *For every* $m^* \in \mathcal{M}$, $\mathsf{Hyb}_2^{d,m^*}(\lambda) \approx_c \mathsf{Hyb}_3^{d,m^*}(\lambda)$.

Proof. The lemma follows by leveraging the fact that $\overline{\Pi}$ is a secure PRF scheme. □

Lemma B.36. *For every* $m^* \in \mathcal{M}$, $\mathsf{Hyb}_3^{d,m^*}(\lambda) \approx_c \mathsf{Hyb}_4^{d,m^*}(\lambda)$.

Proof. The lemma follows by leveraging the fact that $\overline{\Pi}$ is a secure PRF scheme. □

Lemma B.37. *For every* $m^* \in \mathcal{M}$, $\mathsf{Hyb}_4^{d,m^*}(\lambda) \approx_c \mathsf{Hyb}_4^{1-d,m^*}(\lambda)$.

Proof. The lemma follows by leveraging the fact that $\widetilde{\Pi}$ is sel-IND-CPRA-key secure □

By combining Lemmas B.33 to B.37, we obtain that Π is sel-IND-CPRA-key secure.

***Π* is not sel-IND-CPA secure.** Let Π be the PKE scheme output by the application of Theorem 5.7, starting from the SKE scheme Π^*. It is easy to see that Π is not sel-IND-CPA (Definition A.22). This because there always exists an adversary A that, on input $\mathsf{pk} = \widetilde{C}$ (recall that $\widetilde{C}$ is the obfuscation of the circuit $C_{\mathsf{k}}^{\mathsf{Enc}}$ (Figure 6) with respect to the SKE scheme Π^*), is able to correctly decrypt the challenge ciphertext. More formally, let $m_0^*, m_1^* \in \mathcal{M}$ such that $m_0^* \neq m_1^*$ and A be the following adversary (against the sel-IND-CPA security of Π^* with respect to the messages $m_0^*, m_1^* \in \mathcal{M}$):

1. Receive the challenge ciphertext $c^* = (c_0^*, c_1^*, c_2^*)$ and the public key $\mathsf{pk} = \widetilde{C}$.
2. Compute $c' = (c_0', c_1', c_2') = \widetilde{C}(\ell', v', r')$ for some arbitrary $\ell', v', r' \in \{0,1\}^*$.
3. Let C' be the circuit (composed by the gates of $\widetilde{C}$) representing the computation of $\widetilde{C}$ that, on input (ℓ, v, r), output c_0, i.e.,

$$\forall (\ell, v, r) \in \{0,1\}^*, c_0 = c_0' \text{ where } (c_0, c_1, c_2) = \widetilde{C}(\ell, v, r) \text{ and } c_0' = C'(\ell, v, r).$$

4. Compute $(\widehat{\mathsf{k}}, a, b, \mathsf{y}, e) \leftarrow\$ \mathsf{Ext}(1^\lambda, C')$ where Ext is the PPT algorithm satisfying the partial reversibility property of Theorem 6.1.
5. Compute $c_2' \oplus \bar{\mathsf{F}}(\mathsf{y}, (\ell', v', r')) = \widetilde{\mathsf{k}}$.
6. Decrypt c_1^*, i.e., $\widetilde{\mathsf{Dec}}(\widetilde{\mathsf{k}}, c_1^*) = m$.
7. If $m = m_0^*$, return 0. Otherwise, return 1.

By leveraging the partial reversability property of $\mathcal{C}$ (Theorem 6.1), A correctly extracts $\widetilde{\mathsf{k}}$. As a consequence, A breaks the sel-IND-CPA security of Π.