Obfuscation for Cryptographic Purposes[*]

Dennis Hofheinz[†]   John Malone-Lee[‡]   Martijn Stam[§]

May 11, 2009

Abstract

Loosely speaking, an obfuscation O of a function f should satisfy two requirements: firstly, using O, it should be possible to evaluate f; secondly, O should not reveal anything about f that cannot be learnt from oracle access to f alone. Several definitions for obfuscation exist. However, most of them are very hard to satisfy, even when focusing on specific applications such as obfuscating a point function (e.g., for authentication purposes).

In this work, we propose and investigate two new variants of obfuscation definitions. Our definitions are simulation-based (i.e., require the existence of a simulator that can efficiently generate fake obfuscations) and demand only security on average (over the choice of the obfuscated function). We stress that our notions are not free from generic impossibilities: there exist natural classes of function families that cannot be securely obfuscated. Hence we cannot hope for a general-purpose obfuscator with respect to our definition. However, we prove that there also exist several natural classes of functions for which our definitions yield interesting results.

Specifically, we show that our definitions have the following properties:

Usefulness: Securely obfuscating (the encryption function of) a secure private-key encryption scheme yields a secure public-key encryption scheme.

Achievability: There exist obfuscatable private-key encryption schemes. Also, a point function chosen uniformly at random can easily be obfuscated with respect to the weaker one (but not the stronger one) of our definitions. (Previous work focused on obfuscating point functions from arbitrary distributions.)

Generic impossibilities: There exist unobfuscatable private-key encryption schemes. Furthermore, pseudorandom functions cannot be obfuscated with respect to our definitions.

Our results show that, while it is hard to avoid generic impossibilities, useful and reasonable obfuscation definitions are possible when considering specific tasks (i.e., function families).

Keywords: obfuscation, point functions.

1 Introduction

Suppose a software vendor wants to sell its products without giving away internal know-how used in the code. In other words, the software should provide the intended functionality, yet hide internal implementation details completely, even from a curious party that can see and analyze the (compiled) program code. Although there are hardware-based alternatives, the obvious way to achieve this is

[*] This is the full version of [21], which appeared at the Theory of Cryptography Conference (TCC) 2007. Work by the second and third author was partially conducted while employed at the University of Bristol. This work was partially funded by the European Commission through the ICT programme under Contract ICT-2007-216646 ECRYPT II.

[†] CWI, Amsterdam, The Netherlands, Dennis.Hofheinz@cwi.nl

[‡] EMB Consultancy LLP, Epsom, UK, malone@cs.bris.ac.uk

[§] EPFL/LACAL, Lausanne, Switzerland, martijn.stam@epfl.ch

to *obfuscate* the code, i.e., to make it incomprehensible. Intuitively, the obfuscation of a piece of code (or of a function) should provide nothing more than the possibility of evaluating that function. A little more technically, from an obfuscation of a function one should not be able to learn more than one can learn from oracle access to that function. Here we restrict ourselves to learning in a computationally restricted sense.

As another use case of obfuscation techniques, consider a password query during a user login onto a computer terminal. Verification that a user correctly entered his or her password can be done of course by storing all user passwords in the clear on that machine. This works, but an adversary who breaks into that machine can learn all user passwords with just read-only access. We can do much better by storing only hashes (or, images of the password under a one-way function) $H(p)$ for every user password p. Verification of a user-entered password p' with p is then done by comparing $H(p)$ and $H(p')$. One-wayness of H guarantees that p (or any other "good" password p' with $H(p) = H(p')$) is not found even when $H(p)$ becomes known.

Abstracting here, the functionality of this password authentication is that of evaluating a *point function*. (A point function is a function P_x with $P_x(x') = 1$ if $x = x'$, and $P_x(x') = 0$ else.) Informally, hence, the verification procedure that uses $H(p)$ can be considered a useful obfuscation of a point function. Namely, it provides the desired functionality,[1] but in a way such that releasing the implementation, here $H(p)$, does not enable an adversary to learn p.

Focus. The focus of this work is the technical definition of a secure obfuscation suitable for cryptographic purposes. Before we detail our own contribution and our results, we give a survey of previous work.

History and survey of related work. Practical yet informal approaches to code obfuscation were considered by Jaeschke [23] and Linn and Debray [25]. Goldreich and Ostrovsky [13] show how to use a low-memory secure hardware component to obfuscate general programs. A crucial ingredient in their construction is *oblivious* memory access (i.e., they consider machines whose memory access behavior does not depend on the input). Another early theoretical contribution to obfuscate functions was made by Hada [19]. He gave a simulation-based security definition for obfuscation and related it to zero-knowledge proof systems.

In their seminal paper, Barak et al. [3] define a hierarchy of obfuscation definitions, the weakest of which is predicate-based, and the strongest of which is simulation-based. They show that there are function families that cannot be obfuscated, even under the weakest definition that they proposed. Specifically, they show that there are (contrived) sets of functions, such that no single obfuscation algorithm can work for all of them (and output secure obfuscations of the given function). Hence, Barak et al. rule out the possibility of *generic* obfuscation. (And, jumping ahead, we stress that the proof argument they give also applies to our notion.) Yet, Barak et al. leave room for the possibility of obfuscators for *specific* families of functions.

Goldwasser and Kalai [16] present obfuscation definitions that model several types of auxiliary information available to an adversary. Their definitions are predicate-based. One of them, in contrast to the main definitions of Barak et al., models a *random* choice of the function to be obfuscated. They show general impossibility results for these definitions using *filtered functions* (functions whose output is forced to $\perp$ if the input is not "certified" by a witness). In particular, with respect to obfuscation with dependent input, they show the following. *Either* common cryptographic primitives (such as encryption, signing and pseudorandom functions) cannot be obfuscated, *or* a

[1] Technically, in fact, *perfect* functionality is only provided if H is a one-way *permutation*. Otherwise, there might be $x' \neq x$ with $H(x') = H(x)$, so that x' passes the verification although it should not.

large class of filtered functions based on $\mathcal{NP}$-complete problems cannot be obfuscated (or both). (The latter would imply that no $\mathcal{NP}$-complete language has a hard-core predicate.) They also show that with respect to their definitions, an obfuscation of a point function is secure against adversaries without auxiliary information if and only if it is secure against adversaries with (point-)independent auxiliary information.

Even before a precise definition of obfuscation was formulated, positive obfuscation results were given implicitly and in a different context for a special class of functions. Namely, Canetti [8] and Canetti et al. [10] essentially obfuscate point functions. The construction from Canetti [8] works for (almost) arbitrary function distributions and hence requires a very strong computational assumption. On the other hand, one construction from Canetti et al. [10] requires only a standard computational assumption, but is also proven only for a uniform point function distribution. Another construction of [10] works for arbitrary distributions, but assumes the existence of a specific type of hash function.

Positive results for the predicate-based definition of Barak et al. [3] were demonstrated by Lynn et al. [26]. They show how to employ a random oracle to obfuscate access control functions efficiently. This includes point functions. A generalization of point functions can be found in the work of Dodis and Smith [12] who show how to obfuscate a proximity function.

Subsequently Wee [34] showed how to obfuscate point functions in the standard model (still predicate-based). Yet he only does this under very strong computational assumptions and for a very weak definition of obfuscation. Wee also shows that, at least under one of the original obfuscation definitions of Barak et al., strong computational assumptions are necessary for obfuscating point functions.

Recently (and concurrently to the conference version [21] of this paper), relaxed definitions of obfuscation have been considered by Goldwasser and Rothblum [18], and Hohenberger et al. [22]. Goldwasser and Rothblum allow an obfuscation to leak as much information as any implementation of the function of a specific size would. Potentially this reveals more information than can be obtained in a black-box way from the function. They show that this leads to a strictly weaker but still meaningful definition. On the other hand, [22] demands only *average-case* security for probabilistic functions (very similar to our definition). They also show how to obfuscate the task of re-encrypting ciphertexts.

Also related is the recent work on public key obfuscation by Ostrovsky and Skeith III [31] and later by Adida and Wikström [1]. In this setting, obfuscating a function means that one obfuscates the composition of that function followed by encryption. (Thus, querying the obfuscated function results in encrypted function values.) Correctness (or functionality) is defined relative to the decryption. Security is based on the notion of indistinguishability under chosen plaintext attacks. Here the adversary gets to pick two functions, and he gets a randomly chosen public key obfuscation of one of them. The adversary has to guess of which one it is. Public key obfuscation does not seem to fit cleanly within any of the other definitional models.

Finally, Narayanan and Shmatikov [30] investigated to what extent point function obfuscations can be used to bootstrap other obfuscations. They did this under a definition of obfuscation in which adversaries are bounded *only* in their number of oracle queries, but not in the number of their computation steps. With respect to this definition, Narayanan and Shmatikov show that there are circuits which can be obfuscated with a random oracle, but not with just an oracle to a point function. Narayanan and Shmatikov [30] also improve an upper bound on the concurrent self-composability (i.e., security preservation if several obfuscated instances of the same function are available) of Wee's construction for point function obfuscation.

Our results. Our own contribution is two-fold:

Definitional contribution. We consider two specific variants of simulation-based obfuscation definitions. Our weaker definition could be called our "main" definition, because most of our results are formulated with respect to it. However, we also present a stronger definition. Our stronger definition is harder to achieve, and indeed our examples on how to achieve it are less interesting. However, the stronger definition behaves more nicely in larger contexts. Specifically, an obfuscation that satisfies the stronger definition can be simulated even in contexts in which (partial or full) information about the obfuscated input is used.

Note that we call these variants "our definitions" since they have not been considered before in the literature. However, technically they merely combine a number of known definitional ingredients in a new way. We compare and relate our definitions to a number of known definitions, and investigate how known (impossibility) results carry over to our definitions.

Results for our new definitions. We show that our definitions are useful cryptographic definitions in the following sense:

Useful building block. Our definitions serve as a useful building block in a larger cryptographic system. Specifically, secure obfuscators in our sense can be used to turn private-key cryptography into public-key cryptography. For instance, a private-key encryption scheme can be transformed into a public-key encryption scheme by obfuscating the encryption algorithm. If the obfuscation satisfies our weaker obfuscation definition, then this transformation preserves the passive (IND-CPA) security of the encryption scheme. Our stronger notion even preserves active (IND-CCA) security. In that sense, our definitions are "not too weak."

Achievable. Our definitions can be achieved for a number of interesting function classes under standard computational assumptions. In particular, we show that our weaker obfuscation definition can be achieved for point functions (with respect to a uniform distribution on the point function). Furthermore, we exemplify that the encryption algorithm of certain private-key encryption schemes is obfuscatable with respect to both our weaker and our stronger definition. (Although arguably, this merely constitutes a proof of concept.) In that sense, our definitions are "not too strong." However, we stress that there are also natural classes of functions which *cannot* be obfuscated according to our definitions. Examples of such function classes are, e.g., pseudorandom functions.

We give a more detailed explanation of our results below.

Our new definitions. More concretely, we introduce variants of the simulation-based definition of Barak et al. [3]. Roughly, a simulation-based obfuscation definition requires that there is a simulator that, using oracle access to the obfuscated function only, efficiently produces fake obfuscations which are indistinguishable from real obfuscations. We deviate from [3] in the following respects:

Security on average. We randomly choose the function to be obfuscated according to a distribution and demand only "good obfuscations on average." Here "good" refers to simulatability. This is unlike the main definitions of [3], which demand good obfuscations for every function in a given set. Our definitional choice follows from the intuition that in many applications, the function in question is initially sampled from its function class in an honest way. For instance, in a private-key encryption scheme, we can assume that the key itself (which determines the corresponding encryption function) is honestly and uniformly chosen. (If the key is sampled by an adversarial entity, then we cannot guarantee any reasonable form of security anyway.) Technically, our definition follows Canetti's "oracle hashing" definition [8], Hada's security definition [19], and Goldwasser and Kalai [16]. These notions also demand security on average in the same way as we do.[2]

Access to the obfuscated function. When it comes to distinguishing real from fake obfuscations, the distinguisher should have some information about *what* function has been obfuscated (in particular given that the function is sampled from a distribution, as described above). Barak et al. [3] demand security for *every* function in a given set, so that, implicitly, the distinguisher gets the function description itself as input. Other definitions provide the distinguisher with an independently generated obfuscation of the same function [19], with auxiliary information about the function [16],[3] or with oracle access to the function [22]. Our two new definitions differ in the information the distinguisher receives about the obfuscated function. Concretely, our weaker definition grants the distinguisher *oracle access only* to the obfuscated function. The stronger definition gives the function description itself to the distinguisher.

Obfuscation of probabilistic functions. We consider probabilistic functions. That is, we consider functions whose output on a given input is a distribution of output values as opposed to a single output value. This is a minor technical change that can easily be applied to previous definitions. (And to some extent this is already considered in the "sampling algorithms" section of [3].) However, this change is essential, as it enables the obfuscation of for instance encryption algorithms that necessarily behave probabilistically. Furthermore, allowing probabilistic functions for obfuscation opens another door, as follows. A simulation-based obfuscation definition (such as ours or that of Hada [19]) is very restrictive. Namely, following Hada [19] and Wee [34], we remark that any family of deterministic functions must be approximately learnable to be obfuscatable. For probabilistic functions, this level of learnability is not required and one can hope to obfuscate more interesting classes of functions. Indeed, this hope is shown justified in the concurrent work of Hohenberger et al. [22].

For a detailed comparison of existing definitions to ours, see Section 4.3.

Negative results for our definitions. As a first observation, we stress that the generic impossibility results from [3] also carry over to our notions in a meaningful way (see also [3, Discussion after Theorem 4.5]). That means that also for (both of) our notions, there can be no general-purpose obfuscators. Indeed, recall that deterministic functions must be approximately learnable in order to be obfuscatable (in a simulation-based way). As a consequence, we can prove that in particular it is not possible to obfuscate pseudorandom functions (PRFs) under our definition. Barak et al. already consider the obfuscation of PRFs ([3, Theorem 4.12 of full version]). They show that *specific* (contrived) PRFs exist such that any obfuscation would leak the PRF's key. In contrast to that, we show that *no* PRF can be obfuscated. This impossibility however applies only to simulation-based obfuscation definitions such as ours.

Positive results for our definitions. We show how our definitions can be a useful cryptographic building block. Namely, we show that obfuscations secure according to our definitions can be used to turn private-key cryptography into public-key cryptography. For instance, obfuscating the encryption function of a secure private-key encryption scheme *should* (intuitively) yield a secure public-key encryption scheme. We show that this intuition holds true when using our obfuscation definitions. Interestingly, the degree of preserved security depends on which of our obfuscation

[2] When demanding security on average, it is reasonable to ask whether the obfuscator itself gains from being probabilistic. For example, one might hope to extract sufficiently good random coins for the obfuscation algorithm itself from its input; after all, the input itself is chosen randomly and contains a certain min-entropy. We will comment below, after the actual security definition, further on this.

[3] Since the definitions of [16] are predicate-based, there is actually no distinguisher in their setting; there, adversary (and simulator) get auxiliary information about the function.

definitions is used. Namely, the stronger definition preserves active (IND-CCA) security of the encryption scheme, while the weaker definition only preserves passive (IND-CPA) security (and, in general, not IND-CCA security). Hence, our definitions are *useful* in the sense that secure obfuscations can be used in a meaningful way in larger constructions.

On the other hand, our definitions are *achievable.* For instance, there exist (contrived) private-key encryption schemes whose encryption function is obfuscatable according to our definitions. Concretely, any public-key encryption scheme can be modified into an obfuscatable private-key encryption scheme: the obfuscation is the public key. Of course, this construction is not very interesting, but it does show that our notion is in principle achievable, even for the non-trivial class of encryption functions. In particular, it rules out general impossibility results in this direction and leaves hope for more meaningful obfuscations.

We show similar statements for authentication schemes, although in that case there are some serious caveats, as follows. Namely, the analogue of passive security for signature schemes is a trivial security notion. Hence, our weaker security notion is not very useful in that context. Our stronger security notion preserves security against active attacks. However, in case of signature schemes, our stronger notion is not achievable in the standard model.

Also, we prove that point functions can be obfuscated very easily with respect to the weaker (but not the stronger) of our definitions. To obfuscate a point function P_x, publish $\Pi(x)$ for a one-way permutation Π. This closely mimics the way password checks are implemented in practice (as sketched above). However, note that we can prove security only under the assumption that the password is chosen from the uniform distribution. In practice, it may seem more reasonable to model passwords as coming from a low-entropy distribution.

General composability of obfuscations. However, note that the simple point function obfuscation just described is considerably weaker than previous point function obfuscations: security is lost when the point function is not chosen randomly, or when auxiliary information about the point x is published. In particular, security may be lost when the obfuscation is used in larger settings in which x is used in several places. We use this observation as a motivation to investigate when the security of obfuscation is preserved in larger contexts. Concretely, we show that our stronger obfuscation definition *does* remain secure in larger contexts. Technically, we give a formulation of a secure obfuscation in the indifferentiability framework [27] which provides general secure composition. (That is, any security statement automatically also holds under composition in the indifferentiability framework, similarly to the universal composability [9] and reactive simulatability [2] models.) We prove that our stronger definition, along with a suitable correctness requirement, is equivalent to indifferentiable obfuscation, and hence behaves well in larger contexts.

Additionally, we prove that our stronger obfuscation notion behaves well under self-composition. That is, we show that several (different) obfuscations of the same function can be used concurrently without sacrificing security.

2 Notation

Throughout the paper, $k \in \mathbb{N}$ denotes a security parameter. With growing k, attacks should become harder, but we also allow schemes to be of complexity which is polynomial in k. A PPT algorithm/Turing machine is a probabilistic algorithm/Turing machine which runs in time polynomial in k. All algorithms implicitly receive the security parameter as input, so we write $A(\varepsilon)$ for algorithms without further input (where ε denotes the empty bitstring). If A is a PPT algorithm, then $Y \leftarrow A(X)$ denotes that Y is the random variable obtained by running A with uniformly distributed random coins and on input X. If $\mathcal{X}$ is a distribution, then $X \leftarrow \mathcal{X}$ denotes that X is sampled according to $\mathcal{X}$. If $\mathcal{X}$ is a set, then $X \overset{\$}{\leftarrow} \mathcal{X}$ means that X is sampled according to the uniform distribution over $\mathcal{X}$. For a random variable X the expected value of X is denoted by $\mathsf{EV}[X]$. For random variables X_1, X_2 their statistical distance $\frac{1}{2}\sum_x |\mathsf{Pr}[X_1 = x] - \mathsf{Pr}[X_2 = x]|$ is denoted by $\Delta(X_1\,;\,X_2)$. A function $\nu : \mathbb{N} \to \mathbb{R}$ is negligible iff for all $c > 0$, we have that $|\nu(k)| < k^{-c}$ for sufficiently large k. For two functions $f, g : \mathbb{N} \to \mathbb{R}$ we write $f \overset{\text{c}}{\approx} g$ iff their difference $f - g$ is negligible.

3 Previous Obfuscation Definitions

Intuition and worst-case obfuscation. We recall the obfuscation definitions from Barak et al. [3]. As mentioned in the introduction, intuitively an obfuscation O of a function f should provide nothing more than the possibility to evaluate that function. Now it does not make sense to speak about a single, isolated function f here. If f is agreed upon, there is no need for an obfuscation in the first place. Hence we consider a family $\mathcal{F} = (\mathcal{F}_k)_{k \in \mathbb{N}}$ with $\mathcal{F}_k = (f)$ of functions. Henceforth it is understood that all functions $f \in \mathcal{F}_k$ can be represented in polynomial space allowing evaluation in polynomial time (in k). Whenever we input f to an algorithm, we implicitly refer to its description. (And consequently, the particular representation of the function could make a difference.)

Syntactically, an obfuscator $\mathcal{O}$ for $\mathcal{F}$ takes as input the description of a function $f \in \mathcal{F}$ and outputs an obfuscation $O = \mathcal{O}(f)$. Formally, both the function and the obfuscation are represented as a circuit. The definitions of Barak et al. require that for *every* $f \in \mathcal{F}$, the obfuscation $\mathcal{O}(f)$ should be secure, in a sense to be defined. Because security is required for *all* f, we will call Barak et al.'s definitions *worst-case.*

Definition 3.1 (Worst-case obfuscation [3] (generic)). *Let $\mathcal{F} = (\mathcal{F}_k)$, where each $\mathcal{F}_k$ is a family of functions associated with security parameter k. Let $\mathcal{O}$ be a PPT algorithm which maps (descriptions of) functions f to circuits $\mathcal{O}(f)$. We say that $\mathcal{O}$ is a* worst-case obfuscator for $\mathcal{F}$ *iff the following holds.*

Functionality: *For all $k \in \mathbb{N}$, all $f \in \mathcal{F}_k$, and all possible $O = \mathcal{O}(f)$, we have that O computes the same function as f.*

Virtual black-box (informal): *For all $f \in \mathcal{F}_k \subseteq \mathcal{F}$, given access to the obfuscation $O = \mathcal{O}(f)$, an adversary cannot learn anything about the original function f that it could not have learnt from oracle access to f.*

Barak et al. also mention a third requirement, namely polynomial slowdown. This means that the time it takes to evaluate the obfuscated function is polynomially bounded by the runtime of the original. Since we require $\mathcal{O}$ to be PPT and to output a circuit, polynomial slowdown is fulfilled automatically.

We stress that the functionality requirement can be relaxed in meaningful ways. For instance, one might require "approximate functionality" in the sense that O evaluates f, except with negligible probability over the random coins of $\mathcal{O}$. Our results are not affected by such a relaxation. For probabilistic functions f, the functionality requirement is to be understood such that O gives rise to the same output distribution as f. (Appendix A is dedicated to a more detailed discussion of the functionality requirement for probabilistic functions.)

The virtual black-box requirement can be formalized in different ways. For instance, one could require that no adversary can approximate a non-trivial predicate on (the description of) f. Alternatively, one could demand that there exists a simulator that produces fake obfuscations, using oracle access to f only.

We now elaborate on these variants of the virtual black-box requirement.

Predicate-based worst-case obfuscation. This variant of the virtual black-box requirement is based on *computing a predicate on the description of f.* Concretely, the task of an adversary A given the obfuscation $\mathcal{O}(f)$ is to compute a boolean predicate π on the description of f. Of course, there are always predicates, such as constant predicates, which can be easily computed. Hence, we want that $\mathcal{O}(f)$ does not *help* in approximating $\pi(f)$. That is, for any adversary A and any boolean predicate π, the probability that an adversary computes $\pi(f)$ given $\mathcal{O}(f)$ is not significantly greater than the probability that a suitable simulator S, given only oracle access to f, computes $\pi(f)$. This notion is formally defined by a slightly simpler, but equivalent, notion:

Definition 3.2 (Predicate-based worst-case virtual black-box [3])**.** *Let $\mathcal{F}, \mathcal{O}$ be as in Definition 3.1. Then $\mathcal{O}$* satisfies the predicate-based worst-case black-box property *iff for all PPT algorithms A, there exist a PPT algorithm S and a negligible function ν such that for all $k \in \mathbb{N}$ and $f \in \mathcal{F}_k$, we have*

$$\mathsf{Pr}\left[A(\mathcal{O}(f)) = 1\right] - \mathsf{Pr}\left[S^{f(\cdot)}(\varepsilon) = 1\right] \leq \nu(k).$$

Definition 3.2 is the main notion of security used in the impossibility result of Barak et al. [3]. (However, their result holds for a number of other definitional variants, including all the definitions discussed in this work.)

Simulation-based obfuscation This secrecy requirement is based on *computational indistinguishability.* Under this formulation one does not restrict the nature of what an adversary must compute. Concretely, we require that it is possible to produce, from oracle access to f only, "fake obfuscations." These fake obfuscations should be computationally indistinguishable from the real obfuscations $\mathcal{O}(f)$. Hence also every (efficient) function computed on $\mathcal{O}(f)$ can be approximated using oracle access to f. Obfuscators satisfying Definition 3.3 can easily be seen to satisfy Definition 3.2 as well.

Definition 3.3 (Simulation-based worst-case virtual black-box [3], formulation due to [34])**.** *Let $\mathcal{F}$ and $\mathcal{O}$ be as in Definition 3.1. Then $\mathcal{O}$* satisfies the simulation-based worst-case black-box property *iff there exists a PPT simulator S such that for every PPT distinguisher D, some negligible function ν, all $k \in \mathbb{N}$ and all $f \in \mathcal{F}_k$, we have*

$$\mathsf{Pr}\left[D(\mathcal{O}(f)) = 1\right] - \mathsf{Pr}\left[D(S^{f(\cdot)}(\varepsilon)) = 1\right] \leq \nu(k).$$

Connection to learnability. The following has been noticed already by Hada [19] and formally shown by Wee [34], Proposition 5.2. Namely, simulation-based worst-case obfuscation can be achieved for (deterministic) functions f precisely when f is efficiently and exactly learnable through oracle access. For instance, in case the function is learnable, the obfuscation can return the learned function. However, Wee's result uses that the function in question is deterministic, so we can hope to avoid the strict requirement of learnability when considering probabilistic functions.

4 Our Definition

We first present our own definition (which is a variant of simulation-based obfuscation), and then justify the main design choices we made. In fact, we will present *two* definitions, which are syntactically very similar, but differ substantially in their properties (we will explore the properties of our definitions in Section 5 and Section 6). The first, weaker definition can be seen as our "main" definition, since most of our results hold with respect to this definition. However, the second, stronger definition will turn out to have a number of nice properties that make up for some of the weaker definition's drawbacks. After our definitions have been presented and discussed, we will compare them to a number of existing definitions.

4.1 The Definitions

As noted, our definitions are simulation-based, but we do not call them "worst-case," since we only require security on average (over f):

Definition 4.1 (Simulation-based average-case virtual black-box property)**.** *Let $\mathcal{F} = (\mathcal{F}_k)$, where each $\mathcal{F}_k$ is a family of probabilistic functions associated with security parameter k. Let $\mathcal{O}$ be a PPT algorithm which maps (descriptions of) functions f to circuits $\mathcal{O}(f)$. We say that $\mathcal{O}$ has the* simulation-based average-case virtual black-box property *iff for every PPT distinguisher D, there is a PPT simulator S, such that*

$$\mathsf{Adv}^{\mathsf{sbvbb}}_{\mathcal{F},\mathcal{O},D,S}(k) := \mathsf{Pr}\left[f \xleftarrow{\$} \mathcal{F}_k \,:\, D^{f(\cdot)}(\mathcal{O}(f)) = 1\right] - \mathsf{Pr}\left[f \xleftarrow{\$} \mathcal{F}_k \,:\, D^{f(\cdot)}(S^{f(\cdot)}(\varepsilon)) = 1\right] \qquad (1)$$

is negligible in k.

We stress that oracles in this definition (in particular, D's $f(\cdot)$ oracle) may use randomness that is independent and hidden from D. In particular, D does not get to choose or even see the random coins used to evaluate f. In case of probabilistic functions f, this opens a door to sidestep impossibility results (see, e.g., Hohenberger et al. [22], Section 2 for an example and further discussion).

We also introduce a stronger variant, which features a different order of quantifiers and gives the distinguisher D access to the full description of the function f:

Definition 4.2 (Strong simulation-based average-case virtual black-box property)**.** *Let $\mathcal{F} = (\mathcal{F}_k)$, where each $\mathcal{F}_k$ is a family of probabilistic functions associated with security parameter k. Let $\mathcal{O}$ be a PPT algorithm which maps (descriptions of) functions f to circuits $\mathcal{O}(f)$. We say that $\mathcal{O}$ has the* strong simulation-based average-case virtual black-box property *iff there is a PPT simulator S, such that for every PPT distinguisher D, we have that*

$$\mathsf{Adv}^{\mathsf{ssbvbb}}_{\mathcal{F},\mathcal{O},D,S}(k) := \mathsf{Pr}\left[f \xleftarrow{\$} \mathcal{F}_k \,:\, D(f, \mathcal{O}(f)) = 1\right] - \mathsf{Pr}\left[f \xleftarrow{\$} \mathcal{F}_k \,:\, D(f, S^{f(\cdot)}(\varepsilon)) = 1\right] \qquad (2)$$

is negligible in k.

The following lemma is trivial from the definitions:

Lemma 4.3 (Strong average-case implies average-case)**.** *Whenever an obfuscator satisfies the strong simulation-based average-case virtual black-box property (Definition 4.2), it satisfies the simulation-based average-case virtual black-box property (Definition 4.1).*

4.2 Motivation of Design Choices

Based on simulation. Our definitions require the existence of a simulator that is able to generate "fake obfuscations" that are indistinguishable from real obfuscations. This design choice makes our definition easy to use in larger constructions. For instance, in a game-based security proof of a cryptographic system, one substitutes components of the real system one by one with different, but indistinguishable components, until one ends up with an ideal system which is trivially secure. The obfuscation $\mathcal{O}(f)$ and the simulated obfuscation $S^f(\varepsilon)$ can be seen as such indistinguishable and hence interchangeable components. We demonstrate this concept in a number of proofs in Section 5, where we show how to *use* our definitions. We stress that predicate-based obfuscation definitions do *not* support game-based proofs in a similar way. Indeed, while a predicate-based definition states that an obfuscation leaks no given (non-trivial) predicate bit about f, it is unclear how this could be used as an intermediate step in a game-based proof. (Usually it is not a priori clear how the obfuscation is used by an adversary to achieve his overall attack goal. Hence it is not clear what predicate of f the adversary crucially uses.)

We summarize that using a simulation-based definition is the key to the *usability* of our definition.

Probabilistic functions. In contrast to earlier works, we consider *probabilistic functions*. First, this obviously makes it easier to express, say, the (necessarily probabilistic) encryption algorithm of a secure encryption scheme as a suitable function family $\mathcal{F}$. But we gain much more than just syntactic compatibility. More importantly, probabilistic functions avoid the necessity for f to be learnable in order to be obfuscatable. (Recall that for deterministic functions f, learnability and simulation-based obfuscatability are equivalent.)

Concurrently to our work, also Hohenberger et al. [22] use probabilistic functions instead of deterministic ones in order to prove their positive result. Furthermore, probabilistic functions have been implicitly used in the context of obfuscation by Adida and Wikström [1], Ostrovsky and Skeith III [31] in the setting of encryption.

Hence, we gain *expressiveness* and *achievability* by using probabilistic functions.

Average-case security. We do not require security for *all* functions f, but instead we require security *on average*, over a uniform choice of $f \in \mathcal{F}_k$. This relaxation of Barak et al.'s definition will be the key to the *achievability* of our definition. Namely, a random choice of f enables reducing the security of an obfuscator to standard cryptographic assumptions. In such standard assumptions, usually also a random choice of an underlying secret is crucial. For instance, Theorem 5.4 reduces the security of an obfuscator for point functions to the security of a one-way permutation. Now in the proof, the point function P_x, or rather the secret point x, can be directly mapped to the preimage of the one-way permutation, since both are distributed uniformly. This enables a very simple security proof. The two security experiments (obfuscation and underlying primitive) are "compatible."

We stress that, with similar arguments, the idea of average-case obfuscation has already been considered in a number of works, e.g., by Hada [19], Goldwasser and Kalai [16]. Also, concurrently to our work, Hohenberger et al. [22] use and *achieve* (for a non-trivial and natural class of functions) an average-case definition quite similar to ours.

We summarize that average-case security is the key to the *achievability* of our definition.

On the necessity of a probabilistic obfuscator. We consider a distribution on the obfuscated functions f, and in particular f is chosen *after* the distinguisher D. Hence, the obfuscator $\mathcal{O}$ essentially gets two types of random inputs, both independent of D: a randomly selected $f \in \mathcal{F}$, and $\mathcal{O}$'s own random coins. The question arises whether $\mathcal{O}$ can do without its random coins, and derive all needed randomness from its input f.[4] In other words: is it feasible without loss of generality to assume that $\mathcal{O}$ is deterministic?

[4] Such techniques have been successfully applied in the encryption setting, see Bellare et al. [7].

We contend that the answer is no. To explain, consider the stronger Definition 4.2 in case of a deterministic obfuscator $\mathcal{O}$. Hence, the obfuscation $\mathcal{O}(f)$ follows deterministically from the input function f. Specifically, since the distinguisher D in Definition 4.2 gets f as input, D can always compute $\mathcal{O}(f)$ on its own and compare the result to its second input O. When D's second input really is an obfuscation, then $O = \mathcal{O}(f)$. A successful (in the sense of Definition 4.2) simulator must hence achieve that its output matches $\mathcal{O}(f)$ with overwhelming probability.

Thus, to achieve Definition 4.2 in case of deterministic obfuscators, there must exist a simulator S that finds the (unique) real obfuscation $\mathcal{O}(f)$ except with negligible probability. Because we require perfect functionality, $\mathcal{O}(f)$ evaluates f everywhere. This means that a successful S has to learn f efficiently and exactly from oracle access only. (Note that this holds even if f is a probabilistic function, since S has to outputs *the same function* $\mathcal{O}(f)$ with overwhelming probability.) Consequently, while we do not have a natural example of a function family that can *only* be obfuscated with a probabilistic obfuscator, it seems that we gain generality by the obfuscator to be probabilistic.

The difference between our two definitions. Our Definitions 4.1 and 4.2 differ in two aspects. Namely, first, the order of quantifiers ($\forall D \exists S$ vs. $\exists S \forall D$), and second, how D may access f (oracle access to f vs. getting f as input). Definition 4.2 is the stricter definition in both respects. This leads to a more versatile but also harder to achieve definition. Most of our results hold with respect to the weaker Definition 4.1, hence Definition 4.1 can be called our main (or standard) definition. However, we include Definition 4.2 because it shows how to circumvent some of the shortcomings of Definition 4.1.

In particular, Definition 4.1 cannot generally be used in larger contexts in which the function f is itself is used. As an example, consider the obfuscation of the encryption algorithm of a private-key encryption scheme. It is reasonable to assume that in a larger context, the decryption algorithm is used (with respect to the same private key). In such a context, obfuscations that satisfy Definition 4.1 may not remain secure. (In a nutshell, this holds because Definition 4.1 considers an isolated obfuscation of f, while, say, a decryption oracle uses information related to f. For details, Section 5.3.)

Definition 4.2 provides a more useful definition in such settings. Particularly, Definition 4.2 guarantees that the obfuscation of (the encryption algorithm of) a private-key encryption scheme remains secure in the presence of a decryption oracle. More generally, we will show that Definition 4.2 can be used in a large class of contexts (see Section 6). However, at the same time, Definition 4.2 appears to be extremely strict. In fact, we can only show that Definition 4.2 is achievable for toy examples (such as in Theorem 5.15).

Auxiliary input and composability. Goldwasser and Kalai [16] distinguish in their obfuscation definitions two types of auxiliary information that an adversary might possess: *(f-)dependent auxiliary information* and *(f-)independent auxiliary information.* (Their definitions are reproduced below in Definition 4.4 and Definition 4.5.) At first glance, our definitions do not feature any kind of auxiliary information, and it might seem surprising that we can derive any compositional guarantees without auxiliary information. (Usually, auxiliary information is a technical tool to derive security guarantees in larger contexts: the auxiliary information given to the adversary incorporates all information that an adversary could obtain from the context "surrounding" the obfuscation.)

However, at a closer inspection, in particular Definition 4.2 grants the adversary a very specific (f-dependent) auxiliary information about f: namely, f itself. (And consequently, the adversary can derive any information that can be efficiently computed from f.) This is the key to our compositional guarantees we can provide for Definition 4.2 (cf. Section 6). The auxiliary information provided to the adversary in Definition 4.1 is much weaker: here, it consists of oracle access to f. (And consequently, Definition 4.1 fails to provide strong compositional guarantees, cf. the discussion in Section 5.1.)

We stress that the technical way that auxiliary information is incorporated into the definitions by Goldwasser and Kalai (see also Definition 4.4) differs from ours. Namely, Goldwasser and Kalai hand the auxiliary information to both adversary and simulator. (In their case, there is no distinguisher.) In our case, the simulator does not get any form of auxiliary information, not even in our stronger Definition 4.2. This technical difference makes relating their definitions to ours even harder.

On the other hand, adding *independent* auxiliary input to our definitions (similarly to Hohenberger et al. [22]) does not alter our results. All reductions and proofs derived still hold, only that the respective computational assumptions must hold against non-uniform adversaries.

4.3 Comparison with Other Definitions

The relationship with predicate-based definitions. The main definition of Barak et al. is a predicate based definition (see also Definition 3.2). It was later modified by Goldwasser and Kalai [16], who demand security in the presence of auxiliary information on the key. More specifically, one of the definitions by Goldwasser and Kalai [16] models security in presence of auxiliary information that *depends* on the particular function to be obfuscated. The other definition from Goldwasser and Kalai [16] models security in the presence of auxiliary information that is *independent* of the obfuscated function.

To ease a comparison, we recast their definitions in our notation. This results in a minor change in the definition due to their emphasis on circuits, whereas we consider more general function families.

Definition 4.4 (Goldwasser and Kalai's obfuscation w.r.t. dependent auxiliary input [16, Definition 3])**.** *An obfuscator $\mathcal{O}$ for a function family $\mathcal{F} = (\mathcal{F}_k)$ is* secure with respect to dependent auxiliary input *iff for every PPT adversary A, there exist a PPT simulator S and a negligible function ν such that for all k, all $f \in \mathcal{F}_k$, all auxiliary inputs z, and all predicates π, we have*

$$\mathsf{Pr}\left[A(\mathcal{O}(f), z) = \pi(f, z)\right] - \mathsf{Pr}\left[S^{f(\cdot)}(z) = \pi(f, z)\right] \leq \nu(k).$$

Definition 4.5 (Goldwasser and Kalai's Obfuscation w.r.t. independent auxiliary input [16, Definition 4])**.** *An obfuscator $\mathcal{O}$ for a function family $\mathcal{F} = (\mathcal{F}_k)$ is* secure with respect to independent *auxiliary input if for every PPT adversary A, there exist a PPT simulator S and a negligible function ν such that for all k, all auxiliary inputs z, and all predicates π, we have*

$$\mathsf{Pr}\left[A(\mathcal{O}(f), z) = \pi(f, z)\right] - \mathsf{Pr}\left[S^{f(\cdot)}(z) = \pi(f, z)\right] \leq \nu(k),$$

where the probability is over $f \xleftarrow{\$} \mathcal{F}_k$ and the internal coins of A and S.

Firstly, our definitions require security w.r.t. a randomly chosen key from a given set, whereas [3, Definition 2.1] and [16, Definition 3] demand security for every key in that set. In that sense, our definitions are a relaxation (although this does not protect our definitions from impossibility results for general-purpose obfuscation; see below). On the other hand, our definitions require a multi-bit output from the simulator, whereas [3, Definition 2.1] and [16, Definition 3] restrict adversary and simulator to a one-bit output. In that sense our definitions are harder to satisfy and obfuscations satisfying [16, Definition 3] (which is stronger than [3, Definition 2.1]) do not necessarily satisfy our definitions.

Of more interest to us is a comparison with [16, Definition 4], which is also relaxed in the sense that it only requires security for a *random* function chosen from some distribution.[5] Nonetheless, we will show a simple example of a predicate-leaking obfuscation secure in the sense of Definition 4.1, so it seems the two definitions are incomparable.

Secondly, the definitions from [16] give adversary *as well as simulator* auxiliary information. One of the motivations of [16] to incorporate auxiliary input in their definitions is composability; Although we do not explicitly model such auxiliary information in our definitions, we do consider the composability for Definition 4.2 (in Section 6 we show Definition 4.2 implies obfuscation in the indifferentiability framework, which in turn implies a certain form of composability). (See also the discussion after Definition 4.2 for the role of auxiliary information in our definitions.)

The definition of Hohenberger et al. Definition 2 of Hohenberger et al. [22] also introduces average-case secure obfuscation, dealing with probabilistic functionalities as well. This definition is very similar, but subtly stronger than ours. Since Hohenberger et al. give a positive result of an obfuscation, a stronger definition is more desirable than a weaker one.

For ease of comparison, let us first recast [22, Definition 2] in our notation. For the moment we ignore the functionality requirements. (This is done for simplicity; a discussion is deferred to Appendix A.) We give a slightly different but equivalent formulation (polynomial slowdown is implicit, the adversary in [22] is superfluous):

Definition 4.6 (Hohenberger et al.'s average-case secure obfuscation [22, Definition 2])**.** *An obfuscator $\mathcal{O}$ for a function family $\mathcal{F} = (\mathcal{F}_k)$ is* average-case secure *iff there exists a PPT simulator S such that for every PPT distinguisher D and all auxiliary inputs z, we have that*

$$\Pr\left[f \xleftarrow{\$} \mathcal{F}_k \,:\, D^{f(\cdot)}(\mathcal{O}(f), z) = 1\right] - \Pr\left[f \xleftarrow{\$} \mathcal{F}_k \,:\, D^{f(\cdot)}(S^{f(\cdot)}(z), z) = 1\right]$$

is negligible as a function in k.

From our reformulation, it is already clear that the virtual black box requirements (ours and theirs) are surprisingly similar. Compared to Definition 4.1, there are only two differences. Firstly, our weaker Definition 4.1 uses a different order of quantifiers. Namely, whereas Hohenberger et al. require a universal simulator that works for all distinguishers, we use the more relaxed quantification of allowing the simulator to depend on the distinguisher. (The simple point function obfuscation, Section 5.1, we give for Definition 4.1 actually does come with a universal simulator.)

Secondly we do not take into account auxiliary information (although of course it is easy to change our definition so it does). As mentioned before, the main advantage of including auxiliary information in the definition is that it makes the obfuscation more robust in case of composition of the obfuscation with other protocols. We will come back to the issue of composability and the role our stronger Definition 4.2 plays in it in more detail in Section 6.

Perfectly one-way hashing and point functions. We note that a distribution on the function to obfuscate was already considered in other definitions, such as in the security definition for perfect one-way hashing (that is actually an obfuscation of a point function) from Canetti [8]. In that

[5] The relaxation is necessary to obtain independence of the auxiliary information; Goldwasser and Kalai justify it by the observation that in most cryptographic applications, an adversary is confronted with such a randomly chosen (obfuscated) function. This motivation is similar to ours.

case security could be achieved as long as the distribution on the functions is *well-spread*, which basically means that a brute-force search for the function has only negligible success. Our results from Section 5.1 (that also concern an obfuscation of a point function) are formulated with a *uniform* distribution on the functions. (We note that, of course, also perfect one-way hashing can and has been considered with respect to a uniform input distribution, in which case interesting results can be derived from weaker assumptions, see Canetti et al. [10].)

In contrast to the analysis from [8, 10], the analysis of our construction is quite straightforward. Our obfuscation of a point function P_x is $\Pi(x)$ for a one-way permutation Π. Also, the obfuscation security experiment for P and the one-wayness experiment of Π can be related in a very direct manner. However, there *can* be well-spread distributions (different from the uniform one) for which our point function obfuscation becomes insecure. (Imagine a one-way permutation that leaks the upper half of the preimage, and a distribution that keeps the lower half of the preimage constant.) In other words, the price to pay for the simplicity of our analysis is the dependency on a *uniform* distribution of the function.

Also, the constructions from [8, 10] are "semantically secure" in the sense that any predicate on the hashed value (i.e., the key of the point function to be obfuscated) is hidden. Our construction from Section 5.1 does not guarantee this. Just like the one-way permutation that is employed, our construction only hides the key in its entirety. In some applications this might not be sufficient, and in particular not a meaningful "idealization" of a point function. However, in other settings, this may be exactly the idealization one is interested in.

Example: point functions/password queries. With respect to obfuscating point functions in view of implementing a password query (see Section 1 for motivation on this), Canetti's definition and our definition can hence be nicely compared:

- Canetti demands that as long as there is *some* uncertainty about the password, no predicate on the password can be guessed from its obfuscation alone. In particular, the password itself cannot be guessed. Formally, as long as the password has significant min-entropy, no predicate of the password can be guessed from its obfuscation significantly better than without the obfuscation.

- The variation of Canetti's definition with respect to uniform distributions (as used in one result of [10]) requires that if there is *no* a-priori information about the password, then no predicate of the password can be guessed. Formally, if the password has full min-entropy, no predicate on the password can be guessed from its obfuscation significantly better than without the obfuscation.

- We demand that if there is *no* a-priori information about the password, then it cannot be guessed from its obfuscation alone. Formally, if the password has full min-entropy, its obfuscation looks like that of any other password.

This shows that, of course, our notion is considerably weaker than Canetti's (even when considered for uniform input distribution).

Other similar definitions. Technically, our Definition 4.1 is quite similar to Hada [19], Definition 10 (the latter definition which is also formulated with a distribution on the keys). Essentially, the only difference is that [19, Definition 10] equips the distinguisher with an extra copy of the obfuscation instead of oracle access to the function. As argued by Hada [19], this leads to a very strong definition (that is in particular strictly more restrictive than ours).

Finally, the definitions from Wee [34], Section 5.2 are technically similar to ours, in that they are simulation-based. His definitions suffer from strong impossibility results (in particular, a function must be *exactly* learnable for obfuscation). This is partly due to the fact that these definitions demand security for *all* keys in a given set. In our case, a function must be *approximately* learnable for obfuscation, and this enables, e.g., the obfuscation of point functions (see Section 5.1).

4.4 Specific vs. General-Purpose Obfuscation

Impossibility of general-purpose obfuscation. As already indicated our definitions still suffer from certain impossibility results. First, the argument of Barak et al. [3], Section 3 works also for the case of a randomized key distribution, and hence there *are* certain (albeit constructed) examples of unobfuscatable function families. There are even less constructed examples, as we will show in Section 5. In other words: there can be no general-purpose obfuscation.[6]

Specific obfuscators. What we advocate here is to consider *specific* obfuscators for *specific* function families. For example, we will show (in Section 5.3) that obfuscating the encryption algorithm of a private-key encryption scheme yields a public-key encryption scheme, and that such obfuscations (in principle at least) exist. However, our example that such obfuscations exist assumes a public-key encryption scheme in the first place. Plugging this example into the private-key→public-key transformation gives (nearly) the same public-key encryption scheme one started with. So the following question arises:

What is gained? Firstly, the private-key→public-key transformation can be seen, similarly to Diffie and Hellman [11], as a technical paradigm to realize public-key encryption in the first place. In that context, a formalization of obfuscation can provide an interface and a technical guideline of what to aim for.

Secondly, the mentioned impossibility results do not exclude that a sensible formulation of *what* can be obfuscated exists. In other words, there may be a large and easily characterizable class of functions which *can* be obfuscated. Universal, general-purpose obfuscators for this class may exist and provide solutions for applications which correspond to functions inside this class.

5 Results for Our Definitions

Overview. Here we investigate the usefulness of our obfuscation definitions, where we concentrate mainly on the weaker of the two, Definition 4.1. Concretely, we show that this definition is:

- **weak enough** to be achieved with simple constructions: point functions can be easily obfuscated according to our definition (Section 5.1);
- **strong enough** to be used as a useful building block: secure obfuscations according to our definition can be used to transform private-key encryption into public-key encryption (Section 5.3).

In the process, we will encounter a number of concrete and natural examples of obfuscatable and unobfuscatable function families:

- **obfuscatable** according to our definition are point functions, certain private-key encryption schemes, and certain message authentication codes, whereas
- **unobfuscatable** according to our definition are pseudorandom functions, certain private-key encryption schemes, and certain message authentication codes.

[6] It is actually worse: as in [3], there exist function families that cannot be obfuscated with *any* obfuscator.

Our interpretation of these results is that one should not strive for all-purpose obfuscators, since they may not exist (even for restricted tasks such as private-key encryption schemes). Instead, obfuscating very specific function families may often be possible and may prove useful for larger constructions.

5.1 Achievability: Obfuscating Point Functions

Definition 5.1 (Point function)**.** *For $k \in \mathbb{N}$ and $x \in \{0,1\}^k$, we define the* point function $P_x : \{0,1\}^k \to \{0,1\}$ *by $P_x(x') := 1$ iff $x' = x$. Furthermore, we define the families of point functions $\mathcal{P}_k := (P_x)_{x\in\{0,1\}^k}$ and $\mathcal{P} := (\mathcal{P}_k)_{k\in\mathbb{N}}$.*

Our goal is to obfuscate the function family $\mathcal{P}$ according to Definition 4.1. That is, we want to obfuscate the evaluation of a point function P_x sampled uniformly from $\mathcal{P}_k$ for k being the security parameter. As it turns out, $\Pi(x)$ is a secure obfuscation of P_x whenever Π is a one-way permutation. Hence, we make the following definitions:

Definition 5.2 (One-way permutation)**.** *Let $\Pi : \{0,1\}^* \to \{0,1\}^*$ be length-preserving and bijective (i.e., $\Pi(\{0,1\}^k) = \{0,1\}^k$ for all $k \in \mathbb{N}$). We say that Π is a* one-way permutation *iff for all PPT adversaries A, the function $\mathsf{Adv}^{\mathsf{ow}}_{\Pi,A}(k)$ is negligible in k, where*

$$\mathsf{Adv}^{\mathsf{ow}}_{\Pi,A}(k) := \mathsf{Pr}\left[x \overset{\$}{\leftarrow} \{0,1\}^k \,:\, A(\Pi(x)) = x\right].$$

Construction 5.3 (Point function obfuscator)**.** Let $\mathcal{P} = (\mathcal{P}_k)_{k\in\mathbb{N}}$ be the family of point functions from Definition 5.1, and let Π be a one-way permutation as in Definition 5.2. For $P_x \in \mathcal{P}_k$, define the obfuscation $\mathcal{O}(P_x) := \Pi(x)$, with the semantics that $(\mathcal{O}(P_x))(x') = 1$ iff $\Pi(x') = \mathcal{O}(P_x)$.

The actual proof that $\mathcal{O}$ securely obfuscates $\mathcal{P}$ is quite simple:

Theorem 5.4 (Security of Construction 5.3)**.** *$\mathcal{O}$ from Construction 5.3 securely obfuscates the point function family $\mathcal{P}$ from Definition 5.1 in the sense of Definition 4.1.*

Proof. Let an arbitrary PPT distinguisher D as in Definition 4.1 be given. We define a PPT simulator S as follows: S uniformly samples $x' \in \{0,1\}^k$ and outputs $\Pi(x')$. We have to show that

$$\mathsf{Adv}^{\mathsf{sbvbb}}_{\mathcal{P},\mathcal{O},D,S}(k) = \mathsf{Pr}\left[x \overset{\$}{\leftarrow} \{0,1\}^k \,:\, D^{P_x(\cdot)}(\Pi(x)) = 1\right] - \mathsf{Pr}\left[x, x' \overset{\$}{\leftarrow} \{0,1\}^k \,:\, D^{P_x(\cdot)}(\Pi(x')) = 1\right]$$

is negligible in k. We use a game-based proof technique for clarity.

So let **Game** 0 denote the execution of $D^{P_x(\cdot)}(\Pi(x))$ for uniformly chosen $x \in \{0,1\}^k$. By definition,

$$\mathsf{Pr}\left[out_0 = 1\right] = \mathsf{Pr}\left[x \overset{\$}{\leftarrow} \{0,1\}^k \,:\, D^{P_x(\cdot)}(\Pi(x)) = 1\right]$$

with out_0 being D's output in Game 0.

In **Game** 1, we change D's oracle P_x to the all-zero oracle that outputs 0 on every input. We claim that

$$\left|\mathsf{Pr}\left[out_1 = 1\right] - \mathsf{Pr}\left[out_0 = 1\right]\right| \leq \mathsf{Adv}^{\mathsf{ow}}_{\Pi,A}(k) \tag{3}$$

for a suitable PPT adversary A on Π's one-way property and the output out_1 of D in Game 1. To show (3), let bad denote the event that D queries its oracle P_x on input x. Clearly, unless bad occurs, Game 0 and Game 1 proceed identically (in particular, the probability for bad in Game 0 and Game 1 is identical). On the other hand, we have that

$$\mathsf{Pr}\left[\mathsf{bad}\right] \leq \mathsf{Adv}^{\mathsf{ow}}_{\Pi,A}(k) \tag{4}$$

for the following adversary A: On input $\Pi(x)$ it simulates $D^{P_x(\cdot)}(\Pi(x))$, implementing the P_x-oracle using $\Pi(x)$. Concretely, a P_x-query x' is answered with 1 iff $\Pi(x') = \Pi(x)$. If bad occurs (i.e., if D queries P_x with x), then A outputs x. By definition, this adversary establishes (4) and hence (3).

In **Game** 2, we substitute D's all-zero oracle from Game 1 with an oracle that evaluates the point function $P_{x'}$ for a uniformly and independently chosen $x' \in \{0,1\}^k$. Game 1 and Game 2 differ only if D queries x'. Since x' is independently chosen and information-theoretically hidden from D, we have that

$$\left|\mathsf{Pr}\left[out_2 = 1\right] - \mathsf{Pr}\left[out_1 = 1\right]\right| \leq q(k)/2^k,$$

where $q = q(k)$ is a polynomial upper bound on the number of D's oracle queries, and out_2 denotes D's output in Game 2.

Now observe that

$$\mathsf{Pr}\left[out_2 = 1\right] = \mathsf{Pr}\left[x, x' \xleftarrow{\$} \{0,1\}^k \,:\, D^{P_{x'}(\cdot)}(\Pi(x)) = 1\right]$$
$$= \mathsf{Pr}\left[x, x' \xleftarrow{\$} \{0,1\}^k \,:\, D^{P_x(\cdot)}(\Pi(x')) = 1\right].$$

Taking things together proves that

$$\mathsf{Adv}^{\mathsf{sbvbb}}_{\mathcal{P},\mathcal{O},D,S}(k) \leq \mathsf{Adv}^{\mathsf{ow}}_{\Pi,A}(k) + q(k)/2^k$$

is negligible as desired. □

We note that it is easy to see that Construction 5.3 does not *strongly* obfuscate $\mathcal{P}$ (in the sense of Definition 4.2). Moreover, with a little tweak we can also show that Construction 5.3 does not imply the predicate-based Definition 4.5 (even when not taking auxiliary information into consideration). Given a one-way permutation Π define the related permutation Π' by $\Pi'(b||x) = b||\Pi(x)$ (where b a bit). Then Π' is one-way iff Π is, yet the point function obfuscation using Π' clearly leaks the first bit of its input (hence a predicate).

On the weakness of our construction. Construction 5.3 provides significantly weaker secrecy guarantees than previous constructions for point function obfuscations (such as [8, 10, 34]). For instance, our obfuscation $\mathcal{O}(P_x) = \Pi(x)$ might well leak, say, the first half of the bits of the secret point x. The only guarantee we provide is that the whole point x cannot be reconstructed from $\mathcal{O}(P_x)$. In strong contrast to that, e.g., Canetti [8] (and similarly [10, 34]) aims at an "all-or-nothing" property, even for arbitrary distributions of x, and in presence of arbitrary auxiliary information about x. Namely, either the whole secret point x can already be guessed without the obfuscation (in which case the obfuscation of P_x merely provides a confirmation of x), or the obfuscation leaks no additional information about x. And even when assuming a uniform distribution on x (as, e.g., for one construction in [10]), Canetti's definition requires that not even a predicate (such as a bit of x) can be approximated from an obfuscation. This is essentially the secrecy guarantee a random oracle would provide in place of Π in Construction 5.3; and in fact, the main goal of [8] is to provide a computational instantiation of (the secrecy properties of) a random oracle. These extremely strong secrecy guarantees are bought at a certain price: the construction of [8] requires a much stronger and nonstandard computational assumption. (We note that one construction of [10], that provides security only for the uniform distribution on x, makes only standard computational assumptions; however, their construction is comparatively involved.)

So in a nutshell, Construction 5.3 does not provide security in the presence of auxiliary information (about x), in contrast to previous constructions. As a consequence, Construction 5.3 should only be used in contexts in which no auxiliary information about x is used. This effect illustrates the general composability limitations of Definition 4.1 and an alternative based on Definition 4.2 will be discussed in detail in Section 6.

5.2 A Natural Example of an Unobfuscatable Function Family

Despite our good start with point functions, we can show that our obfuscation notion is not free from impossibility results, not even for specific (and natural) classes of function families. Concretely, we will prove that pseudorandom functions cannot be obfuscated under our obfuscation definitions. Intuitively, this is not at all surprising, independently of the used obfuscation notion: obfuscating a pseudorandom function essentially yields a random oracle.

The upcoming definition follows Goldreich et al. [14].

Definition 5.5 (Family of pseudorandom functions)**.** *Let $\mathcal{F} = (\mathcal{F}_k)_{k\in\mathbb{N}}$, $\mathcal{F}_k = (f_s)_{s\in\{0,1\}^k}$ with $f_s : \{0,1\}^{|s|} \to \{0,1\}^{|s|}$ be given. Let $\mathcal{R} = (\mathcal{R}_k)_{k\in\mathbb{N}}$, where $\mathcal{R}_k$ denotes the set of all functions $\{0,1\}^k \to \{0,1\}^k$. Then $\mathcal{F}$ is called a* family of pseudorandom functions *iff for every PPT A, the function $\mathsf{Adv}^{\mathsf{prf}}_{\mathcal{F},A}(k)$ is negligible in k. Here,*

$$\mathsf{Adv}^{\mathsf{prf}}_{\mathcal{F},A}(k) := \mathsf{Pr}\left[s \overset{\$}{\leftarrow} \{0,1\}^k \,:\, A^{f_s(\cdot)}(\varepsilon) = 1\right] - \mathsf{Pr}\left[R \overset{\$}{\leftarrow} \mathcal{R}_k \,:\, A^{R(\cdot)}(\varepsilon) = 1\right].$$

Theorem 5.6 (Pseudorandom function families cannot be obfuscated)**.** *Let $\mathcal{F}$ be a family of pseudorandom functions as in Definition 5.5. Then $\mathcal{F}$ cannot be obfuscated in the sense of Definition 4.1, and with perfect functionality.*

Proof. Let $\mathcal{O}$ be an obfuscator of $\mathcal{F}$ in the sense of Definition 4.1, and with perfect functionality. Consider the distinguisher D that, upon input O and with oracle access to f_s, proceeds as follows. D uniformly chooses $x \in \{0,1\}^k$ and outputs 1 iff $O(x) = f_s(x)$. Clearly,

$$\mathsf{Pr}\left[D^{f_s(\cdot)}(\mathcal{O}(f_s)) = 1\right] = 1$$

by functionality of the obfuscation. Now fix any PPT simulator S in the sense of Definition 4.1. We have that

$$\mathsf{Pr}\left[D^{f_s(\cdot)}(S^{f_s(\cdot)}(\varepsilon)) = 1\right] \overset{\mathsf{c}}{\approx} \mathsf{Pr}\left[D^{R(\cdot)}(S^{R(\cdot)}(\varepsilon)) = 1\right]$$

for a truly random function R as in Definition 5.5. Furthermore, $\mathsf{Pr}\left[D^{R(\cdot)}(S^{R(\cdot)}(\varepsilon)) = 1\right]$ is negligible by the statistical properties of R. To see this, note that unless S queries $R(x)$ for the point $x \in \{0,1\}^k$ chosen by D, we have that S's view is independent of $R(x)$. Furthermore, since $x \in \{0,1\}^k$ is chosen independently by D, we have that S queries $R(x)$ only with negligible probability. Hence, S produces an output O with $O(x) = R(x)$ only with negligible probability. Thus,

$$\mathsf{Pr}\left[D^{f_s(\cdot)}(S^{f_s(\cdot)}(\varepsilon)) = 1\right] \overset{\mathsf{c}}{\approx} \mathsf{Pr}\left[D^{R(\cdot)}(S^{R(\cdot)}(\varepsilon)) = 1\right] \overset{\mathsf{c}}{\approx} 0,$$

which shows that $\mathsf{Adv}^{\mathsf{sbvbb}}_{\mathcal{F},\mathcal{O},D,S}(k)$ is non-negligible, overwhelming even, and so $\mathcal{O}$ does not obfuscate $\mathcal{F}$. □

5.3 How to Use Our Definition: Transforming Private-Key Encryption into Public-Key Encryption

We now exemplify that our obfuscation definition is strong enough to be useful as a building block. Concretely, we take up the motivation of Diffie and Hellman [11], who suggested that one way to produce a public-key encryption scheme was to obfuscate a private-key scheme. This application of obfuscation was also suggested by Barak et al. [3]. Specifically, say we obfuscate the encryption algorithm (with hardwired private key) of a private-key encryption scheme, and call the result the *public key*. Intuitively, the public key then allows encrypting messages, but *nothing more*. We will investigate below when this transformation actually yields a secure public-key encryption scheme.

Encryption schemes. We start by recalling some standard definitions, starting with the definition of a private-key encryption scheme:

Definition 5.7 (Private-key encryption scheme). *A* private-key encryption scheme $\mathsf{SKE} = (\mathsf{K}, \mathsf{E}, \mathsf{D})$ *consists of three PPT algorithms with the following semantics:*

- *The* key generation algorithm K *samples a key K. We write $K \leftarrow \mathsf{K}(\varepsilon)$ and let $\mathcal{K}_k$ denote the set of all keys K in the range of $\mathsf{K}(\varepsilon)$ on security parameter k.*
- *The* encryption algorithm E *encrypts a message $M \in \{0,1\}^*$ and produces a ciphertext C. We write $C \leftarrow \mathsf{E}(K, M)$.*
- *The* decryption algorithm D *decrypts a ciphertext C to a message M. We write $M \leftarrow \mathsf{D}(K, C)$.*

We require perfect correctness *of the scheme, i.e., that $\mathsf{D}(K, \mathsf{E}(K, M)) = M$ for all $M \in \{0,1\}^*$ and all possible $K \leftarrow \mathsf{K}(\varepsilon)$.*

To ease presentation, and to maintain compatibility with our definitional choices for obfuscation, we will assume (without loss of generality) that K samples keys *uniformly* from $\mathcal{K}_k$.

The definition of a public-key encryption scheme is identical, except that encryption is performed with a *public key*, and decryption is performed with a *private key*:

Definition 5.8 (Public-key encryption scheme). *A* public-key encryption scheme $\mathsf{PKE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *consists of three PPT algorithms with the following semantics:*

- *The* key generation algorithm Gen *samples a keypair (pk, sk) consisting of a public key pk along with a private key sk. We write $(pk, sk) \leftarrow \mathsf{Gen}(\varepsilon)$.*
- *The* encryption algorithm Enc *encrypts a message $M \in \{0,1\}^*$ and produces a ciphertext C. We write $C \leftarrow \mathsf{Enc}(pk, M)$.*
- *The* decryption algorithm Dec *decrypts a ciphertext C to a message M. We write $M \leftarrow \mathsf{Dec}(sk, C)$.*

We require perfect correctness *of the scheme, i.e., that $\mathsf{Dec}(sk, \mathsf{Enc}(pk, M)) = M$ for all $M \in \{0,1\}^*$ and all possible $(pk, sk) \leftarrow \mathsf{Gen}(\varepsilon)$.*

We stress that we model encryption schemes which encrypt *arbitrary messages* $M \in \{0,1\}^*$ (as opposed to, say, messages $M \in G$ from a cyclic group). This definitional choice has been made only to ease presentation.[7]

Security of encryption schemes. To capture the security of a (private-key or public-key) encryption scheme, we require indistinguishability of ciphertexts (Goldwasser and Micali [17], Naor and Yung [29], Rackoff and Simon [32]). The following definition captures active attacks (aka chosen-ciphertext attacks, in which an adversary has access to a decryption oracle). A straightforward variant models passive attacks (in which an adversary only gets to observe a ciphertext).

[7] Strictly speaking an infinite message space is not compatible with our computational model where all algorithms are required to run in time polynomial in the security parameter.

Definition 5.9 (Security of an encryption scheme)**.** *Let* $\mathsf{SKE} = (\mathsf{K}, \mathsf{E}, \mathsf{D})$ *be a private-key encryption scheme, and let* $A = (A_1, A_2)$ *be a pair of PPT algorithms we call* adversary*. We define*

$$\mathsf{Adv}^{\mathsf{ind\text{-}cca}}_{\mathsf{SKE},A}(k) := \mathsf{Pr}\left[\mathsf{Exp}^{\mathsf{ind\text{-}cca}}_{\mathsf{SKE},A}(k) = 1\right] - 1/2,$$

where $\mathsf{Exp}^{\mathsf{ind\text{-}cca}}_{\mathsf{SKE},A}(k)$ *is the following experiment:*

```
**Experiment** $\mathsf{Exp}^{\mathsf{ind\text{-}cca}}_{\mathsf{SKE},A}(k)$
    Choose uniformly $b \in \{0,1\}$
    $K \leftarrow \mathsf{K}(\varepsilon)$
    $(M_0, M_1, s) \leftarrow A_1^{\mathsf{E}(K,\cdot),\mathsf{D}(K,\cdot)}(\varepsilon)$
    $C^* \leftarrow \mathsf{E}(K, M_b)$
    $b' \leftarrow A_2^{\mathsf{E}(K,\cdot),\mathsf{D}(K,\cdot)}(C^*, s)$
    Return $b \oplus b'$
```

A_1 *is restricted to always output equal-length messages* M_0, M_1*, and* A_2 *is restricted not to query its decryption oracle* $\mathsf{D}(K, \cdot)$ *on the challenge ciphertext* C^**.* SKE *is* indistinguishable against chosen-ciphertext attacks (IND-CCA) *if the advantage function* $\mathsf{Adv}^{\mathsf{ind\text{-}cca}}_{\mathsf{SKE},A}(k)$ *is negligible in k for all A.*

Define further

$$\mathsf{Adv}^{\mathsf{ind\text{-}cpa}}_{\mathsf{SKE},A}(k) := \mathsf{Pr}\left[\mathsf{Exp}^{\mathsf{ind\text{-}cpa}}_{\mathsf{SKE},A}(k) = 1\right] - 1/2,$$

where $\mathsf{Exp}^{\mathsf{ind\text{-}cpa}}_{\mathsf{SKE},A}(k)$ *is defined like* $\mathsf{Exp}^{\mathsf{ind\text{-}cca}}_{\mathsf{SKE},A}(k)$*, with the difference that* A_1 *and* A_2 *do not get access to a decryption oracle* $\mathsf{D}(K, \cdot)$*.* SKE *is* indistinguishable against chosen-plaintext attacks (IND-CPA) *if the advantage function* $\mathsf{Adv}^{\mathsf{ind\text{-}cpa}}_{\mathsf{SKE},A}(k)$ *is negligible in k for all A.*

For a public-key encryption scheme $\mathsf{PKE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$*, define the experiments* $\mathsf{Exp}^{\mathsf{ind\text{-}cca}}_{\mathsf{PKE},A}(k)$ *and* $\mathsf{Exp}^{\mathsf{ind\text{-}cpa}}_{\mathsf{PKE},A}(k)$ *as above, with the difference that A does not get access to an encryption oracle* $\mathsf{Enc}(pk, \cdot)$*, but instead,* A_1 *gets the public key pk as input.* PKE *is IND-CCA (resp., IND-CPA) if* $\mathsf{Adv}^{\mathsf{ind\text{-}cca}}_{\mathsf{PKE},A}(k)$ *(resp.,* $\mathsf{Adv}^{\mathsf{ind\text{-}cpa}}_{\mathsf{PKE},A}(k)$*) is negligible for all A.*

The transformation. We are now ready to formally define what we mean by obfuscating the encryption algorithm of a private-key encryption scheme.

Definition 5.10 ((Strongly) obfuscatable private-key encryption)**.** *Let* $\mathsf{SKE} = (\mathsf{K}, \mathsf{E}, \mathsf{D})$ *be a private-key encryption scheme as in Definition 5.7. Then, define* $\mathsf{E}_K(\cdot) := \mathsf{E}(K, \cdot)$ *as the encryption algorithm of* SKE *with hardwired private key K. Let* $\mathcal{E}_k := (\mathsf{E}_K)_{K \in \mathcal{K}_k}$ *and* $\mathcal{E} := (\mathcal{E}_k)_{k \in \mathbb{N}}$*. Suppose that* $\mathcal{O}$ *is an obfuscator for the family* $\mathcal{E}$*, such that* $\mathcal{O}$ *satisfies Definition 4.1 and has perfect functionality. Then we say that* $\mathcal{O}$ obfuscates SKE*. If* $\mathcal{O}$ *even satisfies Definition 4.2, then we say that* $\mathcal{O}$ strongly obfuscates SKE*. If an* $\mathcal{O}$ *exists that (strongly) obfuscates* SKE*, then we say that* SKE *is* (strongly) obfuscatable*.*

Construction 5.11 (Obfuscating private-key encryption)**.** Let $\mathsf{SKE} = (\mathsf{K}, \mathsf{E}, \mathsf{D})$ and $\mathcal{O}$ as in Definition 5.10, such that $\mathcal{O}$ obfuscates SKE. Then, define the following public key encryption scheme $\mathsf{PKE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ as follows:

- $\mathsf{Gen}(\varepsilon)$ samples $K \leftarrow \mathsf{K}(\varepsilon)$, sets $pk \leftarrow \mathcal{O}(\mathsf{E}_K)$ and $sk \leftarrow K$, and outputs the keypair (pk, sk).
- $\mathsf{Enc}(pk, M)$ interprets pk as an algorithm, computes $C \leftarrow pk(M)$, and returns C.
- $\mathsf{Dec}(sk, C)$ computes $M \leftarrow \mathsf{D}(sk, C)$ and returns M.

Since we require perfect functionality from an obfuscation, it is clear that the scheme PKE from Construction 5.11 fulfils the correctness requirement from Definition 5.8. However, for the analysis of PKE, the following questions are much more interesting:

- Is PKE secure (in the sense of Definition 5.9) if SKE is?
- Is it realistic to assume that one can obfuscate the encryption of SKE in the first place?

In the following, we will try to answer these questions. Concretely, we will show that PKE is IND-CPA secure if SKE is; however, we will also illustrate that this does not hold in general for IND-CCA security, unless $\mathcal{O}$ *strongly* obfuscates SKE. We will also show that there *exist* private-key encryption schemes with (strongly) obfuscatable encryption algorithm; however, there also exist encryption schemes which are *not* obfuscatable.

What security properties our transformation preserves. We start by showing that our transformation preserves IND-CPA security.

Theorem 5.12 (Obfuscating private-key encryption preserves IND-CPA)**.** *Let* SKE*,* $\mathcal{O}$*, and* PKE *be as in Construction 5.11. Then the transformed public-key encryption scheme* PKE *is IND-CPA secure whenever* SKE *is.*

Proof. Let $A = (A_1, A_2)$ be an adversary on PKE's IND-CPA property as in Definition 5.9. We need to show that $\mathsf{Adv}^{\mathsf{ind\text{-}cpa}}_{\mathsf{PKE},A}(k)$ is negligible in k. We proceed in games.

Let **Game** 0 denote $\mathsf{Exp}^{\mathsf{ind\text{-}cpa}}_{\mathsf{PKE},A}(k)$. We have

$$\mathsf{Pr}\left[out_0 = 1\right] = \mathsf{Pr}\left[\mathsf{Exp}^{\mathsf{ind\text{-}cpa}}_{\mathsf{PKE},A}(k) = 1\right]$$

if we let out_0 denote the experiment output in Game 0.

In **Game** 1, we change the generation of the challenge ciphertext C^*. Recall that in Game 0, $C^* = \mathsf{Enc}(pk, M_b) = (\mathcal{O}(\mathsf{E}_K))(M_b)$. We now set $C^* = \mathsf{E}_K(M_b) = \mathsf{E}(K, M_b)$. Since the obfuscator $\mathcal{O}$ satisfies perfect functionality, we have $(\mathcal{O}(\mathsf{E}_K))(M_b) = \mathsf{E}_K(M_b)$ always, and so

$$\mathsf{Pr}\left[out_1 = 1\right] = \mathsf{Pr}\left[out_0 = 1\right]$$

for the experiment output out_1 in Game 1.

To define **Game** 2, we interpret Game 1 as a PPT distinguisher D as in Definition 4.1. Concretely, D has oracle access to $\mathsf{E}_K(\cdot) = \mathsf{E}(K, \cdot)$, gets as input $pk = \mathcal{O}(\mathsf{E}_K)$, and internally simulates Game 1. Observe that D does not need explicit knowledge about K, but instead can pass its input pk to A_1 and use its oracle $\mathsf{E}_K(\cdot)$ to generate $C^* = \mathsf{E}_K(M_b)$. Finally, D outputs the experiment output out_1. Definition 4.1 guarantees the existence of a PPT simulator S such that

$$\mathsf{Pr}\left[out_1 = 1\right] = \mathsf{Pr}\left[D^{\mathsf{E}_K(\cdot)}(\mathcal{O}(\mathsf{E}_K)) = 1\right] \overset{\mathsf{c}}{\approx} \mathsf{Pr}\left[D^{\mathsf{E}_K(\cdot)}(S^{\mathsf{E}_K(\cdot)}(\varepsilon)) = 1\right].$$

Using S, we define Game 2 as a modification of Game 1 as follows: in Game 2, we now generate pk as $pk := S^{\mathsf{E}_K(\cdot)}(\varepsilon)$; the remaining execution is as in Game 1. We have

$$\mathsf{Pr}\left[out_2 = 1\right] = \mathsf{Pr}\left[D^{\mathsf{E}_K(\cdot)}(S^{\mathsf{E}_K(\cdot)}(\varepsilon)) = 1\right]$$

for the experiment output out_2 in Game 2, and so $\mathsf{Pr}\left[out_2\right] \overset{\mathsf{c}}{\approx} \mathsf{Pr}\left[out_1\right]$.

Finally, observe that in Game 2, the public key $pk = S^{\mathsf{E}_K(\cdot)}(\varepsilon)$ can be generated from oracle access to $\mathsf{E}_K(\cdot) = \mathsf{E}(K, \cdot)$ alone. Hence, we can modify A_1 into A_1' such that A_1' no longer expects a public key as input, but instead requires only oracle access to $\mathsf{E}(K, \cdot)$ to generate pk on its own. Furthermore, recall that we generate C^* using E directly. Hence, we can interpret $A' := (A_1', A_2)$ as a PPT adversary on SKE's IND-CPA security, and we get

$$\mathsf{Pr}\left[\mathsf{Exp}^{\mathsf{ind\text{-}cpa}}_{\mathsf{SKE},A'}(k) = 1\right] = \mathsf{Pr}\left[out_2 = 1\right].$$

Since SKE is IND-CPA secure by assumption, we have that

$$\Pr\left[\mathsf{Exp}^{\mathsf{ind\text{-}cpa}}_{\mathsf{PKE},A}(k)=1\right]=\Pr\left[out_0\right]=\Pr\left[out_1\right]\overset{c}{\approx}\Pr\left[out_2\right]=\Pr\left[\mathsf{Exp}^{\mathsf{ind\text{-}cpa}}_{\mathsf{SKE},A'}(k)=1\right]\overset{c}{\approx}1/2,$$

which proves that $\mathsf{Adv}^{\mathsf{ind\text{-}cpa}}_{\mathsf{PKE},A}(k)$ is negligible as desired. □

If we assume that $\mathcal{O}$ *strongly* obfuscates SKE, we can even show that IND-CCA security is preserved:

Theorem 5.13 (Strongly obfuscating private-key encryption preserves IND-CCA)**.** *Let* SKE*,* $\mathcal{O}$*, and* PKE *as in Construction 5.11, and such that* $\mathcal{O}$ *strongly obfuscates* SKE*. Assume here that the description of* E_K *allows to extract the key* K*.*[8] *Then, the transformed public-key encryption scheme* PKE *is IND-CCA secure whenever* SKE *is.*

Proof. The proof is identical to the proof of Theorem 5.12, except that D now needs to simulate an IND-CCA experiment instead of an IND-CPA experiment. Since we assumed that $\mathcal{O}$ strongly obfuscates SKE, D can do so with using its input E_K, which by assumptions allows to extract the (encryption and decryption) key K. The rest of the proof remains unchanged. □

What security properties our transformation does not preserve. Unfortunately IND-CCA security of SKE is not necessarily preserved by Construction 5.11 in case $\mathcal{O}$ does not *strongly* obfuscate SKE. The intuitive reason is the following. Assume an obfuscator $\mathcal{O}$ in the sense of Definition 4.1 for the probabilistic functions $\mathsf{E}_K(\cdot)=\mathsf{E}(K,\cdot)$. In Definition 4.1, the distinguisher D gets an encryption oracle (namely, an oracle for evaluating the obfuscated function $\mathsf{E}_K(\cdot)$), but no means of decrypting ciphertexts. (This is in contrast to our stronger Definition 4.2 which grants D the full key K.) It is now conceivable that the obfuscation loses its security *only* in the presence of a decryption oracle. In particular, in the IND-CCA security experiment with a transformed encryption scheme PKE constructed from SKE and $\mathcal{O}$, the adversary *does* have access to a decryption oracle, and security of the obfuscation can no longer be guaranteed. Hence, we cannot apply the reasoning of Theorem 5.12 to show that PKE inherits SKE's IND-CCA security when the obfuscator $\mathcal{O}$ does not strongly obfuscate SKE. More generally, we can construct the following counterexample:

Theorem 5.14 ((Not strongly) obfuscating private-key encryption does not preserve IND-CCA)**.** *Assume that obfuscatable IND-CCA secure private-key encryption schemes exist. Then there exists an IND-CCA secure private-key encryption scheme* SKE *and an obfuscator* $\mathcal{O}$ *such that the following holds:*

- $\mathcal{O}$ *obfuscates (but not strongly)* SKE*'s encryption algorithm as in Construction 5.11.*
- *The public-key encryption scheme* PKE *obtained by Construction 5.11 is not IND-CCA.*

Proof. Assume an IND-CCA secure private-key encryption scheme SKE' that is obfuscatable in the sense of Definition 5.10. Say that $\mathcal{O}'$ obfuscates SKE'. We modify $\mathsf{SKE}'=(\mathsf{K}',\mathsf{E}',\mathsf{D}')$ into a scheme $\mathsf{SKE}=(\mathsf{K},\mathsf{E},\mathsf{D})$ as follows:

- K samples $K'\leftarrow\mathsf{K}'$, chooses uniformly $R\in\{0,1\}^k$ and outputs $K=(K',R)$.
- $\mathsf{E}(K,M)$ parses $K=(K',R)$, generates a ciphertext $C'\leftarrow\mathsf{E}'(K',M)$ and outputs $C=(C',\varepsilon)$.
- $\mathsf{D}(K,C)$ parses $K=(K',R)$ and $C=(C',T)$ and determines its output M as follows:
  - If $T=\varepsilon$, then $M\leftarrow\mathsf{D}'(K',C')$.
  - If $T=R$, then $M=K$.
  - Otherwise, $M=\perp$.

[8] We stress that this is a (natural) assumption about the *representation* of E_K, not the encryption algorithm itself.

Assume an adversary $A = (A_1, A_2)$ on SKE's IND-CCA property. We construct an adversary $A' = (A'_1, A'_2)$ on SKE''s IND-CCA property, such that

$$\mathsf{Adv}^{\mathsf{ind\text{-}cca}}_{\mathsf{SKE},A}(k) \overset{\mathrm{c}}{\approx} \mathsf{Adv}^{\mathsf{ind\text{-}cca}}_{\mathsf{SKE}',A'}(k). \tag{5}$$

Now A' proceeds like A, translating A's $\mathsf{E}(K, \cdot)$- and $\mathsf{D}(K, \cdot)$-queries into $\mathsf{E}'(K', \cdot)$- and $\mathsf{D}'(K', \cdot)$-queries for A''s own oracles as follows:

- $\mathsf{E}(K, M)$-queries are answered with $(\mathsf{E}'(K', M), \varepsilon)$.
- $\mathsf{D}(K, (C', T))$-queries are answered with $\mathsf{D}'(K', C')$ if $T = \varepsilon$, and with $\perp$ otherwise.

Let bad denote the event that A asks for a decryption of some $C = (C', T)$ with $T = R$. It is clear that A''s emulation of oracles $\mathsf{E}(K, \cdot)$ and $\mathsf{D}(K, \cdot)$ is perfect unless bad occurs. On the other hand, the probability that bad occurs is at most $q(k)/2^k$ for some polynomial q, since R is information-theoretically hidden from A' and thus A. Hence, (5) follows and so SKE achieves IND-CCA security.

By assumption, $\mathcal{O}'$ obfuscates SKE'. We construct an obfuscator $\mathcal{O}$ for SKE's encryption algorithm $\mathsf{E}_K(\cdot) = \mathsf{E}(K, \cdot)$ as follows: $\mathcal{O}(\mathsf{E}_K)$ parses $K = (K', R)$ and internally runs $O' \leftarrow \mathcal{O}'(\mathsf{E}'_{K'})$. $\mathcal{O}$ then outputs an obfuscation (O, R), where O is obtained from O' by adding a second ciphertext component $T = \varepsilon$ to each encryption. We claim that $\mathcal{O}$ obfuscates SKE. Indeed, assume a PPT distinguisher D on $\mathcal{O}$'s virtual black-box property. Using trivial syntactic modifications, we can construct from D a distinguisher D' on $\mathcal{O}'$'s virtual black-box property, so that

$$\mathsf{Pr}\left[D^{\mathsf{E}_K(\cdot)}(\mathcal{O}(\mathsf{E}_K)) = 1\right] = \mathsf{Pr}\left[D'^{\mathsf{E}'_{K'}(\cdot)}(\mathcal{O}'(\mathsf{E}'_{K'})) = 1\right] \overset{\mathrm{c}}{\approx} \mathsf{Pr}\left[D'^{\mathsf{E}'_{K'}(\cdot)}(S'^{\mathsf{E}'_{K'}(\cdot)}(\varepsilon)) = 1\right]. \tag{6}$$

Here, S' is a PPT simulator whose existence follows from our assumption that $\mathcal{O}'$ obfuscates SKE'. From S', we can construct a PPT simulator S that runs $O' \leftarrow S'(\varepsilon)$ and outputs a simulated obfuscation $(O, \overline{R})$. Here, $\overline{R} \in \{0,1\}^k$ is uniformly chosen by S and O is obtained from O' as above, by adding a second ciphertext component $T = \varepsilon$. Now by definition, D's encryption oracle E_K is independent of the actual value of R, so inventing $\overline{R}$ achieves

$$\mathsf{Pr}\left[D^{\mathsf{E}_K(\cdot)}(S^{\mathsf{E}_K(\cdot)}(\varepsilon)) = 1\right] = \mathsf{Pr}\left[D'^{\mathsf{E}'_{K'}(\cdot)}(S'^{\mathsf{E}'_{K'}(\cdot)}(\varepsilon)) = 1\right]. \tag{7}$$

Taking (6) and (7) together shows that

$$\mathsf{Adv}^{\mathsf{sbvbb}}_{\mathcal{E},\mathcal{O},D,S}(k) = \mathsf{Pr}\left[D^{\mathsf{E}_K(\cdot)}(\mathcal{O}(\mathsf{E}_K)) = 1\right] - \mathsf{Pr}\left[D^{\mathsf{E}_K(\cdot)}(S^{\mathsf{E}_K(\cdot)}(\varepsilon)) = 1\right]$$

is negligible. Hence, $\mathcal{O}$ obfuscates SKE.

It remains to show that the public-key encryption scheme PKE, as obtained from SKE and $\mathcal{O}$ using Construction 5.11, is not IND-CCA secure. To this end, consider the following trivial adversary $A = (A_1, A_2)$ on PKE's IND-CCA property. A_1 gets as input $pk = \mathcal{O}(\mathsf{E}_K) = (O, R)$ and then queries its decryption oracle $\mathsf{Dec}(sk, \cdot)$ on input $C = (\varepsilon, R)$. By definition, $\mathsf{Dec}(sk, C)$ outputs $\mathsf{D}(sk, (\varepsilon, R)) = K$, which can be used to decrypt any challenge ciphertext. Hence, $\mathsf{Adv}^{\mathsf{ind\text{-}cca}}_{\mathsf{PKE},A}(k) = 1/2$, and so PKE is not IND-CCA secure. □

Can private-key encryption be obfuscated? We now show that there exist private-key encryption schemes which can be obfuscated, even strongly; however, we also show that there exist schemes which cannot be obfuscated. But before we explain our own results, we discuss a connected result.

On a negative result due to Barak et al. Barak et al. [3], Theorem 4.12 in the full version, give a transformation that turns any private-key encryption scheme into an equally secure unobfuscatable private-key encryption scheme SKE. The definition of unobfuscatability employed here implies that *any* obfuscation of SKE's encryption algorithm allows extracting the private key K. This also rules out obfuscation with respect to our definitions for private-key encryption schemes meeting any reasonable security standard. (Essentially a simulator should be able to construct a decryption algorithm using a polynomial number of encryption queries: this breaks IND-CPA security.) To achieve their strong result, Barak et al. employ a sophisticated modular construction based on "totally unobfuscatable one-way functions." Our construction of an unobfuscatable private-key encryption scheme below is conceptually much easier, but only works for our security definitions.

(Strongly) obfuscatable private-key encryption schemes exist. Given the preceding discussion, the most pressing question is whether there exist private-key encryption schemes which can be obfuscated according to our definition. Trivially, the answer is yes as already observed by Barak et al. [3], since any public-key encryption scheme can be turned into an obfuscatable private-key scheme. Essentially, the obfuscation is the public key. Formally:

Theorem 5.15 ((Strongly) obfuscatable private-key encryption schemes exist)**.** *Assume that IND-CPA (resp., IND-CCA) secure public-key encryption schemes exist. Then there exist IND-CPA (resp., IND-CCA) secure private-key encryption schemes which are strongly obfuscatable.*

Proof. Let $\mathsf{PKE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ be an IND-CPA (resp., IND-CCA) secure private-key encryption scheme. Without loss of generality, we assume that Gen always uses $p(k)$ random coins for a polynomial p. Then, we interpret PKE as a private-key encryption scheme $\mathsf{SKE} = (\mathsf{K}, \mathsf{E}, \mathsf{D})$ as follows:

- $\mathsf{K}(\varepsilon)$ outputs a uniformly chosen $K \in \{0,1\}^{p(k)}$.
- $\mathsf{E}(K, M)$ uses K as random coins for Gen to deterministically obtain a keypair (pk, sk). Then, E returns[9] $C \leftarrow (pk, \mathsf{Enc}(pk, M))$.
- $\mathsf{D}(K, C)$ uses K as random coins for Gen to deterministically obtain a keypair (pk, sk), and parses $C = (pk', C')$. If $pk = pk'$, then D returns $M \leftarrow \mathsf{Dec}(sk, C)$; else, D returns $\perp$.

Using a merely syntactic reduction, it is clear that SKE is IND-CPA (resp., IND-CCA) whenever PKE is. To show that SKE is obfuscatable, consider the obfuscator $\mathcal{O}$ with $\mathcal{O}(\mathsf{E}_K) = pk$ for the public key pk obtained by running Gen with random coins K. Now a PPT simulator S in the sense of Definition 4.2 can simply obtain pk by its own encryption oracle $\mathsf{E}_K(\cdot)$ and perfectly simulate an obfuscation. Hence, $\mathcal{O}$ strongly obfuscates SKE. □

We stress that Theorem 5.15 is not very useful in conjunction with Construction 5.11. Namely, Theorem 5.15 shows that any public-key encryption scheme can be interpreted as an obfuscatable private-key encryption scheme; Construction 5.11 states that any obfuscatable private-key encryption scheme gives rise to a public-key encryption scheme. Even worse, plugging any public-key encryption scheme into Theorem 5.15 and then into Construction 5.11, one ends up with essentially the original scheme. However, the point of Theorem 5.15 is a structural one: it shows that *some* private-key encryption schemes *can* be obfuscated, and in particular, there can be no generic impossibilities for obfuscating private-key encryption schemes according to our definition.

Unobfuscatable private-key encryption schemes exist. Unfortunately, also private-key encryption schemes which are unobfuscatable exist. We stress that by unobfuscatable, we mean "not

[9] The trick to include pk in each ciphertext to achieve trivial obfuscatability was suggested by a TCC referee.

obfuscatable" in the sense of Definition 5.10. We do *not* mean "totally unobfuscatable" as in Barak et al. [3] (cf. also the discussion about their negative result above).

The idea to construct unobfuscatable private-key encryption schemes is simple: sign each ciphertext with a digital signature scheme, and include the signature verification key in each ciphertext. This way, the encryption is authenticated, in a publicly verifiable way. In particular, any obfuscation of the encryption algorithm can be used to generate fresh signatures for new ciphertexts. In the setting of Definition 4.1 this means that a simulator S has to essentially forge signatures for fresh encryptions. (Since D has direct oracle access to the encryption algorithm, it can obtain the "right" verification key independently of S's output.) This contradicts the security of the signature scheme. Formally:

Theorem 5.16 (Unobfuscatable private-key encryption schemes exist)**.** *Assume that one-way functions exist. Then an IND-CCA secure private-key encryption exists which is not obfuscatable in the sense of Definition 5.10.*

Proof. Let $\mathsf{SKE}' = (\mathsf{K}', \mathsf{E}', \mathsf{D}')$ be an IND-CPA secure private-key encryption scheme, and let $\mathsf{SIG} = (\mathsf{Gen}, \mathsf{Sig}, \mathsf{Ver})$ be an EUF-CMA secure digital signature scheme (see Definition 5.19). These ingredients can be constructed from one-way functions (see, e.g., Bellare et al. [6] and Rompel [33]). We construct a private-key encryption scheme $\mathsf{SKE} = (\mathsf{K}, \mathsf{E}, \mathsf{D})$ which cannot be obfuscated:

- $\mathsf{K}(\varepsilon)$ runs $K' \leftarrow \mathsf{K}'(\varepsilon)$ and $(\mathit{verkey}, \mathit{sigkey}) \leftarrow \mathsf{Gen}(\varepsilon)$, and outputs $K = (K', \mathit{verkey}, \mathit{sigkey})$.
- $\mathsf{E}(K, M)$ parses $K = (K', \mathit{verkey}, \mathit{sigkey})$, computes $C' = \mathsf{E}'(K', M)$, signs C' via $\sigma \leftarrow \mathsf{Sig}(\mathit{sigkey}, C')$, and outputs $C = (C', \sigma, \mathit{verkey})$.
- $\mathsf{D}(K, C)$ parses $K = (K', \mathit{verkey}, \mathit{sigkey})$ and $C = (C', \sigma, \overline{\mathit{verkey}})$. If $\overline{\mathit{verkey}} = \mathit{verkey}$ and $\mathsf{Ver}(\mathit{verkey}, \sigma, C') = 1$, then D outputs $\mathsf{D}'(K', C')$; otherwise, D outputs $\perp$.

It is easy to see that SKE constitutes an authenticated encryption scheme, and consequently achieves IND-CCA security (see, e.g., Bellare and Namprempre [5], Katz and Yung [24]). Now consider an arbitrary obfuscator $\mathcal{O}$ for SKE. Furthermore, consider the following distinguisher D in the sense of Definition 4.1:

1. First, D uses its oracle E_K to encrypt $M_0 = 0$ and so obtain a ciphertext $C_0 = (C_0', \sigma_0, \mathit{verkey})$.
2. Then, D interprets its input O as an (encryption) algorithm and obtains a ciphertext $C_1 = (C_1', \sigma_1, \mathit{verkey}')$ as an encryption of $M_1 = 1$.
3. D outputs 1 iff $\mathsf{Ver}(\mathit{verkey}, \sigma_1, C_1') = 1$ and $C_1' \neq C_0'$.

By construction, $D^{\mathsf{E}_K(\cdot)}(\mathcal{O}(\mathsf{E}_K))$ always outputs 1, since $C_0' \neq C_1'$ by correctness of the encryption scheme. On the other hand, for any fixed PPT simulator S, we have that

$$\mathsf{Pr}\left[D^{\mathsf{E}_K(\cdot)}(S^{\mathsf{E}_K(\cdot)}(\varepsilon)) = 1\right] \leq \mathsf{Adv}^{\mathsf{euf\text{-}cma}}_{\mathsf{SIG},A}(k) \qquad (8)$$

for the adversary A on SIG that proceeds as follows. Namely, A internally runs $D^{\mathsf{E}_K(\cdot)}(S^{\mathsf{E}_K(\cdot)}(\varepsilon))$ and implements the E_K oracles on its own, choosing a key K' for SKE' on its own, and using its own challenge verification key and signature oracle to sign ciphertexts. If D outputs 1, this means that it has constructed a valid signature σ_1 for a fresh message C_1'. The given bound (8) follows.

But since $\mathsf{Adv}^{\mathsf{euf\text{-}cma}}_{\mathsf{SIG},A}(k)$ is negligible by assumption on SIG, we obtain that $\mathsf{Adv}^{\mathsf{sbvbb}}_{\mathcal{E},\mathcal{O},D,S}(k)$ is overwhelming, so that $\mathcal{O}$ does not obfuscate SKE. The claim follows. □

5.4 Another Example: From Message Authentication to Digital Signatures

Message authentication codes (MACs) are the private-key analogue to digital signature schemes. While verification of signatures is public in a digital signature scheme, the verification algorithm of a MAC requires a private key. In particular in view of the results of Section 5.3, it is now tempting to convert a MAC into a signature scheme by obfuscating the verification algorithm (with hardwired key). It will turn out that this does not work as easily as in the encryption case, due to two circumstances. Firstly, the authentication analogue of IND-CPA security is not very meaningful;[10] hence we omit it. Secondly, the interface of the verification algorithm is very restricted: verification only outputs a bit, and only outputs 1 in case of a valid signature.

Consequently, our results concerning MACs and digital signatures are mainly negative. Concretely, we show that

- the situation is non-trivial, i.e., there exist obfuscatable as well as unobfuscatable MACs (Theorem 5.25 and Theorem 5.26; here, "obfuscatable" means that the verification algorithm can be obfuscated),
- a *strongly* (in the sense of Definition 4.2) obfuscatable MAC would give rise to a secure digital signature scheme (Theorem 5.22), while an obfuscatable MAC is not enough (Theorem 5.22),
- however, there exist no strongly obfuscatable MACs (at least in the standard model; see Theorem 5.24).

Summarizing, our results suggest that MACs cannot be turned into signatures schemes as smoothly as private-key into public-key encryption schemes. While our results do not imply that such a transformation is impossible, they suggest that both Definition 4.1 and Definition 4.2 are unsuitable as technical tools for such a transformation. Namely, Definition 4.1 is too weak to guarantee security of the obtained digital signature scheme in general, whereas Definition 4.2 cannot be achieved in case of MACs.[11] We believe that a definition suitable for this transformation would have to be case-tailored (e.g., incorporating black-box access to a signature oracle).

Message authentication codes. Again, we start by recalling some standard definitions.

Definition 5.17 (Message authentication code (MAC))**.** *A* message authentication code (MAC) $\mathsf{MAC} = (\mathsf{K}, \mathsf{S}, \mathsf{V})$ *consists of three PPT algorithms with the following semantics:*

- *The* key generation algorithm K *samples a key K. We write $K \leftarrow \mathsf{K}(\varepsilon)$, where $\mathcal{K}_k$ denotes the set of all keys K in the range of $\mathsf{K}(\varepsilon)$ on security parameter k.*
- *The* signature algorithm S *signs a message $M \in \{0,1\}^*$ and produces a signature σ. We write $\sigma \leftarrow \mathsf{S}(K, M)$.*
- *The* verification algorithm V *verifies a signature σ for a message M. We write $ver \leftarrow \mathsf{V}(K, \sigma, M)$, where $ver \in \{0,1\}$.*

We require perfect correctness*, namely that $\mathsf{V}(K, \mathsf{S}(K, M), M) = 1$ for all $M \in \{0,1\}^*$ and all possible $K \leftarrow \mathsf{K}(\varepsilon)$.*

Again, we assume for simplicity that K samples its keys K uniformly from $\mathcal{K}_k$.

Digital signature schemes. The definition of a digital signature scheme is almost identical:

Definition 5.18 (Digital signature scheme)**.** *A* digital signature scheme $\mathsf{SIG} = (\mathsf{Gen}, \mathsf{Sig}, \mathsf{Ver})$ *consists of three PPT algorithms with the following semantics:*

- *The* key generation algorithm Gen *samples a keypair $(verkey, sigkey)$ consisting of a verification key $verkey$ along with a signing key $sigkey$. We write $(verkey, sigkey) \leftarrow \mathsf{Gen}(\varepsilon)$.*
- *The* signature algorithm Sig *signs a message $M \in \{0,1\}^*$ and produces a signature σ. We write $\sigma \leftarrow \mathsf{Sig}(sigkey, M)$.*

[10] We would like to thank Salil Vadhan for pointing this out to us.

[11] At least in the standard model; it is conceivable that strongly obfuscatable MACs exist, e.g., in the random oracle model.

- *The* verification algorithm Ver *verifies a signature σ for a message M. We write $ver \leftarrow \mathsf{Ver}(verkey, \sigma, M)$, where $ver \in \{0,1\}$.*

We require perfect correctness *of the scheme, i.e., that* $\mathsf{Ver}(verkey, \mathsf{Sig}(sigkey, M), M) = 1$ *for all $M \in \{0,1\}^*$ and all possible $(verkey, sigkey) \leftarrow \mathsf{Gen}(\varepsilon)$.*

Security of signatures. We demand that, even with access to a signing oracle, one cannot forge signatures of new messages.

Definition 5.19 (Security of a MAC/signature scheme)**.** *Let* $\mathsf{MAC} = (\mathsf{K}, \mathsf{S}, \mathsf{V})$ *be a message authentication code, and let A be a PPT algorithm we call* adversary*. We define*

$$\mathsf{Adv}^{\mathsf{euf\text{-}cma}}_{\mathsf{MAC},A}(k) := \mathsf{Pr}\left[\mathsf{Exp}^{\mathsf{euf\text{-}cma}}_{\mathsf{MAC},A}(k) = 1\right],$$

where $\mathsf{Exp}^{\mathsf{euf\text{-}cma}}_{\mathsf{MAC},A}(k)$ *is the following experiment:*

```
Experiment $\mathsf{Exp}^{\mathsf{euf\text{-}cma}}_{\mathsf{MAC},A}(k)$
  $K \leftarrow \mathsf{K}(\varepsilon)$
  $(\sigma, M) \leftarrow A^{\mathsf{S}(K,\cdot),\mathsf{V}(K,\cdot,\cdot)}(\varepsilon)$
  Return $\mathsf{V}(K, \sigma, M)$
```

A is restricted to never return a message M for which it has requested a signature from $\mathsf{S}(sigkey, \cdot)$. We call MAC existentially unforgeable under chosen-message attacks (EUF-CMA) *if* $\mathsf{Adv}^{\mathsf{euf\text{-}cma}}_{\mathsf{MAC},A}(k)$ *is negligible in k for all A.*

For a digital signature scheme $\mathsf{SIG} = (\mathsf{Gen}, \mathsf{Sig}, \mathsf{Ver})$*, define the experiment* $\mathsf{Exp}^{\mathsf{euf\text{-}cma}}_{\mathsf{SIG},A}(k)$ *as above, with the difference that A does not get access to a verification oracle* $\mathsf{Ver}(verkey, \cdot, \cdot)$*, but instead gets the verification key verkey as input.* SIG *is EUF-CMA if* $\mathsf{Adv}^{\mathsf{euf\text{-}cma}}_{\mathsf{SIG},A}(k)$ *is negligible for all A.*

The transformation. We now define what we mean by obfuscating the verification algorithm of a message authentication code:

Definition 5.20 ((Strongly) obfuscatable MAC)**.** *Let* $\mathsf{MAC} = (\mathsf{K}, \mathsf{S}, \mathsf{V})$ *be a message authentication code as in Definition 5.17. Then, define $\mathsf{V}_K(\cdot,\cdot) := \mathsf{V}(K,\cdot,\cdot)$ as the verification algorithm of* SIG *with hardwired private key K. Let $\mathcal{V}_k := (\mathsf{V}_K)_{K \in \mathcal{K}_k}$ and $\mathcal{V} := (\mathcal{V}_k)_{k \in \mathbb{N}}$. Suppose that $\mathcal{O}$ is an obfuscator for the family $\mathcal{V}$, such that $\mathcal{O}$ satisfies Definition 4.1 and has perfect functionality. Then we say that $\mathcal{O}$* obfuscates MAC*. If $\mathcal{O}$ even satisfies Definition 4.2, then we say that $\mathcal{O}$* strongly obfuscates MAC*. If an $\mathcal{O}$ exists that (strongly) obfuscates* MAC*, then we say that* MAC *is* (strongly) obfuscatable*.*

Construction 5.21 (Obfuscating a MAC)**.** Let $\mathsf{MAC} = (\mathsf{K}, \mathsf{S}, \mathsf{V})$ and $\mathcal{O}$ as in Definition 5.20, such that $\mathcal{O}$ obfuscates MAC. Then, define the following digital signature scheme $\mathsf{SIG} = (\mathsf{Gen}, \mathsf{Sig}, \mathsf{Ver})$ as follows:

- $\mathsf{Gen}(\varepsilon)$ samples $K \leftarrow \mathsf{Gen}(\varepsilon)$, sets $verkey \leftarrow \mathcal{O}(\mathsf{V}_K)$ and $sigkey \leftarrow K$, and outputs the keypair $(verkey, sigkey)$.
- $\mathsf{Sig}(sigkey, M)$ computes $\sigma \leftarrow \mathsf{S}(sigkey, M)$ and returns σ.
- $\mathsf{Ver}(verkey, \sigma, M)$ interprets $verkey$ as an algorithm, computes $ver \leftarrow verkey(\sigma, M)$, and returns ver.

Since we require perfect functionality from an obfuscation, it is clear that SIG fulfils the correctness requirement from Definition 5.18.

When obfuscating a MAC preserves EUF-CMA security. The proof of the following theorem is analogous to the proof of Theorem 5.12, resp. Theorem 5.13, so we omit it.

Theorem 5.22 (Strongly obfuscating a MAC preserves EUF-CMA)**.** *Let* MAC*,* $\mathcal{O}$*, and* SIG *as in Construction 5.21, and such that* $\mathcal{O}$ *strongly obfuscates* MAC*. Then, the transformed digital signature scheme* SIG *is EUF-CMA secure whenever* MAC *is.*

Analogously to Theorem 5.14 (and with a similar proof), it can be shown that obfuscation in the sense of Definition 4.1 is not enough:

Theorem 5.23 ((Not strongly) obfuscating a MAC does not preserve EUF-CMA)**.** *Assume that obfuscatable EUF-CMA secure MAC exists. Then there exists an EUF-CMA secure MAC* MAC *and an obfuscator* $\mathcal{O}$ *such that the following holds:*

- $\mathcal{O}$ *obfuscates (but not strongly)* MAC*'s verification algorithm as in Construction 5.21.*
- *The digital signature scheme* SIG *obtained by Construction 5.21 is not EUF-CMA.*

Strongly obfuscatable MACs do not exist. Intuition says that any digital signature scheme is, when interpreted as a message authentication code, obfuscatable. (The obfuscation is simply the verification key.) However, there is a crucial difference to the encryption setting: there, we modified encryption so as to include the public encryption key. A simulator could then obtain this public key through oracle access to the encryption function, and output the public key as a perfect obfuscation. In the authentication case, there is no way to include the public verification key as part of the verification output: the verification algorithm outputs only bits, and a 1-output means that a signature is valid. Hence, if verification outputs 1 too carelessly, the scheme becomes forgeable. In fact, the EUF-CMA security of the scheme implies that the simulator essentially always receives 0-answers from its verification oracle. Hence, the verification oracle is useless to the simulator, and the simulated obfuscation does not depend on the used signing key. So if the distinguisher can generate valid signatures, it can distinguish a real from a simulated obfuscation.[12] We formalize this in Theorem 5.24 below. As an aside, the same proof can be used to show impossibility according to a weaker version of Definition 4.2, where the simulator is allowed to depend on the distinguisher.

Theorem 5.24 (Strongly obfuscatable MACs do not exist)**.** *Let* $\mathsf{MAC} = (\mathsf{K}, \mathsf{S}, \mathsf{V})$ *be an EUF-CMA secure message authentication code. Let* $\mathsf{V}_K(\cdot,\cdot) := \mathsf{V}(K,\cdot,\cdot)$ *be the verification algorithm of* MAC *with hardwired private key* K*. Let* $\mathcal{V}_k := (\mathsf{V}_K)_{K\in\mathcal{K}_k}$ *and* $\mathcal{V} := (\mathcal{V}_k)_{k\in\mathbb{N}}$*. Then no obfuscator* $\mathcal{O}$ *for* $\mathcal{V}$ *achieves Definition 4.2.*

Proof. Fix any PPT simulator S as in Definition 4.2. Consider the following PPT distinguisher D, that gets as input the description of a function V_K (which includes the key K), and an obfuscation O (which is either produced as $O \leftarrow \mathcal{O}(\mathsf{V}_K)$ or as $O \leftarrow S^{\mathsf{V}_K(\cdot,\cdot)}(\varepsilon)$).

1. D independently chooses another key $K' \overset{\$}{\leftarrow} \mathcal{K}_k$.
2. D signs a message $M = 0$ according to K and K' via $\sigma \leftarrow \mathsf{S}(K, 0)$ and $\sigma' \leftarrow \mathsf{S}(K', 0)$.
3. D uses its second input O to check both signatures via $ver \leftarrow O(\sigma, 0)$ and $ver' \leftarrow O(\sigma', 0)$.
4. D outputs 1 iff $ver = 1$ and $ver' = 0$.

Our first claim about D is that D outputs 1 with overwhelming probability when $O \leftarrow \mathcal{O}(\mathsf{V}_K)$. Indeed, when $O \leftarrow \mathcal{O}(\mathsf{V}_K)$, then the functionality of $\mathcal{O}$ and the correctness of MAC guarantee that $ver = 1$. Furthermore, $ver' = 1$ implies that D produced a signature σ' using an independent key K', but σ' turned out to be valid for key K. Hence, D essentially forged a signature, and so

$$\Pr\left[ver' = 1\right] \leq \mathsf{Adv}^{\mathsf{euf\text{-}cma}}_{\mathsf{MAC},A}(k)$$

[12] The idea to distinguish real from simulated obfuscations using honestly generated signatures was also remarked by a Journal of Cryptology referee.

is negligible by assumption about MAC, where A is an adversary that chooses independently a key K' and outputs a signature $\sigma' \leftarrow \mathsf{S}(K', 0)$ for 0. Summarizing,

$$\Pr\left[D(\mathsf{V}_K, \mathcal{O}(\mathsf{V}_K)) = 1\right] = 1 - \Pr\left[ver' = 1\right]$$

is overwhelming.

Conversely, assume $O \leftarrow S^{\mathsf{V}_K(\cdot,\cdot)}(\varepsilon)$. Let bad denote the event that S queries its verification oracle V_K with a signature σ and a message M such that $\mathsf{V}_K(\sigma, M)$ returns 1. Clearly, bad implies that S forged a signature from oracle access to the verification algorithm only, so that

$$\Pr\left[\mathsf{bad}\right] \leq \mathsf{Adv}^{\mathsf{euf\text{-}cma}}_{\mathsf{MAC},S}(k)$$

is negligible by assumption about MAC. In case bad does not occur, however, S only receives 0s as oracle answers from V_K, and so its output O must be independent of K. Hence, for the signatures σ and σ' produced as above, $O(\sigma, 0)$ and $O(\sigma', 0)$ are identically independently distributed with

$$\Pr\left[O(\sigma, 0) = 1 \mid \neg\mathsf{bad}\right] = \Pr\left[O(\sigma', 0) = 1 \mid \neg\mathsf{bad}\right]$$

so that

$$\begin{aligned}\Pr\left[D(\mathsf{V}_K, S^{\mathsf{V}_K(\cdot,\cdot)}(\varepsilon)) = 1\right] &\leq \Pr\left[O(\sigma, 0) = 1 \mid \neg\mathsf{bad}\right](1 - \Pr\left[O(\sigma', 0) = 1 \mid \neg\mathsf{bad}\right]) + \Pr\left[\mathsf{bad}\right] \\ &\leq 1/4 + \Pr\left[\mathsf{bad}\right].\end{aligned}$$

This shows that D successfully distinguishes real from fake obfuscations, and so $\mathcal{O}$ does not achieve Definition 4.2. □

Obfuscatable and unobfuscatable MACs exist. The proof of Theorem 5.24 utilizes that D has access to the key K and can produce signatures under key K. In Definition 4.1, the weaker one of our definitions, a similar argument is not possible, since D only gets oracle access to the *verification algorithm* V_K. (And if MAC is secure, then oracle access to the verification algorithm alone does not allow to produce signatures.)

To construct obfuscatable MACs (in the sense of Definition 4.1), our escape is to let the simulator output a *different*, freshly sampled verification key as obfuscation. Since a distinguisher in the sense of Definition 4.1 only has oracle access to the verification algorithm, it cannot produce a signature for which the verification oracle outputs 1. Hence, the distinguisher's views with a real obfuscation and the simulator's output are identical except with negligible probability. Formally:

Theorem 5.25 (Obfuscatable MACs exist). *Assume that EUF-CMA secure signature schemes exist. Then there exist EUF-CMA secure MACs which are obfuscatable in the sense of Definition 5.20.*

Proof. Let $\mathsf{SIG} = (\mathsf{Gen}, \mathsf{Sig}, \mathsf{Ver})$ be an EUF-CMA secure signature scheme. Without loss of generality, we assume that Gen always uses $p(k)$ random coins for a polynomial p. Then, we interpret SIG as a MAC $\mathsf{MAC} = (\mathsf{K}, \mathsf{S}, \mathsf{V})$ as follows:

- $\mathsf{K}(\varepsilon)$ outputs a uniformly chosen $K \in \{0,1\}^{p(k)}$.
- $\mathsf{S}(K, M)$ uses K as random coins for Gen to deterministically obtain a keypair $(verkey, sigkey)$, and returns $\sigma \leftarrow \mathsf{Sig}(sigkey, M)$.
- $\mathsf{V}(K, \sigma, M)$ uses K as random coins for Gen to deterministically obtain a pair $(verkey, sigkey)$, and returns $ver \leftarrow \mathsf{Ver}(verkey, \sigma, M)$.

Using a merely syntactic reduction to SIG's EUF-CMA security, it is clear that MAC is EUF-CMA secure. To show that SKE is obfuscatable, consider the obfuscator $\mathcal{O}$ with $\mathcal{O}(\mathsf{V}_K) = \mathit{verkey}$ for the verification key *verkey* obtained by running K with random coins K. Consider furthermore the PPT simulator S that outputs a freshly sampled verification key verkey' obtained through $(\mathit{verkey}', \mathit{sigkey}') \leftarrow \mathsf{Gen}(\varepsilon)$. Fix any PPT distinguisher D in the sense of Definition 4.1. Let bad denote the event that D, on input $\mathcal{O}(\mathsf{V}_K) = \mathit{verkey}$ and with oracle access to V_K, queries V_K with a signature σ and a message M such that $\mathsf{V}_K(\sigma, M)$ returns 1. Since bad implies that D forged a signature using a verification key *verkey* only, we have that

$$\mathsf{Pr}\left[\mathsf{bad}\right] \leq \mathsf{Adv}^{\mathsf{euf\text{-}cma}}_{\mathsf{SIG},A}(k)$$

is negligible. Here, A denotes an adversary that internally simulates D and answers D's verification oracle queries using its own verification key *verkey*. Note also that when D gets as input a fake obfuscation $S^{\mathsf{V}_K(\cdot,\cdot)}(\varepsilon) = \mathit{verkey}'$ instead of a real one, the probability for bad does not change. Formally, unless bad occurs, D's oracle calls are all answered with 0, and hence D's view with a real and a fake obfuscation is identical. We get

$$\mathsf{Pr}\left[D^{\mathsf{V}_K(\cdot,\cdot)}(\mathcal{O}(\mathsf{V}_K)) = 1 \mid \neg\mathsf{bad}\right] = \mathsf{Pr}\left[D^{\mathsf{V}_K(\cdot,\cdot)}(S^{\mathsf{V}_K(\cdot,\cdot)}(\varepsilon)) = 1 \mid \neg\mathsf{bad}\right],$$

so that

$$\left|\mathsf{Pr}\left[D^{\mathsf{V}_K(\cdot,\cdot)}(\mathcal{O}(\mathsf{V}_K)) = 1\right] - \mathsf{Pr}\left[D^{\mathsf{V}_K(\cdot,\cdot)}(S^{\mathsf{V}_K(\cdot,\cdot)}(\varepsilon)) = 1\right]\right| \leq \mathsf{Pr}\left[\mathsf{bad}\right] \leq \mathsf{Adv}^{\mathsf{euf\text{-}cma}}_{\mathsf{SIG},A}(k)$$

is negligible as desired. □

We now try to construct a MAC which is unobfuscatable (even according to our weaker obfuscation notion Definition 4.1). Recall that Theorem 5.16 constructs an unobfuscatable private-key encryption scheme by authenticating ciphertexts. This way, a simulator in the sense of Definition 4.1 has to *forge* signatures to generate valid ciphertexts. In the authentication setting, a simulator only has to simulate a verification algorithm which outputs *bits* (instead of bitstrings), hence we must find a different strategy. Also, we must take care that our modifications of the verification algorithm do not damage the unforgeability of our MAC. Facing these difficulties, we resort to non-black-box techniques very similar to those from Barak et al. [3], Section 3. Formally:

Theorem 5.26 (Unobfuscatable MACs exist)**.** *Assume that EUF-CMA secure MACs exist. Then an EUF-CMA secure MAC exists which is not obfuscatable in the sense of Definition 5.20.*

Proof. Let $\mathsf{MAC}' = (\mathsf{K}', \mathsf{S}', \mathsf{V}')$ be an EUF-CMA secure MAC. We construct a MAC $\mathsf{MAC} = (\mathsf{K}, \mathsf{S}, \mathsf{V})$ that cannot be obfuscated:

- $\mathsf{K}(\varepsilon)$ runs $K' \leftarrow \mathsf{K}'(\varepsilon)$, uniformly samples $\alpha \in \{0,1\}^k$ and $\beta = (\beta_1, \ldots, \beta_k) \in \{0,1\}^k$, and returns $K \leftarrow (K', \alpha, \beta)$.
- $\mathsf{S}(K, M)$ parses $K = (K', \alpha, \beta)$, computes $\sigma' \leftarrow \mathsf{S}'(K', M)$ and returns $\sigma \leftarrow (0, \sigma')$.
- $\mathsf{V}(K, \sigma, M)$ parses $K = (K', \alpha, \beta)$ and $\sigma = (i, \sigma')$, and determines its output *ver* as follows:
  - If $i = 0$, then $\mathit{ver} \leftarrow \mathsf{V}'(K', \sigma', M)$.
  - If $1 \leq i \leq k$ and $\sigma' = \alpha$, then $\mathit{ver} \leftarrow \beta_i =$ "the i-th bit of β".
  - If $i = k+1$ and $\sigma'(\alpha) = \beta$ (where σ' is interpreted as an algorithm), then $\mathit{ver} \leftarrow 1$.
  - In all other cases, set $\mathit{ver} \leftarrow 0$.

It is clear that MAC satisfies the correctness requirement of Definition 5.17. Furthermore, for any PPT adversary A, we have

$$\mathsf{Adv}^{\mathsf{euf\text{-}cma}}_{\mathsf{MAC},A} \leq \mathsf{Adv}^{\mathsf{euf\text{-}cma}}_{\mathsf{MAC}',A'} + \mathsf{Pr}\left[\mathsf{bad}\right],$$

where A' is the MAC'-adversary canonically obtained from A by rejecting all signatures of the form (i, σ') with $i \neq 0$, and bad denotes the event that A submits a verification query $(\sigma, M) = ((i, \sigma'), M)$ with $i \neq 0$ but $\mathsf{V}(K, \sigma, M) = 1$. However, $\mathsf{Pr}[\mathsf{bad}]$ is negligible since α and β are information-theoretically hidden from A. Hence, MAC is EUF-CMA secure.

Now consider an arbitrary obfuscator $\mathcal{O}$ for MAC. Furthermore, consider the following distinguisher D in the sense of Definition 4.1:

1. D constructs algorithm O' by concatening k copies of O with some of its inputs already fixed. More precisely, O' will compute the following function

$$O'(x) = (O((1, x), 0), \ldots, O((k, x), 0)),$$

where the range of O' is $\{0, 1\}^k$.

2. D queries its oracle V_K on input $((k + 1, O'), 0)$ and outputs the result.

Now first consider what happens when D receives as input an obfuscation $O = \mathcal{O}(\mathsf{V}_K)$ of MAC's verification algorithm. Then, by construction,

$$O'(\alpha) = (O((i, \alpha), 0)_{i=1}^k = (\mathsf{V}(K, (i, \alpha), 0))_{i=1}^k = (\beta_i)_{i=1}^k = \beta,$$

so that $\mathsf{V}_K((k + 1, O'), 0) = \mathsf{V}(K, (k + 1, O'), 0) = 1$. This implies that $D^{\mathsf{V}_K(\cdot)}(\mathcal{O}(\mathsf{V}_K))$ always outputs 1. On the other hand, for any fixed PPT simulator S, we have that

$$\mathsf{Pr}\left[D^{\mathsf{V}_K(\cdot)}(S^{\mathsf{V}_K(\cdot)}(\varepsilon)) = 1\right] \leq \mathsf{Adv}^{\mathsf{euf\text{-}cma}}_{\mathsf{MAC},A}(k)$$

for the adversary A on MAC that internally simulates D and S by relaying its verification only. Since $\mathsf{Adv}^{\mathsf{euf\text{-}cma}}_{\mathsf{MAC},A}(k)$ is negligible as argued before, $\mathsf{Adv}^{\mathsf{sbvbb}}_{\mathcal{V},\mathcal{O},D,S}(k)$ is overwhelming, so that $\mathcal{O}$ does not obfuscate MAC. The claim follows. □

6 Composable Obfuscators

In Section 5.1, we already noticed the composability defects of our virtual black-box property from Definition 4.1: secure obfuscations may lose their security in larger contexts. Intuitively, our *strong* virtual black-box property from Definition 4.2 should guarantee more: since the distinguisher D gets the to-be-obfuscated function f itself as input, a secure obfuscation stays secure even if auxiliary information about f (efficiently computable from f) is leaked in a larger context.

In this section, we will investigate the compositional properties of Definition 4.2 more closely. It will turn out that Definition 4.2 guarantees indifferentiability (a simulation-based generic notion of security similar to universal composability or reactive simulatability). Indifferentiability provides clean interfaces for the modular design of larger systems that use obfuscation as a tool. In particular, the indifferentiability framework comes with a composition theorem that allows a modular security analysis.

6.1 Indifferentiability

The *indifferentiability framework* of Maurer et al. [27] follows a simulation-based approach (cf. [2, 4, 9, 15, 17, 28]) to define security. Concretely, a cryptographic system is compared to an idealization of the respective protocol task (usually a trusted host that performs the task in an ideal and incorruptible manner). If every attack on the real system has a counterpart attack in the ideal system such that both systems and attacks are indistinguishable, then we say that the real system is secure.

Formally, a (real or ideal) system $S = (\mathsf{pub}_S, \mathsf{priv}_S)$ consists of a public interface pub_S and a private interface priv_S. The private interface priv_S is the input/output interface for honest parties. For instance, in a system for secure message transmission, a private input could be "send message X to Bob," and a private output could be "received message Y from Bob." Conversely, the public interface pub_S interfaces an adversary with the network. For example, in the secure message transmission example, the adversary would receive a ciphertext over the public interface.

To capture the security of a real system, we compare it to a suitable ideal system:

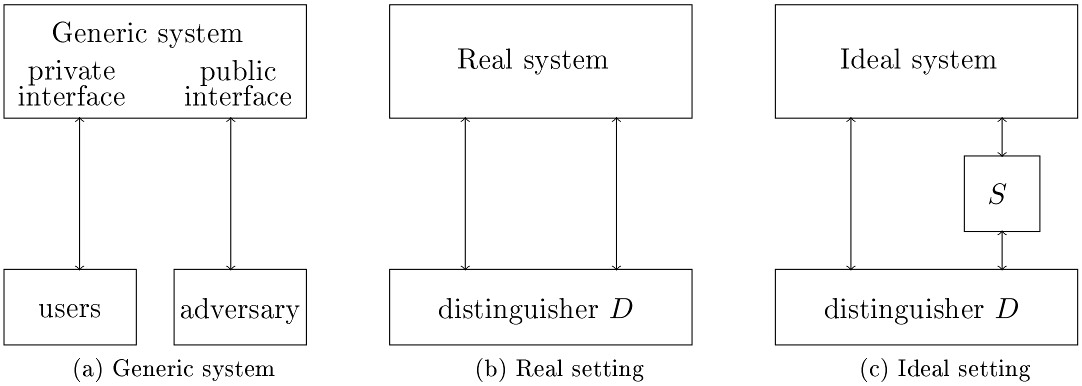
Definition 6.1 (Indifferentiability of systems, sketch)**.** *A system $R = (\mathsf{pub}_R, \mathsf{priv}_R)$ is* indifferentiable *from another system $I = (\mathsf{pub}_I, \mathsf{priv}_I)$ iff there exists a PPT simulator S such that for all PPT distinguishers D, the advantage $\mathsf{Adv}^{\mathsf{indiff}}_{R,I,S,D}(k)$ is negligible in k. Here,*

$$\mathsf{Adv}^{\mathsf{indiff}}_{R,I,S,D}(k) := \Pr\left[D(\mathsf{priv}_R, \mathsf{pub}_R) = 1\right] - \Pr\left[D(\mathsf{priv}_I, S(\mathsf{pub}_I)) = 1\right],$$

where

- *$D(\mathsf{priv}_R, \mathsf{pub}_R)$ denotes the execution of D with access to the interfaces priv_R and pub_R, and*
- *$D(\mathsf{priv}_I, S(\mathsf{pub}_I))$ denotes the execution of D with access to the interface priv_I and to S, where S has access to interface pub_I.*

The situation is illustrated in Figure 1. For a more comprehensive introduction to indifferentiability, we refer to Maurer et al. [27].



(a) Generic system (b) Real setting (c) Ideal setting

Figure 1: Systems in the indifferentiability framework.

On the order of quantifiers. Definition 6.1 deviates from the original security definition in [27] with respect to the order of quantifiers: we demand existence of a universal simulator S that works for all D, whereas [27, Definition 3] only requires the existence of an S for every given D. We chose the stronger order of quantifiers for two reasons:

Stronger composability: Whereas the stronger order of quantifiers ($\exists S \forall D$) provides secure universal composition ([9]), the weaker order of quantifiers ($\forall D \exists S$) does not provide concurrent composability ([20]).[13]

Decomposition of obfuscation definition: Jumping ahead, the stronger order of quantifiers allows us to express the indifferentiable obfuscation definition as a combination of a functionality and a virtual black-box requirement.

[13] The cited works prove composability statements in the frameworks of universal composability and reactive simulatability. However, they do not rely on model specifics and are applicable to indifferentiability as well.

6.2 Our Indifferentiable Obfuscation Definition

To capture obfuscation through indifferentiability, we need to specify a real and an ideal system. The real system should reflect what *really* happens when using an obfuscator, whereas the ideal system should specify what *should* happen. Hence,

the real private interface contains oracle access to an obfuscation $\mathcal{O}(f)$ (since real honest users use the obfuscation only as a black box), as well as a description of the function f itself (so information about f can be used in some other place as well);

the real public interface contains the obfuscation $\mathcal{O}(f)$ (since the obfuscation is public);

the ideal private interface contains oracle access to f (since this is the function that *should be* evaluated), as well as f itself (so that again, f can be used elsewhere);

the ideal public interface contains only oracle access to f (since ideally this is all that should be leaked).

The real and ideal systems for obfuscation are depicted in Figure 2.

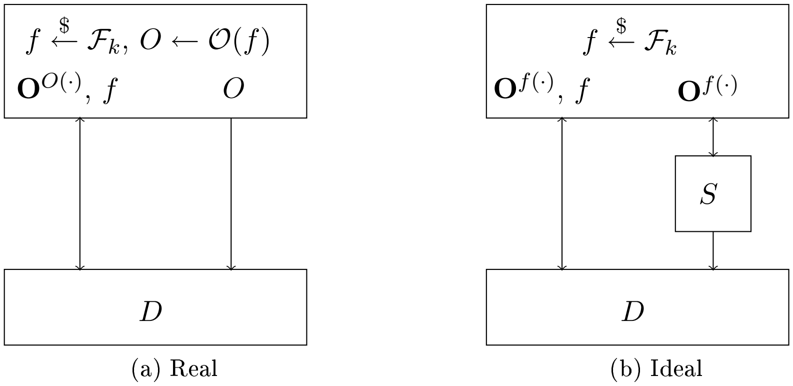

$f \xleftarrow{\$} \mathcal{F}_k$, $O \leftarrow \mathcal{O}(f)$
$\mathbf{O}^{O(\cdot)}$, f
O
D
(a) Real
$f \xleftarrow{\$} \mathcal{F}_k$
$\mathbf{O}^{f(\cdot)}$, f
$\mathbf{O}^{f(\cdot)}$
S
D
(b) Ideal

Figure 2: The real and ideal systems for indifferentiable obfuscation, running with a distinguisher D and a simulator S. Here $\mathbf{O}^{h(\cdot)}$ denotes oracle access to a function h.

Hence, we get the following definition as a special case of Definition 6.1:

Definition 6.2 (Indifferentiable obfuscation). *Let $\mathcal{F} = (\mathcal{F}_k)_{k\in\mathbb{N}}$ be a family of functions. Then $\mathcal{O}$ is an* indifferentiable obfuscator *for $\mathcal{F}$ iff the following holds: there exists a PPT simulator S such that for every PPT distinguisher D, the function $\mathsf{Adv}^{\mathsf{ind\text{-}obf}}_{\mathcal{F},\mathcal{O},S,D}(k)$ is negligible in k. Here,*

$$\mathsf{Adv}^{\mathsf{ind\text{-}obf}}_{\mathcal{F},\mathcal{O},S,D}(k) := \Pr\left[f \xleftarrow{\$} \mathcal{F}_k, O \leftarrow \mathcal{O}(f) \,:\, D^{O(\cdot)}(f,O) = 1\right] - \Pr\left[f \xleftarrow{\$} \mathcal{F}_k \,:\, D^{f(\cdot)}(f, S^{f(\cdot)}(\varepsilon)) = 1\right].$$

On the choice of f. Note that in our ideal obfuscation system, f is chosen at random *by the system itself* (and not, e.g., by a user of the system). This design choice has a number of consequences for the use of the ideal system: it means that a user cannot obfuscate arbitrary functions; instead, the obfuscated function is chosen by the system (but of course made available to the user via the private interface). Hence, we can only model larger systems in which the obfuscated functions (if there is more than one obfuscation) are *independently chosen.* However, while we restrict the *choice* of f, we do not restrict its *usage*: f can be re-used in several places in a larger system, since it is provided to a user as part of the private interface. Any such larger system can then be analyzed modularly, taking advantage of the composition theorem [27, Theorem 1].

As part of this philosophy, it is best (but not necessary) if the description of f given to the distinguisher is as forthright as possible—think the random coins used to sample from $\mathcal{F}_k$—and no hidden secret about f is known or used elsewhere.

What type of composability is implied (and what type is not). Indifferentiability (in the sense of Definition 6.2) implies that one obfuscation can be used in arbitrary larger contexts. (As explained above, these contexts may even make direct use of f.) Using the composition theorem [27, Theorem 1], we can deduce that any constant number of obfuscations can be used concurrently.[14] *However,* recall that in our formulation, the obfuscated function is selected randomly by the system. Hence, the composition theorem only implies that several *different* and *independently selected* obfuscations can be used concurrently. It makes no claim about several different obfuscations *related* functions. However, if the *same* function is obfuscated several times, we can at least say the following:

Theorem 6.3 (Composability of indifferentiable obfuscations of the same function)**.** *Let $\mathcal{F} = (\mathcal{F}_k)_{k\in\mathbb{N}}$ be a family of functions, let $\mathcal{O}$ be an indifferentiable obfuscator for $\mathcal{F}$, and let p be a polynomial. Then there exists a PPT simulator S_p, such that for every PPT distinguisher D_p, the function*

$$\mathsf{Adv}^{\mathsf{ind\text{-}obf\text{-}mult}}_{\mathcal{F},\mathcal{O},S_p,D_p,p}(k) := \Pr\left[f \xleftarrow{\$} \mathcal{F}_k, O_1 \leftarrow \mathcal{O}(f), \ldots, O_{p(k)} \leftarrow \mathcal{O}(f) \,:\, D_p^{O(\cdot)}(f, O_1, \ldots, O_{p(k)}) = 1\right] \\ - \Pr\left[f \xleftarrow{\$} \mathcal{F}_k \,:\, D_p^{f(\cdot)}(f, S_p^{f(\cdot)}(\varepsilon)) = 1\right]$$

is negligible.

Proof. Let S be the PPT simulator that is guaranteed by Definition 6.2. We define S_p as the PPT simulator that runs $p(k)$ independent copies of S. Finally, S_p outputs $(O'_1, \ldots, O'_{p(k)})$, where O'_i denotes the output of the i-th simulation of S. A simple hybrid argument (that uses the assumption that $\mathcal{O}$ is an indifferentiable obfuscator for $\mathcal{F}$) shows

$$\mathsf{Adv}^{\mathsf{ind\text{-}obf\text{-}mult}}_{\mathcal{F},\mathcal{O},S_p,D_p,p}(k) \leq p(k) \cdot \mathsf{Adv}^{\mathsf{ind\text{-}obf}}_{\mathcal{F},\mathcal{O},S,D}(k)$$

for any given PPT D. □

Hence, we can use several obfuscations of the *same* function concurrently. However, note that the function is still chosen from the uniform distribution by the subsystem.

6.3 Basic Properties of Our Indifferentiable Definition

Technically, the only difference between Definition 6.2 and Definition 4.2 (our strong virtual black-box definition) is the following: in Definition 4.2, D always gets oracle access to f, whereas in Definition 6.2, D gets oracle access to $\mathcal{O}(f)$ in the real setting, and oracle access to f in the ideal setting. But when we assume perfect functionality of the obfuscation, we have that $\mathcal{O}(f)$ evaluates f everywhere, so that oracle access to $\mathcal{O}(f)$ and oracle access to f are interchangeable. We get:

[14] [27, Theorem 1] only proves composability for a constant number of subsystems (in our case obfuscations). However, it seems that techniques from Canetti [9] can be used to prove composability for any polynomial number of subsystems, given that we use the stronger order of quantifiers (see the comment in Section 6.1).

Theorem 6.4 (Definition 4.2 $\Leftrightarrow$ Definition 6.2 for obfuscations with perfect functionality)**.** *Let $\mathcal{F} = (\mathcal{F}_k)_{k\in\mathbb{N}}$ be a family of functions, and let $\mathcal{O}$ be an obfuscator for $\mathcal{F}$ that achieves perfect functionality (i.e., $(\mathcal{O}(f))(x) = f(x)$ for all k, $f \in \mathcal{F}_k$, and x). Then $\mathcal{O}$ satisfies Definition 4.2 if and only if $\mathcal{O}$ satisfies Definition 6.2.*

On the other hand, Definition 6.2 already implies a certain form of functionality: if no D can distinguish the real from the ideal setting, then no efficient algorithm can distinguish between oracle access to $\mathcal{O}(f)$ and oracle access to f. This gives rise to the following functionality requirement for obfuscations:

Definition 6.5 (Computational functionality)**.** *Let $\mathcal{F} = (\mathcal{F}_k)_{k\in\mathbb{N}}$ a family of functions. Then $\mathcal{O}$ achieves* computational functionality *for $\mathcal{F}$ iff the following holds: for every PPT distinguisher D, the function $\mathsf{Adv}^{\mathsf{comp\text{-}func}}_{\mathcal{F},\mathcal{O},D}(k)$ is negligible in k. Here,*

$$\mathsf{Adv}^{\mathsf{comp\text{-}func}}_{\mathcal{F},\mathcal{O},D}(k) := \mathsf{Pr}\left[D^{O(\cdot)}(f,O) = 1\right] - \mathsf{Pr}\left[D^{f(\cdot)}(f,O) = 1\right]$$

where the probability is taken over $f \overset{\$}{\leftarrow} \mathcal{F}_k, O \leftarrow \mathcal{O}(f)$

We obtain the following connection between our definitions:

Theorem 6.6 (Indifferentiability is equivalent to computational functionality plus strong virtual black-box)**.** *Let $\mathcal{F} = (\mathcal{F}_k)_{k\in\mathbb{N}}$ a family of functions, and let $\mathcal{O}$ be an obfuscator for $\mathcal{F}$. Then $\mathcal{O}$ satisfies indifferentiable obfuscation (Definition 6.2) if and only if $\mathcal{O}$ satisfies the strong virtual black-box property (Definition 4.2) and computational functionality (Definition 6.5).*

Proof. The definitions can be summarized as:

$$\text{Definition 6.5:} \quad \forall \text{ PPT } D : \qquad D^{O(\cdot)}(f,O) \overset{c}{\approx} D^{f(\cdot)}(f,O) \tag{9}$$

$$\text{Definition 4.2:} \quad \exists \text{ PPT } S \ \ \forall \text{ PPT } D : \qquad D(f,O) \overset{c}{\approx} D(f,S^{f(\cdot)}(\varepsilon)) \tag{10}$$

$$\text{Definition 6.2:} \quad \exists \text{ PPT } S \ \ \forall \text{ PPT } D : \qquad D^{O(\cdot)}(f,O) \overset{c}{\approx} D^{f(\cdot)}(f,S^{f(\cdot)}(\varepsilon)), \tag{11}$$

where we write $X \overset{c}{\approx} Y$ for $\mathsf{Pr}[X = 1] \overset{c}{\approx} \mathsf{Pr}[Y = 1]$, and we silently assume $f \overset{\$}{\leftarrow} \mathcal{F}_k$ and $O \leftarrow \mathcal{O}(f)$ in all probabilities. We have to show that (11) is equivalent to the combination of (9) and (10).

First assume (11). Fix the S from (11) and assume an arbitrary D as in (9). Define D_1 such that $D_1^{h(\cdot)}(f,O) := D^{f(\cdot)}(f,O)$ (i.e., D_1 simulates D but answers oracle queries using its first argument f). Then using (11) twice (once for D and once for D_1) yields

$$D^{O(\cdot)}(f,O) \overset{(11)}{\overset{c}{\approx}} D^{f(\cdot)}(f,S^{f(\cdot)}(\varepsilon)) = D_1^{f(\cdot)}(f,S^{f(\cdot)}(\varepsilon)) \overset{(11)}{\overset{c}{\approx}} D_1^{O(\cdot)}(f,O) = D^{f(\cdot)}(f,O),$$

which shows (9). Now consider a D as in (10). Note that D does not have any oracle access, but we can interpret D as a distinguisher in the sense of (11) that does not use its oracle. We get:

$$D(f,O) = D^{O(\cdot)}(f,O) \overset{(11)}{\overset{c}{\approx}} D^{f(\cdot)}(f,S^{f(\cdot)}(\varepsilon)) = D(f,S^{f(\cdot)}(\varepsilon)),$$

which shows (10).

Conversely, assume (10) and (9). Fix the S from (10) and assume an arbitrary D as in (11). Define D_2 such that $D_2(f,O) := D^{f(\cdot)}(f,O)$, similar to D_1 above. We get

$$D^{O(\cdot)}(f,O) \overset{(9)}{\overset{c}{\approx}} D^{f(\cdot)}(f,O) = D_2(f,O) \overset{(10)}{\overset{c}{\approx}} D_2(f,S^{f(\cdot)}(\varepsilon)) = D^{f(\cdot)}(f,S^{f(\cdot)}(\varepsilon)),$$

which shows (11) as desired. □

We note that the functionality requirement from Definition 6.5 is significantly weaker than perfect functionality or the approximate functionality by Hohenberger et al. [22]. For completeness we include a discussion involving the latter in Appendix A.

7 Conclusion

We have presented a simulation-based definition that, on the one hand, allows for obfuscating point functions, yet at the same time is strong enough for converting private-key cryptography into public-key cryptography.

We would like to stress again that we do *not* rule out unobfuscatability results. In fact, we have shown certain scenarios in which obfuscation is not possible. On the other hand, our positive results (in particular the simplicity of our point function obfuscation) leave hope that obfuscations in interesting cryptographic scenarios are possible. We have given a toy example for the case of private-key encryption.

As it turns out, our relaxed simulation-based definition does *not* behave well under composition. Hence, we have given another, stronger definition that has a built-in composability property. We have shown that this definition naturally splits up into a functionality (correctness) and a virtual black-box (secrecy) requirement. Even though our composable definition does not allow for obfuscating point functions in the standard model, it is an interesting question which positive results are possible here, in the random oracle model for example.

Acknowledgements

We are indebted to the Crypto 2006, the TCC 2007, and the Journal of Cryptology referees who gave very valuable comments that helped to substantially improve the paper. Specifically, the construction from Section 5.2 was suggested by one referee, and one TCC referee had very constructive comments concerning the presentation of our results. Much of the discussion in the paper is inspired by constructive comments or questions from the reports. We also thank Alex Dent for motivating discussions and Salil Vadhan for pointing out the unreasonableness of verify-only secure MACs and signatures.

References

[1] Ben Adida and Douglas Wikström. How to shuffle in public. In Salil P. Vadhan, editor, *TCC 2007*, volume 4392 of *LNCS*, pages 555–574, Amsterdam, The Netherlands, February 21–24, 2007. Springer-Verlag, Berlin, Germany.

[2] Michael Backes, Birgit Pfitzmann, and Michael Waidner. A composable cryptographic library with nested operations. In Sushil Jajodia, Vijayalakshmi Atluri, and Trent Jaeger, editors, *ACM CCS 03*, pages 220–230, Washington D.C., USA, October 27–30, 2003. ACM Press.

[3] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 1–18, Santa Barbara, CA, USA, August 19–23, 2001. Springer-Verlag, Berlin, Germany.

[4] Donald Beaver. Foundations of secure interactive computing. In Joan Feigenbaum, editor, *CRYPTO'91*, volume 576 of *LNCS*, pages 377–391, Santa Barbara, CA, USA, August 11–15, 1992. Springer-Verlag, Berlin, Germany.

[5] Mihir Bellare and Chanathip Namprempre. Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. In Tatsuaki Okamoto, editor, *ASIACRYPT 2000*, volume 1976 of *LNCS*, pages 531–545, Kyoto, Japan, December 3–7, 2000. Springer-Verlag, Berlin, Germany.

[6] Mihir Bellare, Anand Desai, Eric Jokipii, and Phillip Rogaway. A concrete security treatment of symmetric encryption. In *38th FOCS*, pages 394–403, Miami Beach, Florida, October 19–22, 1997. IEEE Computer Society Press.

[7] Mihir Bellare, Alexandra Boldyreva, and Adam O'Neill. Deterministic and efficiently searchable encryption. In Alfred Menezes, editor, *CRYPTO 2007*, volume 4622 of *LNCS*, pages 535–552, Santa Barbara, CA, USA, August 19–23, 2007. Springer-Verlag, Berlin, Germany.

[8] Ran Canetti. Towards realizing random oracles: Hash functions that hide all partial information. In Burton S. Kaliski Jr., editor, *CRYPTO'97*, volume 1294 of *LNCS*, pages 455–469, Santa Barbara, CA, USA, August 17–21, 1997. Springer-Verlag, Berlin, Germany.

[9] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd FOCS*, pages 136–145, Las Vegas, Nevada, USA, October 14–17, 2001. IEEE Computer Society Press.

[10] Ran Canetti, Daniele Micciancio, and Omer Reingold. Perfectly one-way probabilistic hash functions (preliminary version). In *30th ACM STOC*, pages 131–140, Dallas, Texas, USA, May 23–26, 1998. ACM Press.

[11] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.

[12] Yevgeniy Dodis and Adam Smith. Correcting errors without leaking partial information. In Harold N. Gabow and Ronald Fagin, editors, *37th ACM STOC*, pages 654–663, Baltimore, Maryland, USA, May 22–24, 2005. ACM Press.

[13] Oded Goldreich and Rafail Ostrovsky. Software protection and simulation of oblivious rams. *Journal of the ACM*, 43(3):431–473, 1996.

[14] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *Journal of the ACM*, 33:792–807, 1986.

[15] Oded Goldreich, Silvio Micali, and Avi Wigderson. Proofs that yield nothing but their validity or all languages in NP have zero-knowledge proof systems. *Journal of the ACM*, 38(3):691–729, 1991.

[16] Shafi Goldwasser and Yael Tauman Kalai. On the impossibility of obfuscation with auxiliary input. In *46th FOCS*, pages 553–562, Pittsburgh, PA, USA, October 23–25, 2005. IEEE Computer Society Press.

[17] Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270–299, 1984.

[18] Shafi Goldwasser and Guy N. Rothblum. On best-possible obfuscation. In Salil P. Vadhan, editor, *TCC 2007*, volume 4392 of *LNCS*, pages 194–213, Amsterdam, The Netherlands, February 21–24, 2007. Springer-Verlag, Berlin, Germany.

[19] Satoshi Hada. Zero-knowledge and code obfuscation. In Tatsuaki Okamoto, editor, *ASIACRYPT 2000*, volume 1976 of *LNCS*, pages 443–457, Kyoto, Japan, December 3–7, 2000. Springer-Verlag, Berlin, Germany.

[20] Dennis Hofheinz and Dominique Unruh. Simulatable security and polynomially bounded concurrent composability. In *2006 IEEE Symposium on Security and Privacy*, pages 169–183, Berkeley, California, USA, May 21–24, 2006. IEEE Computer Society Press.

[21] Dennis Hofheinz, John Malone-Lee, and Martijn Stam. Obfuscation for cryptographic purposes. In Salil P. Vadhan, editor, *TCC 2007*, volume 4392 of *LNCS*, pages 214–232, Amsterdam, The Netherlands, February 21–24, 2007. Springer-Verlag, Berlin, Germany.

[22] Susan Hohenberger, Guy N. Rothblum, Abhi Shelat, and Vinod Vaikuntanathan. Securely obfuscating re-encryption. In Salil P. Vadhan, editor, *TCC 2007*, volume 4392 of *LNCS*, pages 233–252, Amsterdam, The Netherlands, February 21–24, 2007. Springer-Verlag, Berlin, Germany.

[23] Rex Jaeschke. Encrypting C source for distribution. *Journal of C Language Translation*, 2(1), 1990.

[24] Jonathan Katz and Moti Yung. Complete characterization of security notions for probabilistic private-key encryption. In *32nd ACM STOC*, pages 245–254, Portland, Oregon, USA, May 21–23, 2000. ACM Press.

[25] Cullen Linn and Saumya K. Debray. Obfuscation of executable code to improve resistance to static disassembly. In Sushil Jajodia, Vijayalakshmi Atluri, and Trent Jaeger, editors, *ACM CCS 03*, pages 290–299, Washington D.C., USA, October 27–30, 2003. ACM Press.

[26] Ben Lynn, Manoj Prabhakaran, and Amit Sahai. Positive results and techniques for obfuscation. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 20–39, Interlaken, Switzerland, May 2–6, 2004. Springer-Verlag, Berlin, Germany.

[27] Ueli M. Maurer, Renato Renner, and Clemens Holenstein. Indifferentiability, impossibility results on reductions, and applications to the random oracle methodology. In Moni Naor, editor, *TCC 2004*, volume 2951 of *LNCS*, pages 21–39, Cambridge, MA, USA, February 19–21, 2004. Springer-Verlag, Berlin, Germany.

[28] Silvio Micali and Phillip Rogaway. Secure computation (abstract). In Joan Feigenbaum, editor, *CRYPTO'91*, volume 576 of *LNCS*, pages 392–404, Santa Barbara, CA, USA, August 11–15, 1992. Springer-Verlag, Berlin, Germany.

[29] Moni Naor and Moti Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *22nd ACM STOC*, Baltimore, Maryland, USA, May 14–16, 1990. ACM Press.

[30] Arvin Narayanan and Vitaly Shmatikov. Stronger security of authenticated key exchange. Cryptology ePrint Archive, Report 2006/182, 2006. `http://eprint.iacr.org/`.

[31] Rafail Ostrovsky and William E. Skeith III. Private searching on streaming data. In Victor Shoup, editor, *CRYPTO 2005*, volume 3621 of *LNCS*, pages 223–240, Santa Barbara, CA, USA, August 14–18, 2005. Springer-Verlag, Berlin, Germany.

[32] Charles Rackoff and Daniel R. Simon. Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. In Joan Feigenbaum, editor, *CRYPTO'91*, volume 576 of *LNCS*, pages 433–444, Santa Barbara, CA, USA, August 11–15, 1992. Springer-Verlag, Berlin, Germany.

[33] John Rompel. One-way functions are necessary and sufficient for secure signatures. In *22nd ACM STOC*, pages 387–394, Baltimore, Maryland, USA, May 14–16, 1990. ACM Press.

[34] Hoeteck Wee. On obfuscating point functions. In Harold N. Gabow and Ronald Fagin, editors, *37th ACM STOC*, pages 523–532, Baltimore, Maryland, USA, May 22–24, 2005. ACM Press.

A A Comparison of Approximate Functionality Requirements

In the main body of our paper we have concentrated on the security of obfuscation by presenting and analyzing several definitions for behaving like a virtual black box. For simplicity, we assumed the obfuscation to have perfect functionality, that is for any function $f \in \mathcal{F}$ and any input x, we require, with probability 1, that $(\mathcal{O}(f))(x) = f(x)$ (when f is probabilistic, the equality refers to the output distribution).

We have already given a very relaxed definition of computational approximate functionality (Definition 6.5). A more natural analogue of the "approximate functionality" requirement from Definition 3.1 for the case of function *distributions* would be the following. For a random function f and its obfuscation O, for all inputs x the distributions $f(x)$ and $O(x)$ are close. If we only allow finite domains[15] for f and O this can be formalized as follows.

Definition A.1 (Approximate functionality)**.** *An obfuscator $\mathcal{O}$ satisfies the* approximate functionality requirement *for a family of functions $\mathcal{F} = (\mathcal{F}_k)_{k\in\mathbb{N}}$ iff there exists a negligible function ν such that for all k, we have:*

$$\epsilon^{\mathsf{fun}}_{\mathcal{O},\mathcal{F}}(k) := \mathop{\mathsf{EV}}_{f\leftarrow\mathcal{F}_k, O\leftarrow\mathcal{O}(f)} \left[\max_x \{\Delta(f(x)\,;\,O(x))\}\right] \le \nu(k).$$

(Henceforth we will use ϵ_{fun} as shorthand for $\epsilon^{\mathsf{fun}}_{\mathcal{O},\mathcal{F}}(k)$.)

For deterministic functions, the requirement reduces to the statement that, with overwhelming probability over the choice of f and the obfuscator $\mathcal{O}$, f and $\mathcal{O}(f)$ should be the same functions. This is similar to the approximate functionality of the worst-case definition [3, Definition 4.3], with the caveat that we take our probability over f as well.

The relation with the composable functionality definition. With some work one can show that Definition A.1 implies Definition 6.5. what is more surprising perhaps is that Definition A.1 is *equivalent* to a statistical version of Definition 6.5 where D is computationally unrestricted, but is restricted to only polynomially many queries. For completeness, we first explicitly state the relevant definition, before stating the theorem detailing equivalence and its proof.

[15] The assumption that $f \in \mathcal{F}_k$ can be computed in PPT *in the security parameter* already implies that the domain of $f \in \mathcal{F}_k$ is polynomially bounded (in k) and hence finite.

Definition A.2 (Statistical functionality)**.** *Let $\mathcal{F} = (\mathcal{F}_k)_{k\in\mathbb{N}}$ be a family of functions. Then $\mathcal{O}$ achieves* statistical functionality for $\mathcal{F}$ *iff the following holds: for every (computationally unbounded) distinguisher D making only polynomially many queries to its oracle, the function*

$$\mathsf{Pr}\left[f \xleftarrow{\$} \mathcal{F}_k, O \leftarrow \mathcal{O}(f) \,:\, D^{O(\cdot)}(f,O) = 1\right] - \mathsf{Pr}\left[f \xleftarrow{\$} \mathcal{F}_k, O \leftarrow \mathcal{O}(f) \,:\, D^{f(\cdot)}(f,O)) = 1\right].$$

is negligible in k.

Theorem A.3. *Let $\mathcal{F} = (\mathcal{F}_k)_{k\in\mathbb{N}}$ be a family of functions. Then $\mathcal{O}$ satisfies Definition A.2 if and only if it satisfies Definition A.1.*

Proof (sketch). We show that for all distinguishers according to Definition A.2, their advantage is upper bounded by a polynomial (in k) multiple of ϵ_{fun}. Hence if ϵ_{fun} is negligible according to Definition A.1, so should any distinguisher's advantage according to Definition A.2. Furthermore we exhibit a distinguisher that has advantage ϵ_{fun}, so if all distinguishers have a negligible advantage (Definition A.2) it follows that ϵ_{fun} is negligible, fulfilling Definition A.1.

Recall the definition of ϵ_{fun} and consider a computationally unbounded distinguisher with polynomially many queries. For any single query x, it is well known that the advantage of a distinguisher is at most $\Delta(f(x)\,;\,O(x))$. The best the distinguisher can do given f and $O = \mathcal{O}(f)$ is to determine the x that maximizes the statistical distance $\Delta(f(x)\,;\,O(x))$ (as in ϵ_{fun}) and query the function on that point. Each new query will add at most ϵ_{fun} to the distinguisher's advantage, so with polynomially many queries, the total advantage will still be a polynomial multiple of the maximum statistical distance. Hence we can upper bound the advantage of D in terms of properties of f and O, where we still need to average out over $f \xleftarrow{\$} \mathcal{F}_k$ and $O \leftarrow \mathcal{O}(f)$.

On the other hand, there exists a distinguisher that achieves the advantage governed by the statistical difference. The key observation is that determining the value x for which the maximum $\max_x \Delta(f(x)\,;\,O(x))$ is achieved can be done by a computationally unbounded distinguisher given f and O without using its oracle (in fact, it can even be done in polynomial space). The main point here is that f and O can be computed in probabilistic polynomial time, so in particular there is polynomial bound on the amount of random coins each uses. Consequently for each x we have that $\Delta(f(x)\,;\,O(x))$ can be computed (in finite time). The maximum can be computed exploiting that f (and hence O) have a finite domain. Now consider the distinguisher that, on input f and O, first determines the x that maximizes $\Delta(f(x)\,;\,O(x))$ and queries x to its oracle. When it gets a response s that would have been more likely to have originated from f, or $\mathsf{Pr}\left[f(x) = s\right] > \mathsf{Pr}\left[O(x) = s\right]$ it outputs 1, when $O(x)$ was more likely to have caused s it outputs 0; when both were equally likely it flips a coin. In this case the advantage is equal to ϵ_{fun}. □

The connection with Hohenberger et al.'s functionality requirement. A relaxed functionality requirement for probabilistic functions was given by Hohenberger et al. [22]. Adapted to our notation, it is reproduced below in Definition A.4.

Definition A.4 ([22]-approximate functionality)**.** *An obfuscator $\mathcal{O}$ satisfies the* [22]-approximate functionality requirement *for a family of functions $\mathcal{F} = (\mathcal{F}_k)_{k\in\mathbb{N}}$ iff there exists a negligible function ν such that for all k and all $f \in \mathcal{F}_k$*

$$\mathsf{Pr}\left[O \leftarrow \mathcal{O}(f) \,:\, \exists_x\, \Delta(f(x)\,;\,O(x)) \geq \nu(k)\right] \leq \nu(k)\ .$$

On quick inspection, this functionality requirement looks quite different from previous ones. Let us concentrate on a comparison with Definition A.1. An obvious difference is that Hohenberger et al. quantify universally over $f \in \mathcal{F}_k$, whereas we randomize over $f \xleftarrow{\$} \mathcal{F}_k$. It turns out that if we would account for this (either by using a universal quantifier over f in Definition A.1 or by randomizing over f in Definition A.4), the two seemingly different looking definitions are in fact equivalent, as demonstrated by the following lemma.

Lemma A.5. *Let $\mathcal{O}$ be an obfuscator for a class of functions $\mathcal{F}$. Then the following two statements are equivalent:*

1. *There exists a negligible function ν such that for all k and all $f \in \mathcal{F}_k$,*

$$\mathop{\mathsf{EV}}_{O \leftarrow \mathcal{O}(f)} \left[\max_x \Delta(f(x)\,;\,O(x)) \right] \leq \nu(k).$$

2. *There exists a negligible function ν such that for all k and all $f \in \mathcal{F}_k$*

$$\mathsf{Pr}\left[O \leftarrow \mathcal{O}(f)\ :\ \exists_x\, \Delta(f(x)\,;\,O(x)) \geq \nu(k)\right] \leq \nu(k)\ .$$

Proof. Write $\sigma(x) = \Delta(f(x)\,;\,O(x))$ (where f and O should be clear from the context).

We first show that the second statement implies the first. We notice that the event $\exists_x \sigma(x) \geq \nu(k)$ occurs iff $\max_x \sigma(x) \geq \nu(k)$. Thus

$$\mathsf{Pr}\left[O \leftarrow \mathcal{O}(f)\ :\ \exists_x \sigma(x) \geq \nu(k)\right] = \mathsf{Pr}\left[O \leftarrow \mathcal{O}(f)\ :\ \max_x \sigma(x) \geq \nu(k)\right]\ .$$

We can now split the expectancy in two and use that any statistical distance, so in particular $\sigma(x)$, is always upper bounded by 1:

$$\mathop{\mathsf{EV}}_{O \leftarrow \mathcal{O}(f)} [\max_x \sigma(x)] \leq \mathsf{Pr}\left[\max_x \sigma(x) \geq \nu(k)\right] \cdot 1 + \mathsf{Pr}\left[\max_x \sigma(x) < \nu(k)\right] \nu(k) \leq 2\nu(k)\ ,$$

where the probabilities are over the choice of the obfuscation $O \leftarrow \mathcal{O}(f)$. Since $2\nu(k)$ is negligible if $\nu(k)$ is, this concludes the first implication.

Conversely, $\mathsf{Pr}\left[O \leftarrow \mathcal{O}(f)\ :\ \max_x \sigma(x) \geq \nu(k)\right] > \nu(k)$ implies that

$$\mathop{\mathsf{EV}}_{O \leftarrow \mathcal{O}(f)} [\max_x \sigma(x)] \geq \mathsf{Pr}\left[O \leftarrow \mathcal{O}(f)\ :\ \max_x \sigma(x) \geq \nu(k)\right] \nu(k) > (\nu(k))^2\ .$$

Therefore, if $\mathsf{EV}_{O \leftarrow \mathcal{O}(f)}[\max_x \sigma(x)] \leq \nu(k)$ for some negligible function ν, it follows that for the negligible function $\nu'(k) = \sqrt{\nu(k)}$ it holds that $\mathsf{Pr}\left[O \leftarrow \mathcal{O}(f)\ :\ \max_x \sigma(x) \geq \nu'(k)\right] \leq \nu'(k)$. □