Random Oracle Uninstantiability from Indistinguishability Obfuscation

Christina Brzuska[1] Pooya Farshim[2] Arno Mittelbach[3]

[1]Microsoft Research Cambridge, UK
[2]Queen's University Belfast, Northern Ireland, UK
[3]Darmstadt University of Technology, Germany
christina.brzuska@gmail.com pooya.farshim@gmail.com arno.mittelbach@cased.de

Abstract. Assuming the existence of an indistinguishability obfuscator (iO), we show that a number of prominent constructions in the random-oracle model are *uninstantiable* in the standard model. We first show that the Encrypt-with-Hash (EwH) transform of Bellare, Boldyreva, and O'Neill (CRYPTO 2007) for converting randomized public-key encryption (PKE) to deterministic PKE is not instantiable in the standard model. The techniques that we use to establish this result are flexible and lend themselves easily to other transformations. These include the classical Fujisaki–Okamoto transform (CRYPTO 1998) for converting CPA to CCA security, a transformation akin to that by Bellare and Keelveedhi (CRYPTO 2011) for obtaining key-dependent security, as well as the convergent encryption transform for obtaining messaged-locked encryption by Bellare, Keelveedhi, and Ristenpart (EUROCRYPT 2013). Our techniques build on the recent work of Brzuska, Farshim, and Mittelbach (CRYPTO 2014) and rely on the existence of iO for Turing machines or circuits to derive two flavors of uninstantiability. Our results call for a re-assessment of scheme design in the random-oracle model and point out the need for new transforms which do not suffer from our attacks.

Keywords. Random oracle, uninstantiability, indistinguishability obfuscation, deterministic PKE, hedged PKE, message-locked encryption, Fujisaki–Okamoto, KDM security, UCE.

Contents

1 Introduction

1.1 Background

The random-oracle model (ROM) is an idealized model of computation where all parties, honest or otherwise, have oracle access to a random function. Random oracles model ideal hash functions and were first formalized in the work of Bellare and Rogaway [BR93]. They have since found a plethora of applications, enabling the security proofs of a wide range of practical cryptosystems which include, amongst others, digital signature schemes, CCA-secure encryption, key-exchange protocols, identity-based encryption, cryptosystems that are resilient to related-key and key-dependent-message attacks, as well as more advanced security goals such as deterministic encryption of high-entropy messages, de-duplication schemes, and point-function obfuscation.

In this paper we revisit the random-oracle methodology, whereby one first designs a scheme in the ROM and then instantiates the oracle via a concrete hash function. We show that a number of prominent ROM cryptosystems cannot be securely instantiated in the standard model.

1.2 Uninstantiability

The power and practicality of random oracles drew attention to their standard-model instantiations and Canetti, Goldreich, and Halevi (CGH) [CGH98] demonstrated a general negative result by showing that one can construct digital signature and encryption schemes which are secure in the random-oracle model but become insecure as soon as the oracle is instantiated with *any* concrete hash function. Roughly speaking, these *uninstantiable* schemes rely on the existence of a compact description for the hash functions (and lack of one for a truly random function). The idea is to take a secure ROM scheme and tweak it so that it behaves in an "obviously insecure" way (e.g., it returns the signing key or the message) when run on messages that match the code of the hash function used in instantiation (and behaves securely otherwise).

A number of follow-up works have further studied uninstantiability problems associated with random oracles in the standard model. CGH, in a follow up work, extend their result to signature schemes which only support short messages [CGH03]. Bellare, Boldyreva and Palacio [BBP04] show that no instantiation of the hashed ElGamal key-encapsulation mechanism composes well with symmetric schemes, even though it enjoys this property in the ROM. Goldwasser and Tauman-Kalai [GK03] study the Fiat–Shamir heuristic and establish uninstantiability results for it. Nielsen [Nie02] gives an uninstantiable ROM *task*, namely that of non-interactive, non-committing encryption. CGH-type uninstantiability has been adapted to other idealized models of computations such as the ideal-cipher model [Bla06] and the generic-group model [Den02].

A number of recent works have looked into ROM (un)instantiability in light of the recently proposed candidate for indistinguishability obfuscation (iO) [GGH$^+$13]. Roughly speaking, a secure indistinguishability obfuscator guarantees that the obfuscations of two functionally equivalent algorithms (which may be modeled as circuits or Turing machines) are computationally indistinguishable. On the positive side, Hohenberger, Sahai, and Waters [HSW14] show how to instantiate the hash function in full-domain hash signatures using iO. Bellare, Stepanovs and Tessaro [BST14] show the first standard-model construction for polynomially many hardcore bits for any one-way function. Recently, Brzuska and Mittelbach [BM14c] have shown how to use iO to instantiate assumptions from the UCE (Universal Computational Extractor) framework of Bellare, Hoang and Keelveedhi [BHK13a]. Roughly speaking, UCEs model various strong extractor properties enjoyed by random oracles.

On the negative side, Brzuska, Farshim, and Mittelbach [BFM14] show that if iO exists then several security notions in the UCE framework are uninstantiable in the standard model. Brzuska and Mittelbach [BM14b] show that assuming iO, multi-bit output point-function obfuscation secure in the presence of auxiliary information cannot be realized. Both results can be interpreted as conditional

uninstantiability results as ROM constructions for both UCEs [BHK13a, Mit14] and strong multi-output bit point obfuscation [LPS04] exist. Bitansky et al. [BCPR14] show that indistinguishability obfuscation rules out the existence of certain types of extractable one-way function families which can be easily constructed in the random-oracle model [CD08]. We note that indistinguishability obfuscation is also mutually exclusive with a number of other assumptions [BCC$^+$14, MO14, KRW13].

1.3 Our results

Our work continues the study of uninstantiability of random oracles and shows that a number of well-known and widely employed ROM constructions are provably uninstantiable if indistinguishability obfuscators exist. More specifically, we are interested in ROM transformations T^{RO} which take as input any scheme S that satisfies a mild form of security, and convert S to a scheme $\mathsf{T}^{\mathsf{RO}}[\mathsf{S}]$ which meets a stronger level of security. We ask whether or not T^{RO} can be instantiated with in standard mode hash function. That is, whether or not there exists an efficient hash function H such that $\mathsf{T}^{\mathsf{H}}[\mathsf{S}]$ is strongly secure for *any* mildly secure S. A negative result in this direction would therefore take the form: there is a mildly secure scheme S^* such that no matter which hash function H is picked, scheme $\mathsf{T}[\mathsf{S}^*, \mathsf{H}] := \mathsf{T}^{\mathsf{H}}[\mathsf{S}^*]$ is provably insecure.

Our results come in two flavors depending on the class of programs that the indistinguishability obfuscator supports. Assuming iO for the class of polynomial-sized circuits, we show that for any hash function of size at most p, an a priori fixed parameter, there is a ROM cryptosystem which is uninstantiable with respect to (keyed) hash functions of description size at most p. This means that there exists a scheme S_{p} such that for any hash function H that has a description size of at most p the scheme $\mathsf{T}[\mathsf{S}_{\mathrm{p}}, \mathsf{H}]$ will be insecure. This yields an impossibility result for any fixed, finite set of hash functions. On the other hand, this result does not rule out instantiating the oracle with hash functions which are "more complex" than the ROM scheme that they are instantiating. By assuming the existence of iO for the class of polynomial-time *Turing machines*, we can strengthen this result to one which rules out instantiations with respect to *any*, possibly scheme-dependent, hash function.

Overview of BFM. We build on the techniques by Brzuska, Farshim, and Mittelbach (BFM) [BFM14] to construct our uninstantiable schemes and briefly recall their result here. BFM utilize the power of indistinguishability obfuscation to show that a recent notion of security for hash functions known as UCE1 (later renamed as UCE[$\mathcal{S}^{\mathrm{cup}}$]) is uninstantiable in the standard model.[1] To this end, BFM attack UCE1 by leaking an indistinguishability obfuscation of the circuit

$$\mathsf{C}[x, y](\cdot) := (\mathsf{H}(\cdot, x) = y) \ ,$$

where x is a random domain point and y is the corresponding hash value received from the oracle. Both these values are hardcoded into C. BFM need to argue that indistinguishability obfuscation of this circuit hides x when y is truly random. They prove this by showing that, under appropriate restriction on the size of the range and the key space, the above circuit implements the constant all-zero circuit with overwhelming probability. They then employ the security of the obfuscator to conclude as the zero circuit is independent of x. The size restriction that they need requires the number of hash keys to be much smaller than the size of the co-domain, which means y, with overwhelming probability, is outside the set of values that $\mathsf{H}(\cdot, x)$ assumes.

[1] In UCE1 security a two-stage adversary needs to distinguish a hash function from a random oracle. The first-stage adversary is given oracle access to either the hash function under an unknown key or a random oracle. It does not get to see the hash key but may leak a message once to the second-stage adversary which additionally does get to see the hash key. The second-stage adversary can no longer call the oracle. UCE1 security requires that the leaked message should be such that it does not reveal any of the oracle queries when the oracle is random.

TECHNIQUES. We consider a *universal* variant of the BFM circuit which takes as input the description of a hash function (including its key if there is one) and returns the result of running the BFM circuit on the input hash function. It performs the latter in the standard way by using a universal machine UEval:

$$\mathsf{P}[x, y](\mathsf{H}_{\mathsf{hk}}) := (\mathsf{UEval}(\mathsf{H}_{\mathsf{hk}}, x) = y) \ .$$

In other words, we no longer consider a fixed hash function and varying keys, but instead (potentially) look at the set of *all* hash functions on a given range and domain. Similar ideas have been recently used by Brzuska and Mittelbach [BM14b] to study the feasibility of multi-bit output point function obfuscation in the presence of auxiliary inputs under the iO assumption. In adopting this approach, however, a number of technicalities need to be addressed, which we discuss next.

Our ultimate goal is to derive a strong result which rules out instantiations by any arbitrary hash function. This means that P above should accept inputs of arbitrary length. This, however, lies beyond the powers of the circuit model of computation. We address this problem in two (incomparable) ways. First, we weaken the target uninstantiability result and consider only bounded sized circuits. Put differently, assuming iO for circuits, we are able to rule out instantiations by a priori bounded size circuits. Second, in order to strengthen this result to full uninstantiability, we consider a stronger form of iO which supports the class of *Turing machines*. The crucial difference with the circuit class is that Turing machines can take inputs of *arbitrary* size. Note that the actual machine that we need to obfuscate is a universal Turing machine of size, say, λ which is able to accept arbitrarily large inputs. Our theorem statements will contain two parts to reflect this trade off between the strength of assumptions and the strength of the negative result than can be achieved.

A second problem arises from the fact that the number of possible hash functions might be greater than $2^{|y|}$ so that we cannot directly apply the counting argument used by BFM. We overcome this obstacle by composing both sides of the equality in P with a pseudorandom generator (PRG):

$$\mathsf{P}[x, y](\mathsf{H}_{\mathsf{hk}}) := (\mathsf{PRG}(\mathsf{UEval}(\mathsf{H}_{\mathsf{hk}}, x)) = \mathsf{PRG}(y)) \ .$$

This does not affect the success probability of the attack. Next, to argue that x remains hidden, we observe that the right hand side, i.e., $\mathsf{PRG}(y)$, can be changed to a truly random value by the security of the PRG. Note that in this step we do not rely on the security of iO as the two circuits might implement significantly different functionalities. Finally, we use the fact that a truly random value is outside the range of a PRG with sufficiently long stretch with overwhelming probability. As a result we conclude that the obfuscations of the above circuit are computationally indistinguishable from those of the all-zero circuit. We note that our usage of the PRG is somewhat similar to that by Sahai and Waters in their construction of a CCA-secure PKE scheme from iO [SW14] as well as the range-extension of Matsuda and Hanaoka [MH14] of a multi-bit point function to obtain shorter point values, the range-extension of a UCE1-secure hash function by Bellare et al. [BHK13c] and the 1-out-of-2 result for iO and multi-bit output point function obfuscation in the presence of auxiliary inputs by Brzuska and Mittelbach [BM14b].

PLAUSIBILITY OF ASSUMPTIONS. Garg et al. [GGH$^+$13] base indistinguishability obfuscation for all circuits in $\mathcal{NC}^1$ on intractability assumptions related to multi-linear maps, which they validate in generic models of computation. To achieve indistinguishability obfuscation for all polynomial-time circuits, they then apply a bootstrapping technique using fully homomorphic encryption with decryption in $\mathcal{NC}^1$. Recent results show how to improve the assumptions used in constructing indistinguishability obfuscators [PST14, BR14, BGK$^+$14, AGIS14a, GLSW14a], further supporting their plausibility.

Indistinguishability obfuscation for Turing machines has been constructed in the works of Boyle, Chung, and Pass [BCP14] and Ananth et al. [ABG$^+$13]. The authors study a stronger primitive called *extractability* or *differing-inputs* obfuscation (diO) which extends iO to circuits (and Turing machines) that are not necessarily functionally equivalent. The requirement is that any adversary that can break

the indistinguishability property can be converted to an extractor that can output a point on which the two circuits differ. Boyle et al. [BCP14], and independently Ananth et al. [ABG+13], show how to build iO for *Turing machines* assuming diO for circuits.

The plausibility of differing-inputs obfuscation has become somewhat controversial due to a recent result of Garg et al. [GGHW14] who show that the existence of a special-purpose obfuscator for a signature scheme implies that diO with arbitrary auxiliary input cannot exist. We currently do not know how to build this special-purpose obfuscator. However, as the special-purpose obfuscator appears to be a milder assumption than diO, one can consider its existence to be more likely. It is therefore important to seek alternative instantiations of iO for Turing machines from assumptions that are weaker than diO for circuits.

Deterministic PKEs. Our first result establishes the uninstantiability of the Encrypt-with-Hash (EwH) transform of Bellare, Boldyreva and O'Neill [BBO07], whereby one converts a randomized public-key encryption scheme into a deterministic public-key encryption (D-PKE) scheme for high-entropy messages by extracting the randomness needed for encryption by hashing the message and the public key. This simple transformation meets the strongest notion of security that has been proposed for deterministic encryption (that is, PRIV security) in the ROM if the underlying encryption scheme is IND-CPA secure. This notion of security has only been met in idealized models, and standard-model counterparts achieve a weaker level of security (e.g., for block sources [BFOR08, BFO08] or q-bounded adversaries [FOR12, BM14a]).

We ask if a hash function can be used to instantiate the random oracle within the EwH transform. Assuming indistinguishability obfuscation, we answer this question in the negative. Starting with an arbitrary scheme PKE we convert it to a new scheme PKE^* which includes an obfuscation of the following circuit as part of its ciphertexts.

$$\mathsf{P}[pk, m, r](\mathsf{H}_{\mathsf{hk}}) := \text{if } (\mathsf{PRG}(\mathsf{UEval}(\mathsf{H}_{\mathsf{hk}}, pk \| m)) = \mathsf{PRG}(r)) \text{ return } m \text{ else return } 0$$

This circuit performs the same check as that in the universal BFM circuit above, and if it passes, instead of returning a Boolean value, it returns the encrypted messages. This circuit can be used to attack the security of $\mathsf{EwH}^{\mathsf{H}}[\mathsf{PKE}^*]$, by running it on a description H_{hk} (with hardcoded key hk) that is used in the instantiation. A corollary of this result is that *any* security assumption which is strong enough to build D-PKEs via EwH cannot be instantiated in the standard model. In particular, under iO for Turning machines, any UCE assumption [BHK14] that suffices to instantiate EwH is uninstantiable assuming iO for Turing Maschines (and p-bounded uninstantiable assuming iO for circuits). We note that our results are incomparable to those of Wichs [Wic13] who shows an unprovability result for D-PKEs using arbitrary techniques from single-stage assumptions. In turn, assuming iO, our result shows uninstantiability of EwH regardless of the assumptions used. An extension of our result establishes that the Randomized-Encrypt-with-Hash [BBN+09] transform for building hedged PKEs is also uninstantiable.

The Fujisaki–Okamoto transform. We are able to generalize the above result to a wider class of (possibly randomized) structured transformations that we term *admissible*. These are transformations which use their underlying PKE scheme in a specific way and admit a recovery algorithm (we leave the details to the main body). Somewhat surprisingly, we show that the Fujisaki–Okamoto (FO) transform for converting CPA into CCA security falls under this class of transformations and thus suffers from similar problems to EwH. The FO transform, which dates back to the 1990s, is a simple and flexible technique to boost the security of various schemes and has been widely used in primitives such as identity-based encryption (IBE) [BF01], hierarchical and fuzzy IBEs [GS02, SW05], forward-secure encryption [CHK03], and certificateless and certificate-based encryption [ARP03, Gen03]. As before, our uninstantiability results for FO also come in two flavors depending on the strength of the underlying obfuscator. We remark that our results also show that one cannot instantiate the oracle used within

the asymmetric component of the FO transform, and in this sense complement those of Boldyreva and Fischlin [BF05] on partial instantiation of oracles in FO.

We give a new transform for building D-PKEs that does not fall into the class of admissible transformations. This transform is specifically designed to bypass our attacks above by encrypting two values independently across two invocations of the encryption algorithm so that information needed for the attack is shared out. Perhaps surprisingly, we show that the circuits used in the attacks above can be split into several circuits, and by feeding obfuscations of one circuit into obfuscation of another we can launch an attack on this transform. Consequently, our attacks apply to a potentially wider class of transforms whose characterization we leave as an interesting open problem.

Other transforms. The uninstantiability problems arising from the existence of indistinguishability obfuscators are not limited to D-PKEs and its generalizations. We revisit the work of Bellare and Keelveedhi (BK) [BK11] on authenticated and misuse-resistant encryption of key-dependent data and show that it too suffers from uninstantiability problems. Roughly speaking, BK give a transformation to convert authenticated encryption into one which resists key-dependent-message (KDM) attacks by hashing the key with a random nonce before use. More precisely, in this transform one encrypts M as $(R, \mathsf{enc}(\mathsf{H}(\mathsf{hk}, R\|K), M))$ for a random R. Our iO-based uninstantiability result describes an IND-CPA and INT-CTXT-secure authenticated encryption (AE) scheme whose transformation is not KDM secure.

Interestingly, BK assume a stronger notion than IND-CPA security in their work, namely, that ciphertexts are indistinguishable from random strings. BK do not consider this difference to be a major issue, as they consider their transform to work for any authenticated encryption scheme. Our result brings this stronger requirement to light, and shows that assuming that ciphertexts are pseudorandom might be a way to circumvent uninstantiability: the current state-of-the-art obfuscators produce programs that are structured and do not look random. It is an interesting problem to give positive feasibility results in contexts that exploit this subtlety in the strength of security definition. Moreover, it is unclear whether indistinguishability obfuscation can produce obfuscations of the all-zero circuit that look random.[2] If possible, then reverting to the stronger security definition of indistinguishability from random strings would not be of help to circumvent our uninstantiability result.

As another example we show that the convergent encryption transform originally proposed by Douceur et al. [DAB+02] and formalized by Bellare, Keelveedhi and Ristenpart (BKR) [BKR13] for building message-locked encryption is uninstantiable. Note that the same considerations as for BK apply and that also BKR used the stronger assumption of pseudorandom ciphertexts in their proofs.

Comparison with CGH. It is natural to ask if CGH-like techniques can be directly used so as to obtain uninstantiability results that do not rely on iO. With respect to *unkeyed* hash functions one can indeed construct anomalous PKE schemes which follow the CGH paradigm and give the desired uninstantiability result. For keyed hash functions, on the other hand, there seems to be an inherent limitation to CGH-like techniques. For instance, the security model for D-PKEs do *not* allow message distributions to depend on the hash key as this value is a component of the public key and the latter is denied to the first-stage adversary. Consequently there is no way to generate messages which contain the full description of the hash function used, *including its key*. It might appear that this issue can be resolved by noting that the encryption routine *does* have access to the hash key and a full description of the hash function can be formed at this point. However, the caveat is that such an uninstantiable scheme would not fall under the umbrella of schemes arising from the encrypt-with-hash transform. More precisely, although we can freely modify the base PKE to prove uninstantiability, the transformation is *fixed* and in particular it only allows black-box access to the hash function and denies encryption access

[2] Note that generally, obfuscations of circuits cannot look like random strings, because obfuscation maintains functionality and thus, the all-zero circuit and the all-one circuit are distinguishable. However, such a trivial attack does not apply here if we only require pseudorandomness for the all-zero circuit.

to the hash key. These observations also apply to the other transformations that we consider. For the case of the FO transformation, note that the message that is PKE-encrypted is chosen uniformly at random and thus cannot be set to the description of the hash function. Despite this, CGH-like techniques render Encrypt-with-Hash uninstantiable when stronger notions of security are considered [RSV13].

ROM PROTOCOL DESIGN. The shortcomings that we have identified in this work call for a re-assessment of the security of protocols whose security analyses have only been carried out in idealized models of computation such as ROM. We believe the structural soundness of such schemes should be further validated by studying if iO-based attacks similar to those given above can be launched against them. One way to rule out such attacks would be to prove security under assumptions, which although strong, are known to be instantiable in the standard model or at least are not known to be uninstantiable. Candidate examples include UCEs against statistically or strongly unpredictable sources [BFM14, BHK13b, BM14c] and indistinguishability obfuscation [GGH$^+$13, ABG$^+$13, BCP14].

One can also combine this approach with stronger assumptions on the base schemes such as pseudorandomness of ciphertexts. For instance, it would be interesting to find positive results that circumvent iO-based uninstantiability by merely exploiting the pseudorandomness of ciphertexts, even if this were done for somewhat artificial tasks. This, in turn, would substantially strengthen our confidence in modular RO-based design despite the broad uninstantiability results presented in this paper.

2 Preliminaries

NOTATION. We denote the security parameter by $\lambda \in \mathbb{N}$ and assume that it is implicitly given to all algorithms in the unary representation 1^λ. We denote the set of all bit strings of length ℓ by $\{0,1\}^\ell$, the set of all bit strings of finite length by $\{0,1\}^*$, the length of $x \in \{0,1\}^*$ by $|x|$, the concatenation of two strings $x_1, x_2 \in \{0,1\}^*$ by $x_1 \| x_2$, and the exclusive or of two strings $x_1, x_2 \in \{0,1\}^*$ of the same length by $x_1 \oplus x_2$. The i-th bit of a string x is indicated by $x[i]$. A vector of strings $\mathbf{x}$ is written in boldface, and $\mathbf{x}[i]$ denotes its i-th entry. The number of entries of $\mathbf{x}$ is denoted by $|\mathbf{x}|$. For a finite set X, we denote the cardinality of X by $|X|$ and the action of sampling x uniformly at random from X by $x \leftarrow\!\!\$\ X$. We say a function $\nu(\lambda)$ is negligible if $\nu(\lambda) \in \mathcal{O}(\lambda^{-\omega(1)})$. We denote the set of all negligible functions by negl. For a random variable X we denote the support of X by $[X]$.

TURING MACHINES AND CIRCUITS. Throughout this paper we will consider two models of computation: Turing machine and circuits. We denote the (worst-case) runtime of a Turing machine M on input x by $\mathsf{time}_{\mathsf{M}}(x)$ and the description size by $|\mathsf{M}|$. We denote the size (aka. runtime) of a circuit C by $|\mathsf{C}|$. Recall that a Turing machine can take inputs of arbitrary length whereas the input length to a circuit is fixed. A *universal* Turing machine UM is a machine that takes two inputs (M, x), interprets M as the description of a Turing machine and returns $\mathsf{M}(x)$. A *universal circuit* UC is defined analogously by taking the description of a circuit C. Note that UC only accepts inputs (C, x) of a specific length, whereas UM can take inputs of arbitrary length. In order to simplify we use the term *program* to refer to either a Turing machine or a circuit. Analogously, we may speak of a universal program UEval, which denotes either a universal Turing machine UM or a universal circuit evaluator UC, and evaluates a program P on some input x. When defining a program P we will often use the notation $P[z](\cdot)$ to denote that program has z hardcoded into it.

We assume all algorithms are randomized unless otherwise stated. We call an algorithm efficient or PPT if it runs in time polynomial in the (length of the) security parameter. The action of running an algorithm $\mathcal{A}$ on input x and random coins r is denoted by $y \leftarrow \mathcal{A}(x; r)$. If $\mathcal{A}$ is randomized and no randomness is specified, then we assume that $\mathcal{A}$ is run with freshly and uniformly chosen random coins and write this as $y \leftarrow\!\!\$\ \mathcal{A}(x)$. We often refer to algorithms, or tuples of algorithms, as schemes or adversaries.

INDISTINGUISHABILITY OBFUSCATION. In the following we define indistinguishability obfuscation for circuits and Turing machines. Roughly speaking, an indistinguishability obfuscator (iO) ensures that the obfuscations of any two functionally equivalent programs (that is, circuits or Turing machines) are computationally indistinguishable. Indistinguishability obfuscation was originally proposed by Barak et al. [BGI+01, BGI+12] as a potential weakening of virtual-black-box obfuscation for which wide infeasibility results are known. We here give a game-based definition of indistinguishability obfuscation in the style of [BST14] with extensions to also cover obfuscation for Turing machines [ABG+13]. We note that we only consider the weaker uniform setting for both sampler and distinguisher but consider samplers that only need to output equivalent circuits with overwhelming probability.

A PPT Turing machine iO is called an *indistinguishability obfuscator* for a program class $\mathcal{P} = \{\mathcal{P}_\lambda\}_{\lambda\in\mathbb{N}}$ if iO on input the security parameter 1^λ and (the description of) a program P outputs a program P' and furthermore the following conditions are satisfied:

- CORRECTNESS. For all $\lambda \in \mathbb{N}$, all $P \in \mathcal{P}_\lambda$, and all $P' \leftarrow\!\!\$\ \mathsf{iO}(1^\lambda, P)$, the programs P and P' are functionally equivalent.
- SUCCINCTNESS. There is a polynomial poly such that for all $\lambda \in \mathbb{N}$, all $P \in \mathcal{P}_\lambda$ and all $P' \leftarrow\!\!\$\ \mathsf{iO}(1^\lambda, P)$ we have that $|P'| \in \mathcal{O}(\mathsf{poly}(\lambda + |P|))$.
- INPUT-SPECIFIC RUNTIME. There is a polynomial poly such that for all $\lambda \in \mathbb{N}$, all $P \in \mathcal{P}_\lambda$ and all $P' \leftarrow\!\!\$\ \mathsf{iO}(1^\lambda, P)$ and all input values x we have that $\mathsf{Time}_{P'}(x) \in \mathcal{O}(\mathsf{poly}(\lambda + \mathsf{Time}_P(x)))$.
- SECURITY. For any pair of PPT adversaries $(\mathcal{S}, \mathcal{D})$, where $\mathcal{S}$ is an equivalent sampler, i.e.,

  $$\mathsf{Adv}^{\mathrm{eq}}_{\mathcal{S}}(\lambda) := \Pr\left[\exists x \,\mathsf{s.t.}\, P_0(x) \neq P_1(x) \vee \mathsf{Time}_{P_0}(x) \neq \mathsf{Time}_{P_1}(x) : (P_0, P_1, aux) \leftarrow\!\!\$\ \mathcal{S}(1^\lambda)\right] \in \mathsf{negl},$$

  we have that

  $$\mathsf{Adv}^{\mathrm{io}}_{\mathsf{iO},\mathcal{S},\mathcal{D}}(\lambda) := 2 \cdot \Pr\left[\mathrm{IO}^{\mathcal{S},\mathcal{D}}_{\mathsf{iO}}(\lambda)\right] - 1 \in \mathsf{negl}\ ,$$

  where game IO is shown in Figure 1 on the left.

When working with circuits, the succinctness and input-specific runtime requirements follow from the facts that iO runs in polynomial time and that the size of a circuit is its runtime.

Garg et al. [GGH+13] prove that under intractability assumptions related to multi-linear maps an indistinguishability obfuscator for all $\mathcal{NC}^1$ circuits exists. Assuming the existence of a perfectly correct, levelled fully homomorphic encryption scheme and a perfectly sound non-interactive witness-indistinguishable proof system, they also show how to bootstrap the $\mathcal{NC}^1$ construction to support all polynomial-size circuits, i.e., the family $\mathcal{C} := \{\mathcal{C}_{\mathrm{p}(\lambda)}\}_{\lambda\in\mathbb{N}}$ where p is a polynomial and

$$\mathcal{C}_{\mathrm{p}(\lambda)} := \{C : C \text{ is a valid circuit of size at most } \mathrm{p}(\lambda)\}\ .$$

REMARK. For $\mathcal{C}_{\mathrm{p}(\lambda)}$, our definition is implied by the (non-uniform) definition of Garg et al. [GGH+13]. We define indistinguishability obfuscation with respect to circuit samplers that are overwhelmingly equivalent, i.e., where

$$\mathsf{Adv}^{\mathrm{eq}}_{\mathcal{S}}(\lambda) \in \mathsf{negl}\ .$$

Although we allow samplers to not always output functionally equivalent circuits, the randomized sampler only errs with negligible probability. To reduce to the definition of Garg et al., we can average over the coins of the sampler and yield a bad event analysis where the bad event captures the event that the sampler outputs two circuits which are not functionally equivalent.

Several follow-up works improved the assumptions underlying indistinguishability obfuscators as well as the performance [PST13, BR14, AGIS14b, BGK+14, GLSW14b]. As mentioned above, circuits and

$\underline{\mathrm{IO}^{\mathcal{S},\mathcal{D}}_{\mathsf{iO}}(\lambda)}$

$(\mathsf{P}_0, \mathsf{P}_1, aux) \leftarrow\!\$\ \mathcal{S}(1^\lambda)$
$b \leftarrow\!\$\ \{0,1\}$
$\tilde{\mathsf{P}} \leftarrow\!\$\ \mathsf{iO}(1^\lambda, \mathsf{P}_b)$
$b' \leftarrow\!\$\ \mathcal{D}(\tilde{\mathsf{P}}, aux)$
return $(b = b')$

$\underline{\mathrm{IND\text{-}CPA}^{\mathcal{A}}_{\mathsf{PKE}}(\lambda)}$

$(sk, pk) \leftarrow\!\$\ \mathsf{PKE.Kg}(1^\lambda)$
$(m_0, m_1) \leftarrow\!\$\ \mathcal{A}(pk)$
$b \leftarrow\!\$\ \{0,1\}$
$c \leftarrow\!\$\ \mathsf{PKE.enc}(pk, m_b)$
$b' \leftarrow\!\$\ \mathcal{A}(c)$
return $(b = b')$

$\underline{\mathrm{IND}^{\mathcal{A}_1,\mathcal{A}_2}_{\mathsf{D\text{-}PKE}}(\lambda)}$

$(\mathbf{m}_0, \mathbf{m}_1) \leftarrow\!\$\ \mathcal{A}_1(1^\lambda)$
$(sk, pk) \leftarrow\!\$\ \mathsf{D\text{-}PKE.Kg}(1^\lambda)$
for $i = 1 \ldots |\mathbf{m}_0|$ **do**
$b \leftarrow\!\$\ \{0,1\}$
  $\mathbf{c}[i] \leftarrow \mathsf{D\text{-}PKE.enc}(pk, \mathbf{m}_b[i])$
$b' \leftarrow\!\$\ \mathcal{A}_2(pk, \mathbf{c})$
return $(b = b')$

Figure 1: Left: The IO game defines the security of an indistinguishability obfuscator. **Middle**: The IND-CPA game for randomized PKEs. **Right**: The IND security game for deterministic PKEs.

obfuscations thereof admit fixed-length inputs only. Ananth et al. [ABG+13] and Boyle et al. [BCP13] give constructions of indistinguishability obfuscators for Turing machines which admit inputs of arbitrary lengths. Their constructions achieve the stronger notion of *differing-inputs* (aka. extractability) obfuscation, initially also suggested in the work of Barak et al. [BGI+01, BGI+12]. This type of obfuscation can be regarded as a generalization of indistinguishability obfuscation to programs which are not necessarily functionally equivalent. We recall [ABG+13, Theorem 3] and refer the reader to the original works for details and discussion.

Theorem 2.1 (Ananth et al. [ABG+13])**.** *Under the existence of CPA-secure levelled fully homomorphic encryption, succinct non-interactive arguments of knowledge (SNARKs), differing-inputs obfuscation for all circuits in $\mathcal{P}$/poly, and collision-resistant hash functions, there exists a differing-inputs obfuscator for the class of all Turing machines $\mathcal{M} := \{\mathcal{M}_\lambda\}_{\lambda\in\mathbb{N}}$, where*

$$\mathcal{M}_\lambda := \{M : M \text{ is a valid Turing machine of description size at most } \lambda\} .$$

Public-key encryption. A public-key encryption scheme $\mathsf{PKE} := (\mathsf{PKE.Kg}, \mathsf{PKE.enc}, \mathsf{PKE.dec})$ consists of three PPT algorithms as follows. On input the security parameter, the randomized key-generation algorithm $\mathsf{PKE.Kg}(1^\lambda)$ generates a key pair (pk, sk). The randomized encryption algorithm $\mathsf{PKE.enc}(pk, m; r)$ gets a message m, a public key pk and possibly some explicit random coins r and outputs a ciphertext c. The deterministic decryption algorithm $\mathsf{PKE.dec}(c, sk)$ is given a ciphertext c and secret key sk and outputs a plaintext or a special symbol $\perp$. We denote the supported message-length by $\mathsf{PKE.il}(\lambda)$ and the length of the random string for an encryption by $\mathsf{PKE.rl}(\lambda)$.

We say that scheme PKE is *correct* if for all $\lambda \in \mathbb{N}$, all $m \in \mathsf{PKE.il}(\lambda)$, all $(sk, pk) \in [\mathsf{PKE.Kg}(1^\lambda)]$ and all $c \in [\mathsf{enc}(pk, m)]$ we have that $\mathsf{PKE.dec}(sk, c) = m$. We say that PKE is IND-CPA secure, if the advantage of any PPT adversary in the IND-CPA game (shown in Figure 1; center) defined by

$$\mathsf{Adv}^{\mathsf{ind\text{-}cpa}}_{\mathsf{PKE},\mathcal{A}}(\lambda) = 2 \cdot \Pr\left[\mathrm{IND\text{-}CPA}^{\mathcal{A}}_{\mathsf{PKE}}(\lambda)\right] - 1$$

is negligible.

Hash function families. Following [BST14] we define a function family FF as a five tuple of PPT algorithms $(\mathsf{FF.Kg}, \mathsf{FF.Ev}, \mathsf{FF.kl}, \mathsf{FF.il}, \mathsf{FF.ol})$ where the algorithms FF.kl, FF.il, and FF.ol are deterministic and on input 1^λ specify the key length, input length, and output length, respectively. The key-generation algorithm FF.Kg gets the security parameter 1^λ as input and outputs a key $\mathsf{fk} \in \{0,1\}^{\mathsf{FF.kl}(\lambda)}$. The deterministic evaluation algorithm FF.Ev takes as input the security parameter 1^λ, a key fk, a message $x \in \{0,1\}^{\mathsf{FF.il}(\lambda)}$ and generates a hash value $\mathsf{FF.Ev}(1^\lambda, \mathsf{fk}, x) \in \{0,1\}^{\mathsf{FF.ol}(\lambda)}$.

Pseudorandom functions and generators. We say that a function family FF is pseudorandom if for any PPT Turing machine $\mathcal{A}$ we have that

$$\mathsf{Adv}^{\mathrm{prf}}_{\mathsf{FF},\mathcal{A}}(\lambda) := \Pr\left[\mathcal{A}^{\mathsf{FF.Ev}(\mathsf{fk},\cdot)}(1^\lambda) = 1\right] - \Pr\left[\mathcal{A}^{\mathsf{RO}(\cdot)}(1^\lambda) = 1\right] \in \mathsf{negl} \ .$$

In the first term, the probability is taken over a random choice of a key $\mathsf{fk} \in \{0,1\}^{\mathsf{FF.kl}(\lambda)}$ and in the second over a random choice of a function RO with domain $\{0,1\}^{\mathsf{FF.il}}(\lambda)$ and range $\{0,1\}^{\mathsf{FF.ol}(\lambda)}$. We will often refer to function families as hash functions in this work.

We say $(\mathsf{PRG}, \mathsf{PRG.il}, \mathsf{PRG.ol})$ is a secure pseudorandom generator if PRG on a string of length $\mathsf{PRG.il}(\lambda)$ outputs a string of length $\mathsf{PRG.ol}(\lambda)$ and for any PPT Turing machine $\mathcal{A}$ we have that

$$\mathsf{Adv}^{\mathrm{prg}}_{\mathsf{PRG},\mathcal{A}}(\lambda) := \Pr\left[\mathcal{A}(1^\lambda, \mathsf{PRG}(s)) = 1 : s \leftarrow\!\!\$\, \{0,1\}^{\mathsf{PRG.il}(\lambda)}\right] - \Pr\left[\mathcal{A}(1^\lambda, y) = 1 : y \leftarrow\!\!\$\, \{0,1\}^{\mathsf{PRG.ol}(\lambda)}\right]$$

is negligible.

Keyed random oracles. As we are interested in the standard-model instantiations of random oracles via *keyed* hash functions, we use a slight generalization of the ROM whereby all parties (as usual) have oracle access to a random *keyed* function of the form

$$\mathsf{RO}(\cdot,\cdot) : \mathsf{Key}_\lambda \times \mathsf{Dom}_\lambda \to \mathsf{Rng}_\lambda \ .$$

Whether or not a party gets to see the *hash key* will depend on how the random oracle is used within a cryptosystem. For instance, if a construction appends hash keys to the public keys, a party sees the hash key as long as it also sees the public key.[3] We note that in defining the correctness and security of a cryptosystem all probability spaces are extended to include a random choice of the keyed oracle. Further note that we recover the standard unkeyed random-oracle model if the key space contains only a single key 1^λ.

(Un)instantiability. Given a scheme in the keyed random-oracle model, we consider its standard-model instantiation via (concrete) keyed hash functions. Formally, this means using a keyed hash function which has key space, domain and range that are identical to those of the oracle, using its key generation algorithm whenever a hash key is needed, and calling the evaluation routine of the hash function whenever an oracle query is placed. Given a KROM scheme and a KROM security model for it, we say that the scheme is *securely instantiable* if there exists a hash function which when used to instantiate the scheme (and the security model) results in a secure scheme (with respect to the instantiated model). Conversely, we say that a scheme is *(strongly) uninstantiable* if no hash function can securely instantiate the KROM scheme. As discussed in the introduction our uninstantiability results come in two levels of strength depending on the strength of the underlying assumption. For a polynomial p, we call a KROM scheme p-*uninstantiable*, if no hash function of size at most $\mathrm{p}(\lambda)$ securely instantiates the scheme.

If T is a transform for public-key encryption schemes we denote the random oracle scheme resulting from plugging in scheme PKE by $\mathsf{T}^{\mathsf{RO}}[\mathsf{PKE}]$. When the random oracle is instantiated by a hash function H we denote the resulting standard model scheme by $\mathsf{T}[\mathsf{PKE}, \mathsf{H}]$.

[3] Again note that we prove impossibility results and thus considering restricted adversaries (i.e., not giving every adversary access to the random oracle) strengthens the result.

REMARK. We note that random oracle transformations in the literature—for example Encrypt-with-Hash [BBO07] or the Fujisaki–Okamoto transform [FO99]—are usually not analyzed in the keyed random-oracle model, but only in the random-oracle model. This reflects the idea of a single, unkeyed hash-function. However, keyed hash-functions are usually more powerful when it comes to instantiating random oracles. Thus, this leaves open the question of how the scheme is to be instantiated with a keyed hash function, that is, how the hash key is to be generated and who gets access to it. For example if we consider a transformation from a symmetric encryption scheme, the hash key could be part of the key generation process in which case the hash key would remain hidden from the adversary, or it could be a parameter generated during setup time, in which case it would be available to an adversary. We elaborate on this in Appendix A where we examine the Randomized-Hash-then-Encrypt transformation by Bellare and Keelveedhi [BK11] and analyze it with respect to both possibilities. Note that all our results also apply in to the unkeyed setting.

3 Deterministic Encryption

We start by studying the Encrypt-with-Hash (EwH) transform of Bellare, Boldyreva and O'Neil [BBO07] for building deterministic encryption from standard (randomized) encryption schemes. We will show that under the existence of indistinguishability obfuscation there is an IND-CPA public-key encryption scheme that cannot be safely used within EwH. We begin by formally defining the syntax and security of deterministic PKEs and the EwH transform.

3.1 Definitions

DETERMINISTIC PUBLIC-KEY ENCRYPTION. Deterministic public-key encryption was first introduced by Bellare, Boldyreva and O'Neil [BBO07] and syntax and correctness of a deterministic public-key encryption scheme $\mathsf{D\text{-}PKE} := (\mathsf{D\text{-}PKE.Kg}, \mathsf{D\text{-}PKE.enc}, \mathsf{D\text{-}PKE.dec})$ is defined similarly to a randomized PKE scheme with the difference that the encryption routine is now deterministic, i.e., $\mathsf{D\text{-}PKE.rl}(\lambda) := 0$. Bellare et al. [BBO07] capture security via a form of semantic security called Priv. In later works Bellare et al. [BFOR08] and independently Boldyreva et al. [BFO08] introduced an indistinguishability style security notion called IND (which we adopt in this paper) and showed that it implies Priv-security. The IND security game for deterministic (public-key) encryption is formally defined in Figure 1 on the right.[4] Roughly speaking, an IND adversary $\mathcal{A} := (\mathcal{A}_1, \mathcal{A}_2)$ consists of two stages. On input the security parameter, adversary $\mathcal{A}_1$ outputs a pair of message vectors $(\mathbf{m}_0, \mathbf{m}_1)$ of the same length that have distinct components and component-wise contain messages of the same length. Furthermore, each component is required to have super-logarithmic min-entropy. This condition is formalized by requiring that for any $x \in \{0,1\}^{\mathsf{D\text{-}PKE.il}(\lambda)}$, any $b \in \{0,1\}$ and any $i \in [|\mathbf{m}_b|]$

$$\Pr\left[x = \mathbf{m}_b[i] : (\mathbf{m}_0, \mathbf{m}_1) \leftarrow\!\!\$\ \mathcal{A}_1(1^\lambda)\right] \in \mathsf{negl}\ .$$

Then a key pair $(pk, sk) \leftarrow\!\!\$\ \mathsf{D\text{-}PKE.Kg}(1^\lambda)$ is chosen and according to a secret bit b either of the two message vectors is encrypted component-wise. The second stage adversary $\mathcal{A}_2$ is run on the resulting vector of ciphertexts and the public key, and wins the game if it correctly guesses the hidden bit b. We define the advantage of an adversary $\mathcal{A}$ in the IND game (see Figure 1) against scheme D-PKE by

$$\mathsf{Adv}^{\mathrm{ind}}_{\mathsf{D\text{-}PKE},\mathcal{A}_1,\mathcal{A}_2}(\lambda) = 2 \cdot \Pr\left[\mathrm{IND}^{\mathcal{A}}_{\mathsf{D\text{-}PKE}}(\lambda)\right] - 1\ .$$

[4] Bellare et al. [BFOR08] define the IND security game with an additional zeroth stage adversary that outputs shared state for adversaries $\mathcal{A}_1$ and $\mathcal{A}_2$. This is done to facilitate the presentation of proofs and in their security definition Bellare et al. quantify only over adversaries with a trivial zeroth stage (i.e., no shared state). As we proof an impossibility result we choose the weaker definition, i.e., do not consider adversaries that are allowed shared state.

We call scheme $\mathsf{D\text{-}PKE}$ IND-secure if the advantage of any admissible PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ in the IND game is negligible.

Encrypt-with-Hash. The Encrypt-with-Hash (EwH) transform constructs a deterministic public-key encryption scheme from a (randomized) public-key encryption scheme PKE [BBO07] in the keyed random-oracle model. We assume the keyed RO to have a range which matches the randomness space of the PKE scheme, and a domain consisting of all bit strings of length $|pk| + |m|$, that is, the length of a public key of the PKE scheme plus the length of a message. The Encrypt-with-Hash transform now constructs a D-PKE scheme as follows. The key-generation generates a key pair using the key generation algorithm of the base PKE scheme as well as a hash key $\mathsf{hk} \leftarrow\!\!\$\ \mathsf{Key}_\lambda$. It returns $(sk, (\mathsf{hk}, pk))$. Algorithm $\mathsf{D\text{-}PKE.enc}^{\mathsf{RO}(\cdot,\cdot)}(m, (\mathsf{hk}, pk))$ first computes random coins $r \leftarrow \mathsf{RO}(\mathsf{hk}, pk\|m)$ and then invokes the base encryption algorithm on m and pk using coins r to generate the ciphertext. The decryption routine is identical to that of the underlying scheme (plus an additional ciphertext check where the ciphertext is recomputed and checked against the input). We denote this scheme by $\mathsf{EwH}^{\mathsf{RO}}[\mathsf{PKE}]$. We note that the original EwH transform [BBO07] uses an unkeyed oracle, and we recover this transformation for singleton key spaces $\mathsf{Key}_\lambda := \{1^\lambda\}$. The EwH transform results in an IND secure D-PKE scheme in keyed ROM when starting from an IND-CPA public-key encryptions scheme. When instantiated with a keyed hash function H we denote the resulting scheme by $\mathsf{EwH}[\mathsf{PKE}, \mathsf{H}]$ and assume that the hash key is part of the public-key. In particular this means that the first adversary does not get the hash key.

Key access in EwH. With the formalisms introduced above, both adversaries $\mathcal{A}_1$ and $\mathcal{A}_2$ get access to $\mathsf{RO}(\cdot,\cdot)$. The first-stage adversary, however, does *not* get to see hk (although the second does). One may consider a stronger model where the hash key *is* given out. For example, [BBO07] give the first stage (Priv)-adversary access to the (unkeyed) random oracle, which corresponds to giving out the hash key to the first adversary. We note that EwH meets this stronger notion of security, however, since our results are negative we use the conventional (and weaker) IND model and note that our result also rules out the stronger variant.

3.2 Uninstantiability of EwH

We start by observing that when the EwH transformation is used in conjunction with respect to an *unkeyed* random oracle a CGH-style uninstantiability result can be established [CGH98]. Put differently, this shows that the use of a keyed hash function is essential for standard-model instantiability. As in CGH we take an arbitrary PKE scheme PKE and consider a tweaked variant of it PKE' such that $\mathsf{EwH}[\mathsf{PKE}', \mathsf{H}]$ for *any* H fails to be secure. In more detail, on input a message m, public key pk and randomness r the encryption algorithm of PKE' first interprets parts of the message as the description of a hash function (together with its single key) and then checks if the provided random coins r match $\mathsf{H.Ev}(pk\|m)$. If so, it returns $0\|m$ and else it returns $1\|\mathsf{PKE.enc}(pk, m; r)$. Scheme PKE' is IND-CPA secure because the probability that a random r matches $\mathsf{H.Ev}(pk\|m)$ is $2^{-\mathsf{H.ol}(\lambda)}$, and thus negligible. On the other hand, when the random coins are generated deterministically by applying the hash function to the message and the public key, an IND adversary against $\mathsf{EwH}[\mathsf{PKE}', \mathsf{H}]$ can easily distinguish the encryptions of $m_i\|\mathsf{H}$ for any two messages m_0 and m_1 which differ on, say, their most significant bits.

Observe that the above attack generalizes to the setting where the first-stage adversary can guess the hash key that will be used within the construction with non-negligible probability. In particular, EwH is uninstantiable with respect to the stronger IND model where the first-stage adversary gets to see the hash key, even for keyed hash functions. The standard IND game, however, restricts the first-stage adversary not to learn the public key of the deterministic PKE scheme, and thus, it cannot guess the (high min-entropy) hash key.

We next show how to use indistinguishability obfuscation to extended uninstantiability to the IND model with respect to keyed hash functions. As mentioned in the introduction, our result comes in the weak and strong flavors depending on the programs that the obfuscator securely supports. Assuming iO for Turing machines we obtain a strong uninstantiability result stating that there exists an IND-CPA public-key encryption scheme that cannot be securely used in the EwH construction for any keyed hash function. Assuming the weaker notion of iO for circuits, we get the weaker p-uninstantiability for any polynomial p. In other words, we show that there exists an IND-CPA public-key encryption scheme that cannot be securely used in EwH for any hash function whose description size is at most p. This, in particular means that for any finite set of hash functions, we can give a PKE scheme that when used within EwH yields an insecure D-PKE scheme for any hash function in the set.

Theorem 3.1 (Uninstantiability of EwH)**.** *Assuming the existence of indistinguishability obfuscation for Turing machines $\mathcal{M}$ (resp.* p*-bounded circuits $\mathcal{C}_{\mathrm{p}}$), the* EwH *transform is uninstantiable (resp.* p*-uninstantiable) with respect to* IND*-security in the standard model.*

We start by giving a high-level description of the proof before presenting the formal details. Note that we may assume that an IND-CPA-secure PKE scheme exists, as otherwise uninstantiability follows trivially. This, in turn, implies that we can also assume the existence of a pseudorandom generator. Now given an IND-CPA-secure scheme PKE, we construct a tweaked scheme $\overline{\mathsf{PKE}}$ such that the D-PKE scheme $\mathsf{EwH}[\overline{\mathsf{PKE}}, \mathsf{H}]$ is not IND secure.

To construct an adversarial scheme $\overline{\mathsf{PKE}}$ we follow the CGH blueprint. The fundamental difference here is that when encrypting we do not have access to the hash key. We solve this problem by considering obfuscation of programs which take as input a hash key hk and "run the CGH attack." We do this by considering a *universal* variant of the BFM circuit (see [BFM14]) which takes as input the description of a hash function $H(\mathsf{hk}, \cdot)$, including its key, runs the code on hardcoded values m and pk, and checks if the result matches another hardcoded value r:

$$\mathsf{P}[pk, r, m]\left(\mathsf{H}(\mathsf{hk}, \cdot)\right) : \text{if } \mathsf{H}(\mathsf{hk}, pk \| m) = r \text{ return } m \text{ else return } 0 \ .$$

The tweak that we introduce in $\overline{\mathsf{PKE}}$ is that we change the encryption operation to append the ciphertext with an obfuscation of the above program $\mathsf{P}[pk, r, m]$ where values pk, r and m are hardcoded into the program.

Next we need to argue that outputting an indistinguishability obfuscation of P does not hurt the IND-CPA security of PKE. This would be the case if we can argue that for a randomly chosen r the above circuit implements the constant zero circuit Z. Indeed, for any fixed $H(\mathsf{hk}, \cdot)$ and over a random choice of r the check performed by P fails with all but negligible probability (assuming r is sufficiently longer than $|m\|pk|$). This, however, does not mean that the circuit is zero as there could *exist* a hash function $\mathsf{H}(\mathsf{hk}, \cdot)$ which passes the check. Contrary to BFM, we cannot bound the probability of existence of such a bad code via the union bound as the number of hash descriptions might exceed the size of the randomness space.

To resolve this issue, we consider a further tweak to the base scheme. We construct a scheme PKE^* which has a much smaller randomness space and runs scheme $\overline{\mathsf{PKE}}$ on coins that are generated pseudorandomly via a PRG. This means that the randomness space used by PKE is sparse within the set of all possible coins. We need to adapt the program above to cater for the new tweaks:

$$\mathsf{P}[pk, r, m]\left(\mathsf{H}(\mathsf{hk}, \cdot)\right) : \text{if } \mathsf{PRG}(\mathsf{H}(\mathsf{hk}, pk \| m)) = \mathsf{PRG}(r) \text{ return } m \text{ else return } 0 \ .$$

At this point it might appear that no progress has been made as the above program, for reasons similar to those given above, is not functionally equivalent to Z. We note, however, that for a random $s \in \{0,1\}^{\mathsf{PRG.ol}(\lambda)}$ the program

$$\mathsf{P}'[pk, s, m]\left(\mathsf{H}(\mathsf{hk}, \cdot)\right) : \text{if } \mathsf{PRG}(\mathsf{H}(\mathsf{hk}, pk \| m)) = s \text{ return } m \text{ else return } 0 \ .$$

has a description which is *indistinguishable* from that of $\mathsf{P}[pk, r, m]$ (down to the security of PRG), and furthermore *is* functionally equivalent to the zero circuit with overwhelming probability as s will be outside the range of PRG with overwhelming probability. This allows us to prove that obfuscations of P', and hence those of P as well, leak no information about the message m. This means the modified scheme PKE^* constructed above is IND-CPA secure. To finish the proof, observe that an obfuscation of P allows an adversary to break the IND property of the EwH-transformed scheme: simply run P on the descriptions of the hash function used in instantiation to recover the encrypted message.

We note that program P will be using a universal evaluator to run the input hash functions. If the (obfuscated) program is a Turing machine, it can be run on descriptions of arbitrary size, and consequently arbitrarily sized hash functions are ruled out. On the other hand, if the (obfuscated) program is a circuit, it has an a priori *fixed* input length, and thus can only be run on hash functions that can be encoded according to the input size restrictions.

Proof (of Theorem 3.1). Throughout the proof, we will state our constructions independently of the underlying model of computation and will simply speak of programs. The obfuscated programs that we consider contains a universal program (a universal Turing machine or a universal circuit evaluator) which we denote by UEval.

Let PKE be an IND-CPA-secure public-key encryption scheme, PRG be a pseudorandom generator of appropriate stretch and iO be an indistinguishability obfuscator. We define a modified PKE scheme PKE^* as follows. The key-generation algorithm is left unchanged. The modified decryption algorithm on input a ciphertext $(c_1, \overline{\mathsf{P}})$ uses the decryption routine of PKE to decrypt c_1 (and ignores $\overline{\mathsf{P}}$). The adapted encryption algorithm is as follows:

Algo. $\mathsf{PKE}^*.\mathsf{enc}(pk, m; r\|r')$	Prog. $\mathsf{P}[pk, m, s](\mathsf{H})$
$s \leftarrow\$\ \mathsf{PRG}(r)$	$r\|r' \leftarrow \mathsf{UEval}(\mathsf{H}, pk\|m)$
$c_1 \leftarrow\$\ \mathsf{PKE}.\mathsf{enc}(pk, m; s)$	$s' \leftarrow \mathsf{PRG}(r)$
$\overline{\mathsf{P}} \leftarrow\$\ \mathsf{iO}(\mathsf{P}[pk, m, s](\cdot); r')$	**if** $(s' = s)$ **then return** m
return $(c_1, \overline{\mathsf{P}})$	**return** 0

When the above construction is considered with respect to circuits, an extra parameter p specifying the size of the inputs to the universal circuit evaluator needs to be provided. This parameter thus controls the maximum size of programs that the universal circuit admits, which in our settings translates to the size of the hash function and inputs to it. Note that when the construction is considered for Turing machines, the input size is arbitrary.

We show that the above tweaked scheme PKE^* is IND-CPA secure via a sequence of four games that we describe next. We present the pseudocode in Figure 2.

Game_0**:** This game is identical to the original IND-CPA game for PKE^*.

Game_1**:** This game proceeds as Game_0 except that the randomness s is sampled uniformly at random and is no longer generated via a PRG call.

Game_2**:** This game proceeds as Game_1 except that the ciphertext component $\overline{\mathsf{P}}$ is now generated as an indistinguishability obfuscation of the constant zero-program Z (Turing machine or circuit, respectively) padded to the appropriate length (resp. running time).

We now show that each of the above transitions only negligibly changes the probability that the game returns true.

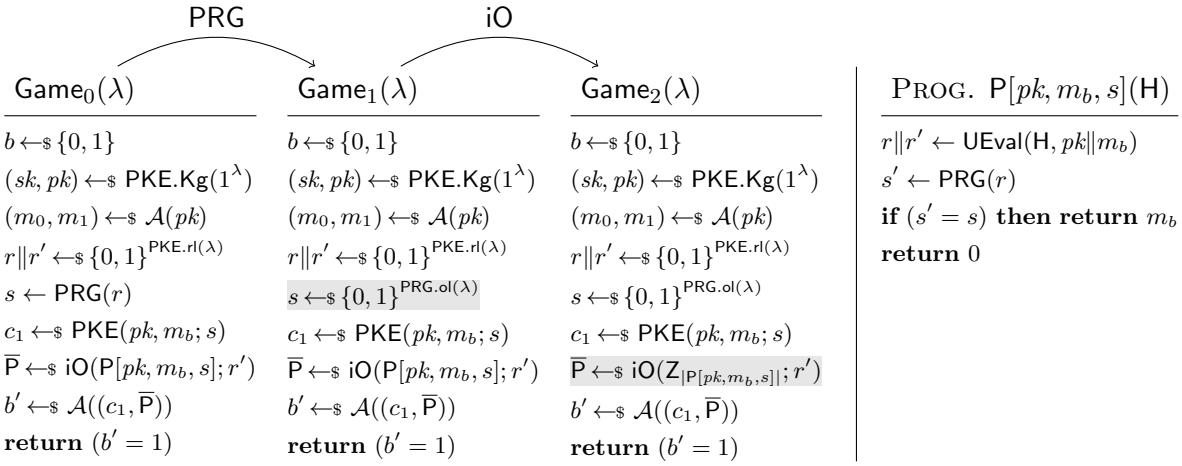

Figure 2: The hybrids for the proof of Theorem 3.1 on the left and the program P obfuscated in the first two games on the right. We have highlighted the changes between the games with a light-grey background. By $\overline{\mathsf{P}} \leftarrow \mathsf{P}[pk, m_b, s]$ we denote the generation of the program (Turing machine or circuit, respectively) which is obfuscated to yield the second part of the ciphertext for scheme PKE^*.

Game_0 to Game_1. We bound the difference in these games by the security of PRG. Note that a PRG adversary that gets as input a value y, which is either a PRG image under a uniformly random seed or a truly uniformly random value. We can perfectly simulate games Game_0 and Game_1 by using the provided y as s in the two games. If y is a PRG image, then Game_0 is run and if y is uniformly random the Game_1 is run. Hence:

$$\Pr[\,\mathsf{Game}_0(\lambda)] - \Pr[\,\mathsf{Game}_1(\lambda)] \leq \mathsf{Adv}^{\mathrm{prg}}_{\mathsf{PRG},\mathcal{A}}(\lambda)\ .$$

Game_1 to Game_2. We show that this hop negligibly affects the winning probabilities using the security of the indistinguishability obfuscator. We let $\mathcal{S}$ be the sampler which runs all the steps of Game_1 using the first phase of $\mathcal{A}$ up to the generation of $\overline{\mathsf{P}}$. It sets $\mathsf{P}_0 := \mathsf{P}$ and $\mathsf{P}_1 := \mathsf{Z}_{|\mathsf{P}_0|}$ and sets aux to be c_1 and the internal state of the first phase of the IND-CPA adversary. Algorithm $\mathcal{D}$ receives an obfuscation of either P_0 or P_1, sets $\overline{\mathsf{P}}$ to this obfuscation, and resumes the second phase of $\mathcal{A}$ on $(c_1, \overline{\mathsf{P}})$ using the state recovered from aux. When P_0 is obfuscated $\mathcal{A}$ is run according to the rules of Game_2 and when P_1 is obfuscated $\mathcal{A}$ is run according to the rules of Game_2:

$$\Pr[\,\mathsf{Game}_1(\lambda)] - \Pr[\,\mathsf{Game}_2(\lambda)] \leq \mathsf{Adv}^{\mathrm{io}}_{\mathsf{iO},\mathcal{S},\mathcal{D}}(\lambda)\ .$$

To conclude we must further argue that our sampler $\mathcal{S}$ outputs functionally equivalent circuits with overwhelming probability. To this end, we require the stretch of the PRG to be sufficiently large, i.e., $\mathsf{PRG.ol}(\lambda) \geq 2 \cdot \mathsf{PRG.il}(\lambda)$. By the union bound, the probability over a random choice of s that there *exists* an $r \in \{0,1\}^{\mathsf{PRG.il}(\lambda)}$ such that $\mathsf{PRG}(r) = s$ is upper bounded by $2^{\mathsf{PRG.il}(\lambda)-\mathsf{PRG.ol}(\lambda)} \leq 2^{-\mathsf{PRG.il}(\lambda)}$. Hence, the probability that P_0 is different from the all zero circuit is upper bounded by $2^{-\mathsf{PRG.il}(\lambda)}$, that is,

$$\Pr[\,\exists x\, \mathsf{P}_0(x) \neq 0] \leq 2^{-\mathsf{PRG.il}(\lambda)}$$

where the probability is over the random coins of sampler $\mathcal{S}$.

Game_2. We reduce the advantage of $\mathcal{A}$ in Game_2 to the IND-CPA security of the base PKE scheme PKE. To do so we note that the only difference between this game and the IND-CPA game is that an obfuscation of $\mathsf{Z}_{|\mathsf{P}[pk,m_b,s]|}$ is attached to the ciphertext. This program has a public description, and hence its obfuscation can be perfectly simulated:

$$\Pr[\,\mathsf{Game}_2(\lambda)] \leq \mathsf{Adv}^{\text{ind-cpa}}_{\mathsf{PKE}^*,\mathcal{A}}(\lambda)\ .$$

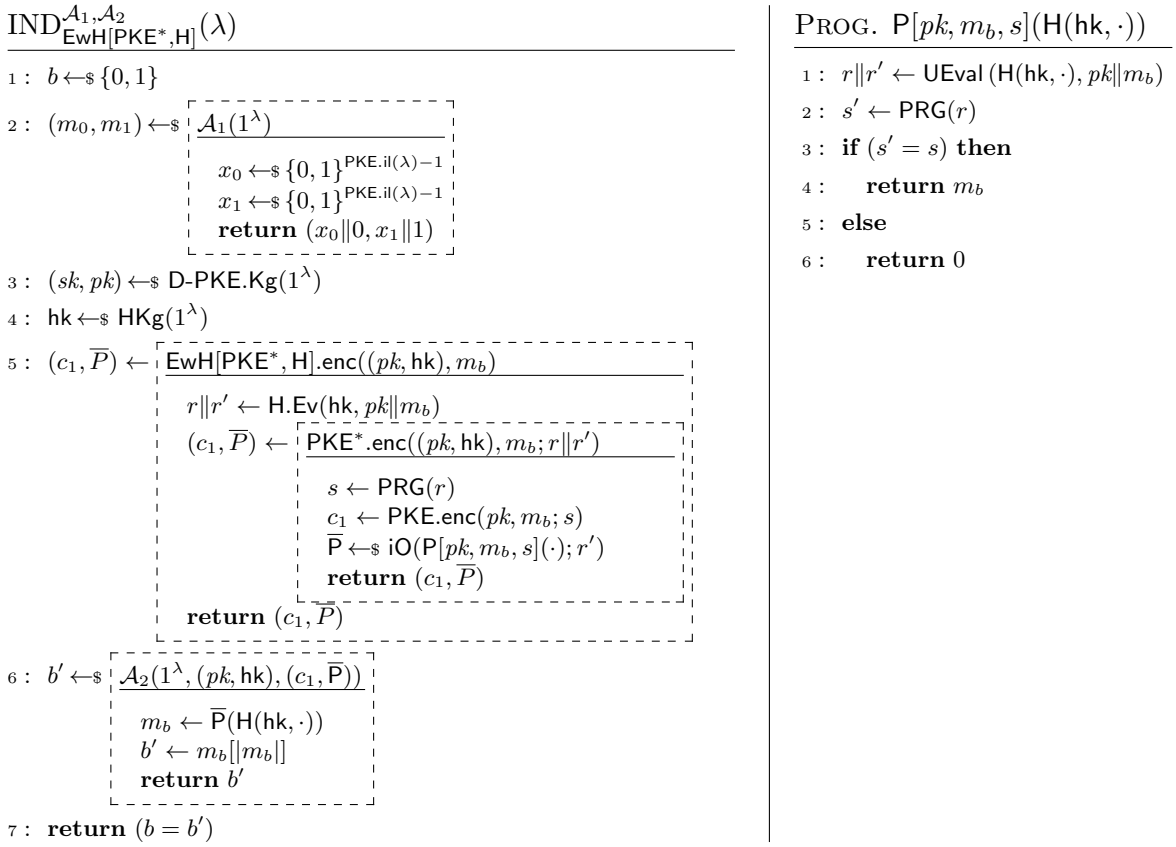

Figure 3: The IND-security game for scheme $\mathsf{EwH}[\mathsf{PKE}^*, \mathsf{H}]$ with our adversary $(\mathcal{A}_1, \mathcal{A}_2)$ as constructed in the proof of Theorem 3.1. The boxed algorithms are to be understood as subroutines. Program P that is obfuscated as part of ciphertexts is given on the right.

To conclude the proof of the theorem it suffices to show that for any function H there exists an adversary that breaks IND-security for scheme $\mathsf{EwH}[\mathsf{PKE}^*, \mathsf{H}]$, that is, the scheme constructed with the Encrypt-with-Hash transformation from our adapted PKE scheme PKE^* and hash function H. We will construct this adversary $(\mathcal{A}_1, \mathcal{A}_2)$ next and visualize the attack in Figure 3. Let $\mathsf{PKE.il}(\lambda)$ denote the message length of the resulting D-PKE scheme $\mathsf{EwH}[\mathsf{PKE}^*, \mathsf{H}]$. Adversary $\mathcal{A}_1$ chooses two values $x_0, x_1 \leftarrow\$ \{0,1\}^{\mathsf{PKE.il}(\lambda)-1}$ uniformly at random and outputs messages $m_0 := x_0\|0$ and $m_1 := x_1\|1$. Observe that $\mathcal{A}_1$ adheres to the high-entropy requirements of admissible IND adversaries, because x_0 and x_1 both have (almost) full entropy. Adversary $\mathcal{A}_2$ gets as input the public key (pk, hk) and a ciphertext $(c_1, \overline{\mathsf{P}})$. It then evaluates the program encoded in $\overline{\mathsf{P}}$ on the description of hash function $\mathsf{H}(\mathsf{hk}, \cdot)$ with key hk recovered from the public key and hardcoded into the program description. Note that if we are considering circuits, the description of this circuit must have size at most $\mathrm{p}(\lambda)$. Adversary $\mathcal{A}_2$ returns the last bit of the answer of the evaluation. This adversary always wins the IND security game irrespective of which message is encrypted under the ciphertext. That is, the check performed by the obfuscated program will always pass, and hence the least significant bit of the output of the obfuscated circuit matches the challenge bit in the IND game:

$$\mathsf{Adv}^{\mathrm{ind}}_{\mathsf{D\text{-}PKE},\mathcal{A}_1,\mathcal{A}_2}(\lambda) = 1 \ .$$

□

3.3 Consequences for UCEs

We turn to Universal Computational Extractors (UCEs) a novel notion introduced by Bellare, Hoang, and Keelveedhi (BHK) [BHK13a]. UCEs are a new set of assumptions on hash functions that can be used

to instantiate random oracles in a diverse set of applications including the EwH transform for D-PKE. In [BHK14], BHK show that if a scheme PKE is IND-CPA secure and a hash function H meets what they call UCE[$\mathcal{S}^{\text{cup}} \cap \mathcal{S}_{\mathsf{PKE}}$] security then $\mathsf{EwH}[\mathsf{PKE}, \mathsf{H}]$ is IND secure. Note that this security definition depends on the PKE scheme, because the source class $\mathcal{S}_{\mathsf{PKE}}$ runs the PKE scheme as a subroutine. Our negative result on EwH allows us to obtain the following corollary.

Corollary 3.2 (UCE[$\mathcal{S}^{\text{cup}} \cap \mathcal{S}_{\mathsf{PKE}}$] Uninstantiability)**.** *Assuming the existence of indistinguishability obfuscation for Turing machines $\mathcal{M}$ (resp.* p*-bounded circuits $\mathcal{C}_{\text{p}}$),* UCE[$\mathcal{S}^{\text{cup}} \cap \mathcal{S}_{\mathsf{PKE}}$] *security is uninstantiable (resp.* p*-uninstantiable) in the standard model.*

The security of EwH can also be based on other, stronger assumptions from the UCE-framework, see [BHK13a, BHK13b], which are also mutually exclusive with indistinguishability obfuscation [BFM14].

3.4 Extension to hedged PKEs

Hedged public-key encryption, introduced by Bellare et al. [BBN+09] models the security of public-key encryption schemes where the random coins used in encryption have low (or possibly no) entropy. Indistinguishability under a chosen-distribution attacks (IND-CDA) formalize what it means for a hedged PKE to be secure (see Figure 4). This notion is defined similarly to the IND game for D-PKEs, with the only difference being that the adversary, additionally to the two message vectors, also outputs a randomness vector which is used for the encryption operation. The high min-entropy restriction is spread over the message and randomness vectors. When the length of the randomness entries is 0, one recovers the IND model for D-PKEs. A transform similar to EwH can be defined for Hedged PKEs: hash the message, public key and the randomness to obtain new coins, and use them in encryption which yields the Randomized-Encrypt-with-Hash transformation of [BBN+09]. Our uninstantiability result can be adapted to this transform by considering a construction similar to before but with a slightly changed program:

PROG. $\mathsf{P}[pk, m, s](\mathsf{H}, \rho)$

$r \leftarrow \mathsf{UEval}(\mathsf{H}, pk \| m \| \rho)$
$s' \leftarrow \mathsf{PRG}(r)$
if $(s' = s)$ **then return** m
return 0

That is, the program takes an additional parameter ρ that allows the attacker to specify the randomness. We note that this requires the adversary to choose the randomness in a predictable way, which does however not violate the min-entropy requirements as long as the min-entropy on the messages is sufficiently high. We note that if one strengthens the IND-CDA notion to require that the randomness distribution needs to have super-logarithmic min-entropy that then our attacks would not work any longer.

4 Uninstantiability beyond EwH

In this section we show that our uninstantiability result applies to a wide class of transformations including the classical and widely deployed Fujisaki–Okamoto transformation [FO99].

4.1 Generalizing Encrypt-with-Hash

Let $\mathsf{RT}^{\mathsf{RO}}[\mathsf{PKE}]$ be a transformation in the ROM mapping PKE schemes to PKE schemes.[5] When the random oracle in transformation $\mathsf{RT}^{\mathsf{RO}}[\mathsf{PKE}]$ is instantiated with hash function H we write $\mathsf{RT}[\mathsf{PKE}, \mathsf{H}]$.

[5] Without loss of generality we assume that there is only a single random oracle. Multiple random oracles can be simulated, for example, via domain separation.

$\underline{\text{IND-CDA}^{\mathcal{A}_1,\mathcal{A}_2}_{\mathsf{H\text{-}PKE}}(\lambda)}$

$b \leftarrow\!\!\$\ \{0,1\}$
$(\mathbf{m}_0, \mathbf{m}_1, \mathbf{r}_0) \leftarrow\!\!\$\ \mathcal{A}_1(1^\lambda)$
$(sk, pk) \leftarrow\!\!\$\ \mathsf{H\text{-}PKE.Kg}(1^\lambda)$
for $i = 1 \ldots |\mathbf{m}_0|$ **do**
  $\mathbf{c}[i] \leftarrow \mathsf{H\text{-}PKE.enc}(pk, \mathbf{m}_b[i]; \mathbf{r}[i])$
$b' \leftarrow\!\!\$\ \mathcal{A}_2(pk, \mathbf{c})$
return $(b' = 1)$

Figure 4: The IND-CDA security game for hedged public-key encryption schemes here in the formulation of [RSS11] without initial adversaries. Our results carry over to a setting where an initial adversary that passes state to the first and second phase of the attack is present.

We say RT *respects its input* if it takes the following form where T^{RO} is a (possibly randomized) oracle PPT machine.

$\underline{\mathsf{RT}^{\mathsf{RO}}[\mathsf{PKE}].\mathsf{enc}(pk, m; r)}$

$(pk', m', r', c') \leftarrow \mathsf{T}^{\mathsf{RO}}(pk, m; r)$
$c \leftarrow \mathsf{PKE.enc}(pk', m'; r')$
return (c, c')

We note that in order to obtain a deterministic encryption routine, the transformation T needs to be deterministic but as we also want to capture other transformations—for example the IND-CCA transformation by Fujisaki–Okamoto [FO99]—this is not necessarily a requirement. In addition to the transformation taking the above form, we also require the transformation to admit a deterministic *recovery* algorithm $\mathcal{R}^{\mathsf{RO}}$ that on input (pk', m', c, c'), where $(pk', m', r', c') \leftarrow\!\!\$\ \mathsf{T}^{\mathsf{RO}}(pk, m, r)$ and $c \leftarrow \mathsf{PKE.enc}(pk', m'; r')$, recovers the message m and random coins r' used in the PKE encryption for every choice of (pk, m), coins r of T, and oracle RO. (Note that $\mathcal{R}^{\mathsf{RO}}$ does not get to see the secret key.) Finally, we call a transformation *admissible* if it is both *PKE respecting* **and** *recoverable*.

Let us explain how the Encrypt-with-Hash transformation falls under this more general class of transformations. In Encrypt-with-Hash the encryption operation is defined as

$$\mathsf{EwH}^{\mathsf{RO}}[\mathsf{PKE}].\mathsf{enc}(pk, m) := \mathsf{PKE.enc}(pk, m; \mathsf{RO}(pk \| m)).$$

To show that $\mathsf{EwH}^{\mathsf{RO}}[\mathsf{PKE}]$ is admissible we first need to rewrite it in the above form. For EwH the transformation T keeps message and public key, does not output any additional ciphertext component c' and outputs random coins as $\mathsf{RO}(pk \| m)$. That is, we can write

$\underline{\mathsf{RT}^{\mathsf{RO}}[\mathsf{PKE}].\mathsf{enc}(pk, m; r)}$

$(pk', m', r', c') \leftarrow$ $\underline{\mathsf{T}^{\mathsf{RO}}(pk, m)}$
  $r' \leftarrow \mathsf{RO}(pk \| m)$
  return (pk, m, r', ε)
$c \leftarrow \mathsf{PKE.enc}(pk', m'; r')$
return (c, c')

where ε denotes the empty bit-string.

We further need to specify a recovery algorithm $\mathcal{R}^{\mathsf{RO}}$ that on input (pk', m', c, c') recovers m and r'. As $m = m'$ and $pk = pk'$ recovery algorithm $\mathcal{R}^{\mathsf{RO}}$ outputs

$$\left(m', \mathsf{RO}(pk' \| m')\right).$$

We now give our generalized uninstantiability result. We note that we now consider both randomized transformations (such as the FO transformation) as well as deterministic transformations (such as EwH). In the former we show that the result is not instantiable with respect to IND-CPA security while for the latter we show that the result is not instantiable with respect to IND security.

Theorem 4.1 (Uninstantiability of Admissible Transformations)**.** *Let* PKE *be an* IND-CPA *secure PKE scheme and let* $\mathsf{RT}^{\mathsf{RO}}[\mathsf{PKE}]$ *be an admissible transformation. Assuming the existence of indistinguishability obfuscation for Turing machines* $\mathcal{M}$ *(resp.* p*-bounded circuits* $\mathcal{C}_{\mathrm{p}}$*), the* RT *transform is uninstantiable (resp.* p*-uninstantiable) in the standard model with respect to* IND *security in case* RT *is deterministic and uninstantiable (resp.* p*-uninstantiable) with respect to* IND-CPA *if* RT *is randomized.*

Proof. Similarly to the case of EwH, we will modify a scheme PKE to a tweaked variant PKE^* by attaching an obfuscation of a program that can be used to win the IND game for $\mathsf{RT}[\mathsf{PKE}, \mathsf{H}]$, in case the transformation yields a deterministic scheme, or the IND-CPA game, in case the transformation yields a randomized encryption scheme (as, for example, in the FO-transformation [FO99]).

The adapted PKE scheme PKE^* will output an additional obfuscation which depends on algorithm $\mathcal{R}$ and uses it to recompute the randomness. If the check works, the obfuscated program will, as before, output the original message m.

ALGO. $\mathsf{PKE}^*.\mathsf{enc}(pk, m; r \| r')$	PROG. $\mathsf{P}[pk, m, s, c_1](\mathsf{H}, c')$
$s \leftarrow\!\$\ \mathsf{PRG}(r)$	$(m, r\|r') \leftarrow \mathcal{R}^{\mathsf{UEval}(\mathsf{H},\cdot)}(pk, m, c_1, c')$
$c_1 \leftarrow\!\$\ \mathsf{PKE}.\mathsf{enc}(pk, m; s)$	$s' \leftarrow \mathsf{PRG}(r)$
$\overline{P} \leftarrow\!\$\ \mathsf{iO}(\mathsf{P}[pk, m, s, c_1](\cdot,\cdot); r')$	**if** $(s' = s)$ **then return** m
return $(c_1, \overline{P})$	**return** 0

The important conceptual difference here is that the program P not only takes a description of a hash function as input, but it also gets a ciphertext component as input. In other words the program allows the adversary to exploit the extra information that it has, namely a ciphertext, to break the modified scheme.

The tweaked scheme remains IND-CPA secure even in presence of obfuscations of P. The proof is analogous to that of Theorem 3.1. First we replace s with a truly random value s in generation of P which is indistinguishable down to the security of the pseudorandom generator. Next we note that this program would output a non-zero value only if s is within the range of PRG. This occurs with only a negligible probability. The remainder of the proof follows from the security of the indistinguishability obfuscator.

It remains to show that $\mathsf{RT}[\mathsf{PKE}^*, \mathsf{H}]$ is an IND-insecure deterministic scheme (assuming that T is deterministic) and in case the transformation is randomized (i.e., T is not deterministic) we show that $\mathsf{RT}[\mathsf{PKE}^*, \mathsf{H}]$ is not IND-CPA secure.

IND INSECURITY. In the following we assume that $\mathsf{RT}[\mathsf{PKE}^*, \mathsf{H}]$ is deterministic and we show that it fails to be IND-secure where IND-insecurity is proved similarly to before in Theorem 3.1. We define $\mathcal{A}_1$ to output two random messages such that one ends in 0 and the other in 1. The second stage adversary gets as input $(pk, (c, c'))$, where $c = (c_1, \overline{P})$ and $\overline{P}$ is an obfuscation of P. It constructs a description of the hash function H with the hash key hk hardcoded into the description and runs the obfuscation of P on (H, c'). It returns the least significant bit of the output. We give the pseudocode of the attack in Figure 5.

Let us analyze the case that m_0 is encrypted. In this case the adversary gets an obfuscation of $\mathsf{P}[pk', m', s, c_1]$ where pk' and m' are generated by T on input (pk, m_0) and random coins ε (note that we currently consider deterministic transformations). On input (H, c') program $\mathsf{P}[pk', m', s, c_1]$ runs the recovery algorithm $\mathcal{R}$ on input (pk', c_1, c', m') and giving it access to a hash oracle for function H. By

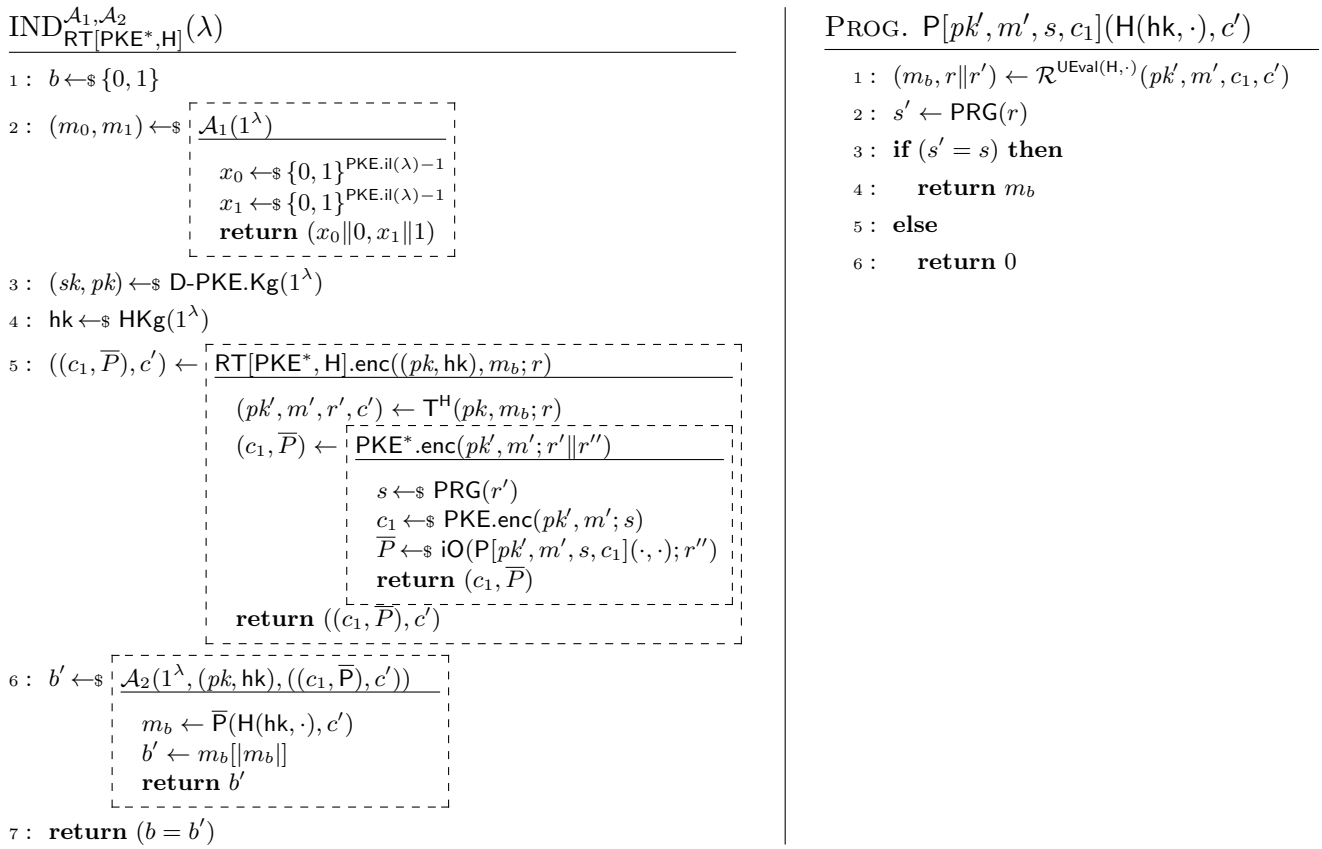

Figure 5: The IND-security game for scheme $\mathsf{RT}[\mathsf{PKE}^*, \mathsf{H}]$ with our adversary $(\mathcal{A}_1, \mathcal{A}_2)$ as constructed in the proof of Theorem 4.1. Note that for IND-security we assume that $\mathsf{RT}[\mathsf{PKE}^*, \mathsf{H}]$ is deterministic and thus r is the empty string ε.

assumption the transformation is recoverable and thus the recovery algorithm outputs $(m_0, r\|r')$ where r were the coins given to PKE^*. Program P then recomputes $\mathsf{PRG}(r)$ and as this will always match s it will output m_0.

BREAKING IND-CPA. Next we show, that assuming the transformation RT is randomized, that then the resulting scheme $\mathsf{RT}[\mathsf{PKE}^*, \mathsf{H}]$ will not be IND-CPA secure, even though $\mathsf{RT}^{\mathsf{H}}[\mathsf{PKE}]$ could still be IND-CPA secure in the ROM. To this end, consider an IND-CPA adversary $\mathcal{A}$—note that IND-CPA is defined relative to a single stateful adversary—that outputs two messages $m_0 = 0$ and $m_1 = 1$ as its chosen plaintexts in its first phase and then in its second phase launches the second stage of the above IND attack. The analysis is identical to before noting that the recovery algorithm recovers the encrypted message as well as the random coins given to the PKE encryption operation. □

Remark. We note that for our uninstantiability results it suffices if recovery algorithm $\mathcal{R}$ is probabilistic and only has noticeable success probability in recovering message m and randomness r'.

4.2 Strong IND and other transformations

The IND security game for deterministic public-key encryption is rather restrictive in that the message distribution must be completely independent of the public-key. Raghunathan et al. [RSV13] strengthen the IND definition to allow adversaries to adaptively choose messages after learning some information about the public-key. They then give constructions in the standard model and also two new constructions in the random-oracle model. The first ROM scheme generates an additional random value u as part of the public-key and generates randomness as $\mathsf{H.Ev}(\mathsf{hk}, m\|u)$; that is, the entire public-key is *not* used

for randomness generation but only a specific part of it. The second ROM scheme is parameterized by a polynomial q and generates randomness as $\oplus_{i=1}^{q+1} \mathsf{H.Ev}(\mathsf{hk}, m\|i)$. Both of these schemes fall prey to our iO-based uninstantiability results as they can be shown to be admissible similarly to the EwH transformation.

4.3 The Fujisaki–Okamoto transformation

The Fujisaki–Okamoto (FO) transformation [FO99] is a ROM technique to convert weak public-key encryption schemes, e.g., those which are indistinguishable (or even one-way) against chosen-plaintext attacks, into strong ones which resist chosen-ciphertext attacks (i.e., are IND-CCA secure). In this transform a public-key encryption scheme PKE and an additional (deterministic) symmetric encryption scheme SE and two random oracles RO_1 and RO_2 are used. The FO encryption of a message m is generated by picking a random value σ (note that FO is randomized) and computing:

$$\mathsf{FO}^{\mathsf{RO}_1,\mathsf{RO}_2}[\mathsf{PKE},\mathsf{SE}].\mathsf{enc}(pk, m; \sigma) := \mathsf{PKE.enc}(pk, \sigma; \mathsf{RO}_1(\sigma\|m)), \mathsf{SE.enc}(\mathsf{RO}_2(\sigma), m) \ .$$

In the standard model the two random oracles are replaced by keyed hash functions H and G and the hash keys are assumed to be part of the public key. We denote the standard model instantiation by $\mathsf{FO}[\mathsf{PKE},\mathsf{SE},\mathsf{H},\mathsf{G}]$.

Under the FO transform, a ciphertext for a message m is generated by picking a random value σ—value σ is chosen independently from message m—which will be hashed and then used as key for the symmetric scheme which in turn is used to encrypt the actual message m. The asymmetric scheme PKE is then used to encrypt value σ, however, in a checkable way: the randomness used to encrypt can be derived from message m and value σ.

Similarly to EwH, the FO-transformation is admissible so that Theorem 4.1 implies the following result.

Theorem 4.2 (Uninstantiability of FO)**.** *Assuming the existence of indistinguishability obfuscation for Turing machines $\mathcal{M}$ (resp.* p*-bounded circuits $\mathcal{C}_\mathrm{p}$), the* FO *transform is uninstantiable (resp.* p*-uninstantiable) with respect to* IND-CPA *in the standard model.*

Proof. To show that $\mathsf{FO}^{\mathsf{RO}_1,\mathsf{RO}_2}[\mathsf{PKE},\mathsf{SE}]$ is admissible we first give a transformation T and then a recovery algorithm $\mathcal{R}$ (in order to be consistent with earlier presentations of the FO transform we call the random coins σ):

$\underline{\textsc{Algo.}\ \mathsf{FO}^{\mathsf{RO}_1,\mathsf{RO}_2}[\mathsf{PKE},\mathsf{SE}].\mathsf{enc}(pk, m; \sigma)}$

$(pk', m', r', c') \leftarrow \underline{\mathsf{T}^{\mathsf{RO}_1,\mathsf{RO}_2}(pk, m; \sigma)}$

$m' \leftarrow \sigma$
$r' \leftarrow \mathsf{RO}_1(\sigma\|m)$
$c' = \mathsf{SE.enc}(\mathsf{RO}_2(\sigma), (m)$
return (pk, m', r', c')

$c \leftarrow \mathsf{PKE.enc}(pk', m'; r')$
return (c, c')

Next we give the recovery algorithm $\mathcal{R}^{\mathsf{RO}_1,\mathsf{RO}_2}$ which on input (pk', m', c, c') outputs the original message m and random coins r'. As $\sigma = m'$ the recovery algorithm can decrypt c' and then recompute r'. That is

$\underline{\textsc{Algo.}\ \mathcal{R}^{\mathsf{RO}_1,\mathsf{RO}_2}[\mathsf{PKE},\mathsf{SE}](pk', m', c, c')}$

$\sigma \leftarrow m'$
$m \leftarrow \mathsf{SE.dec}(\mathsf{RO}_2(\sigma), c')$
$r' \leftarrow \mathsf{RO}_1(\sigma\|m)$
return (m, r')

The proof follows with Theorem 4.1. □

Impossibility of partial instantiations. Boldyreva and Fischlin [BF05] study the security of the FO transformation when only *one* of the two random oracles in the construction is instantiated.[6] They consider perfectly one-way hash functions (POWHF) [Can97], which, on a high-level, hide all information about pre-images even given the hash key. Boldyreva and Fischlin [BF05] show that under the so-called *POWHF-encryption assumption* one can securely instantiate the H oracle in the FO transformation. The POWHF-encryption assumption asks that for any efficient message distribution $\mathcal{M}$ the following two distributions are computationally indistinguishable.

$$
\begin{array}{l}
(sk, pk) \leftarrow\!\$\ \mathsf{PKE.Kg}(1^\lambda) \\
k \leftarrow\!\$\ \mathsf{POWHF.Kg}(1^\lambda) \\
r \leftarrow\!\$\ \mathsf{POWHF.coins}(\lambda) \\
(m, \mathsf{st}) \leftarrow\!\$\ \mathcal{M}(pk, k, r) \\
\sigma \leftarrow\!\$\ \{0,1\}^\lambda \\
\omega \leftarrow \mathsf{POWHF.Ev}(k, \sigma \| m; r) \\
c \leftarrow \mathsf{PKE.enc}(pk, \sigma; \omega) \\
\textbf{return}\ (pk, k, r, \mathsf{st}, c)
\end{array}
\quad \approx \quad
\begin{array}{l}
(sk, pk) \leftarrow\!\$\ \mathsf{PKE.Kg}(1^\lambda) \\
k \leftarrow\!\$\ \mathsf{POWHF.Kg}(1^\lambda) \\
r \leftarrow\!\$\ \mathsf{POWHF.coins}(\lambda) \\
(m, \mathsf{st}) \leftarrow\!\$\ \mathcal{M}(pk, k, r) \\
\sigma \leftarrow\!\$\ \{0,1\}^\lambda \\
\omega \leftarrow\!\$\ \{0,1\}^{\mathsf{PKE.rl}(\lambda)} \\
c \leftarrow \mathsf{PKE.enc}(pk, \sigma; \omega) \\
\textbf{return}\ (pk, k, r, \mathsf{st}, c)
\end{array}
$$

Looking at the proof of Theorem 4.2 and in particular the obfuscated program P we see that it uses both random oracles, that is, the recovery algorithm $\mathcal{R}$ which is a subroutine in program P first decrypts using oracle RO_2 and then recomputes the randomness.

We can simplify this algorithm and get an impossibility result also for partial uninstantiability. Namely, instead of the decryption operation, we can hardcode the message $m_1 := 1^\lambda$ into the circuit. For this, it is crucial that now, we operate in the setting of IND-CPA security where the adversary can always submit the same two messages, say $m_0 = 0^\lambda$ and $m_1 = 1^\lambda$. In contrast, for IND-security as considered before, the messages were required to have high entropy.

Now, we will use the same obfuscated program P as in the proof of Theorem 4.2, but we will change the subroutine $\mathcal{R}$ as follows:

$$
\begin{array}{l}
\underline{\textsc{Algo}.\ \mathcal{R}^{\mathsf{RO}_1}[\mathsf{PKE}, \mathsf{SE}, m_1](pk', m', c, c')} \\
\sigma \leftarrow m' \\
r' \leftarrow \mathsf{RO}_1(\sigma \| m_1) \\
\textbf{return}\ (m_1, r')
\end{array}
$$

The second phase of IND-CPA adversary $\mathcal{A}$, as before, runs $\overline{P}$ and outputs the last bit of the result. If m_0 was encrypted it will get 0 as output and output 0. In turn, if m_1 was encrypted, it will receive 1^λ and returns 1 in this case.

Note that this way, we removed the dependency on the second random oracle altogether and thus, we can restate our result as: the first random oracle in the FO transformation is uninstantiable (resp. p-uninstantiable). Or in terms of the POWFH-encryption assumption we get that assuming the existence of indistinguishability obfuscation the POWHF-encryption assumption does not hold for all PKE schemes.

4.4 KDM security and message-locked encryption

So far we applied our techniques to transformations for (randomized) public-key encryption schemes. In Appendix A and B we show that our techniques are not limited to this setting but can be used also for other random oracle transformations. The transformations that we consider are for *deterministic and symmetric* encryption schemes and target security in the presence of key-dependent messages (Appendix A) as well as message-locked encryption for secure de-duplication schemes (Appendix B) where the encryption key is generated from the encrypted message.

[6] Note that the security analysis is still in the random-oracle model, as only one of the random oracles is instantiated.

5 Careful with Conversion

We explore new classes of transformations that lie beyond those captured by admissible transformations. In this section we present one such candidate transformation for deterministic public-key encryption which seemingly bypasses our techniques. We first show that this transformation is structurally sound by giving a ROM security proof for it. We then show how to extend our techniques to this (and potentially other) transforms. The goal of this section is to illustrate that our main technique can be tweaked and extended in many ways, e.g., by splitting the attack circuit into two. In particular, it is currently not clear to us how to design transforms that circumvent attacks of this flavour.

The underlying idea behind this new transformation, which we term Hybrid Double-Encrypt-with-Hash (HD-EwH), is to use a *fixed* symmetric encryption scheme, and furthermore ensure that the randomness and message given to the public-key scheme are separated out among two independent invocations so that it is computationally infeasible to compute one set of inputs from another.

Let PKE be a public key-encryption scheme and let H be a hash function. We define the *Hybrid Double-Encrypt-with-Hash* transformation HD-EwH[PKE, H] as follows. Key generation creates $(pk, sk) \leftarrow\!\!\$\ \mathsf{PKE.Kg}(1^\lambda)$ as well as $\mathsf{hk}_1 \leftarrow\!\!\$\ \mathsf{H.Kg}(1^\lambda)$, $\mathsf{hk}_2 \leftarrow\!\!\$\ \mathsf{H.Kg}(1^\lambda)$ and $\mathsf{gk}_1 \leftarrow\!\!\$\ \mathsf{HKg}(1^\lambda)$, $\mathsf{gk}_2 \leftarrow\!\!\$\ \mathsf{HKg}(1^\lambda)$. The public-key is set to $(pk, \mathsf{hk}, \mathsf{gk})$. An encryption of a message m consists of the following three components

$$\mathsf{PKE.enc}\left(pk, \mathsf{H}(\mathsf{hk}_1, pk\|m); \mathsf{H}(\mathsf{gk}_1, pk\|m)\right) ,$$
$$\mathsf{PKE.enc}\left(pk, \mathsf{H}(\mathsf{hk}_2, pk\|m); \mathsf{H}(\mathsf{gk}_2, pk\|m)\right) ,$$
$$m \oplus \mathsf{H}^2(\mathsf{hk}_1, pk\|m) \oplus \mathsf{H}^2(\mathsf{hk}_2, pk\|m) ,$$

where $\mathsf{H}^2(\mathsf{hk}, pk\|m) \leftarrow \mathsf{H}(\mathsf{hk}, \mathsf{H}(\mathsf{hk}, pk\|m))$.

We establish the soundness of the above transform, by showing that it indeed results in a secure D-PKE in the random-oracle model, that is, we show that the scheme is IND secure down to the IND-CPA security of the underlying PKE scheme. In the random-oracle model, we safely ignore dependance on a key, and treat $\mathsf{H}(\mathsf{hk}_1, \cdot)$, $\mathsf{H}(\mathsf{hk}_2, \cdot)$, $\mathsf{H}(\mathsf{gk}_2, \cdot)$, and $\mathsf{H}(\mathsf{gk}_2, \cdot)$ as four independent random oracles $\mathsf{H}_1, \mathsf{H}_2$ and $\mathsf{G}_1, \mathsf{G}_2$:

$$\mathsf{PKE.enc}\left(pk, \mathsf{H}_1(pk\|m); \mathsf{G}_1(pk\|m)\right), \mathsf{PKE.enc}\left(pk, (\mathsf{H}_2(pk\|m); \mathsf{G}_2(pk\|m)\right), m \oplus \mathsf{H}_1^2(pk\|m) \oplus \mathsf{H}_2^2(pk\|m) .$$

We prove the following theorem in Appendix C.

Theorem 5.1. *Let* PKE *be an* IND-CPA*-secure public-key encryption scheme. Then, in the random-oracle model, the transformation* HD-EwH[PKE] *is* IND *secure.*

It is easily seen that this transformation falls outside the realm of our generalized result (Section 4). This, however, does not mean that it cannot be attacked using indistinguishability obfuscation as we next show, that is, we show that our techniques can be extended to also cover HD-EwH. For this we will slightly generalize the construction and introduce another function F which is used to generate a one-time pad key. That is, we consider the following generalized version of $\mathsf{HD\text{-}EwH}^{\mathsf{H}_1,\mathsf{H}_2,\mathsf{G}_1,\mathsf{G}_2}[\mathsf{PKE}, \mathsf{F}]$ which on input a message m and a public key pk generates a ciphertext as:

$$\mathsf{PKE.enc}\left(pk, \mathsf{H}_1(pk\|m); \mathsf{G}_1(pk\|m)\right) ,$$
$$\mathsf{PKE.enc}\left(pk, (\mathsf{H}_2(pk\|m); \mathsf{G}_2(pk\|m)\right) ,$$
$$m \oplus \mathsf{F}\left(\mathsf{H}_1(pk\|m) \oplus \mathsf{H}_2(pk\|m)\right) .$$

Note that if we set F to $\mathsf{F}(x, x') := \mathsf{H}_1.\mathsf{Ev}(\mathsf{hk}_1, x) \oplus \mathsf{H}_2.\mathsf{Ev}(\mathsf{hk}_2, x')$ we get our original scheme.

In the following we will prove that also HD-EwH is uninstantiable. The proof technique will be similar to our previous approach in that we construct a PKE scheme PKE^* which outputs an obfuscation as part of the ciphertext. In this case, however, the scheme will not output the obfuscation of a single program, but rather the obfuscations of two independent programs P_1 and P_2. In the following figure we present our adapted scheme PKE^* (which is based on an IND-CPA secure scheme PKE) on the left and the two programs P_1 and P_2 on the right. We denote the message given to the public-key encryption scheme by x instead of m in order to be consistent with notation throughout the proof. That is, m will be the message encrypted under $\mathsf{HD\text{-}EwH}^{\mathsf{H}_1,\mathsf{H}_2,\mathsf{G}_1,\mathsf{G}_2}[\mathsf{PKE}^*,\mathsf{F}]$ while x and x' will be the messages encrypted using the pke scheme PKE^*.

ALGO. $\mathsf{PKE}^*.\mathsf{enc}(x, pk; r\|r_1\|r_2)$

$s \leftarrow \mathsf{PRG}(r)$
$c_0 \leftarrow \mathsf{PKE.enc}(pk, x; s)$
$\overline{\mathsf{P}_1} \leftarrow \mathsf{iO}(\mathsf{P1}[pk, x, s](\cdot); r_1)$
$\overline{\mathsf{P}_2} \leftarrow \mathsf{iO}(\mathsf{P2}[pk, x, s](\cdot); r_2)$
return $(c_0, \overline{\mathsf{P}_1}, \overline{\mathsf{P}_2})$

PROG. $\mathsf{P}_1[pk, x, s](\mathsf{G}_1, \mathsf{G}_2, \mathsf{F}, c, \mathsf{P2})$

$m \leftarrow \mathsf{UEval}(\mathsf{P}_2, (\mathsf{G}_2, \mathsf{F}, x, c))$
$(r\|r_1\|r_2) \leftarrow \mathsf{G}_1(pk, m)$
if $(\mathsf{PRG}(r) = s)$ **then return** m
return 0

PROG. $\mathsf{P}_2[pk, x, s](\mathsf{G}_2, \mathsf{F}, x', c)$

$m \leftarrow c \oplus \mathsf{UEval}(\mathsf{F}, (x, x'))$
$(r\|r_1\|r_2) \leftarrow \mathsf{G}_2(pk\|m)$
if $(\mathsf{PRG}(r) = s)$ **then return** m
return 0

The proof that PKE^* is still IND-CPA secure is analogous to the proof of Theorem 3.1. We rely on the indistinguishability security of the obfuscator and the security of the pseudorandom generator to show that the obfuscations of the above programs are indistinguishable from those of the zero program in the ROM. We first replace s with a truly uniform random string. This change affects any adversary's advantage with negligible probability down to the security of PRG. Now except for the unlikely event that s is in the range of PRG, both programs P1 and P2 implement the all-zero program. Hence we can replace the obfuscation of P1 and P2 by those of the zero program (padded to the right size).

We now show that using scheme PKE^* in the HD-EwH transform would yield an insecure scheme for any choice of H_1, H_2, G_1, G_2 and F.

ALGO. $\mathsf{HD\text{-}EwH}[\mathsf{H}_1, \mathsf{H}_2, \mathsf{G}_1, \mathsf{G}_2, \mathsf{F}](pk, m)$

$r_1||r_1'||r_1'' \leftarrow \mathsf{G}_1(pk\|m)$, $r_2||r_2'||r_2'' \leftarrow \mathsf{G}_2(pk\|m)$
$s_1 \leftarrow \mathsf{PRG}(r_1)$, $s_2 \leftarrow \mathsf{PRG}(r_2)$
$x_1 \leftarrow \mathsf{H}_1(pk\|m)$, $x_2 \leftarrow \mathsf{H}_2(pk\|m)$
$c_1 \leftarrow \mathsf{PKE.enc}(pk, x_1; \tilde{r}_1)$, $c_1' \leftarrow \mathsf{PKE.enc}(pk, x_2; \tilde{r}_2)$
$\overline{\mathsf{P}_1} \leftarrow \mathsf{iO}(\mathsf{P}_1[pk, x_1, s_1](\cdot); r_1')$, $\overline{\mathsf{P}_1}' \leftarrow \mathsf{iO}(\mathsf{P}_1[pk, x_2, s_2](\cdot); r_2')$
$\overline{\mathsf{P}_2} \leftarrow \mathsf{iO}(\mathsf{P}_2[pk, x_1, s_1](\cdot); r_1'')$, $\overline{\mathsf{P}_2}' \leftarrow \mathsf{iO}(\mathsf{P}_2[pk, x_2, s_2](\cdot); r_2'')$
$c \leftarrow m \oplus \mathsf{F}(x_1, x_2)$
return $((c_1, \overline{\mathsf{P}_1}, \overline{\mathsf{P}_2}), (c_1', \overline{\mathsf{P}_1}', \overline{\mathsf{P}_2}'), c)$

We construct an adversary $(\mathcal{A}_1, \mathcal{A}_2)$ against the IND security of scheme $\mathsf{HD\text{-}EwH}[\mathsf{PKE}^*, \mathsf{H}_1, \mathsf{H}_2, \mathsf{G}_1, \mathsf{G}_2, \mathsf{F}]$. The first adversary $\mathcal{A}_1$ chooses two random values $d_0, d_1 \leftarrow_\$ \{0,1\}^{\mathsf{PKE.il}(\lambda)-1}$ uniformly at random and outputs messages $m_0 \leftarrow d_0\|0$ and $m_1 \leftarrow d_1\|1$. The second adversary $\mathcal{A}_2$ then receives as input a ciphertext $((c_1, \overline{\mathsf{P}_1}, \overline{\mathsf{P}_2}), (c_1', \overline{\mathsf{P}_1}', \overline{\mathsf{P}_2}'), c)$, where components $\overline{\mathsf{P}_1}$ and $\overline{\mathsf{P}_1}'$ are obfuscations of $\mathsf{P}_1[pk, x_1, s_1]$ and $\mathsf{P}_1[pk, x_2, s_2]$ respectively, and $\overline{\mathsf{P}_2}$ and $\overline{\mathsf{P}_2}'$ are obfuscations of $\mathsf{P}_2[pk, x_1, s_1]$ and $\mathsf{P}_2[pk, x_2, s_2]$ respectively. Adversary $\mathcal{A}_2$ then runs $\overline{\mathsf{P}_1}[pk, x_1, s_1]$ on input descriptions of functions $\mathsf{G}_1(\mathsf{gk}_1, \cdot)$ and $\mathsf{G}_2(\mathsf{gk}_2, \cdot)$,

a description of function F, the ciphertext component c and the obfuscated program $\overline{\mathsf{P}_2}'[pk, x, s_2]$.[7] It returns the least significant bit of the output as its guess.

To see that this attack is successful, observe that the program consisting of composition of P_1 with P_2 as run by the adversary is, with overwhelming probability, functionally equivalent to program P below and that the least significant bit of m_b is b.

PROG. $\overline{\mathsf{P}}[pk, x_1, s_1, x_2, s_2](\mathsf{G}_1, \mathsf{G}_2, \mathsf{F}, c)$

$m \leftarrow c \oplus \mathsf{UEval}(\mathsf{F}, (x_1, x_2))$
$r_1||r_1'||r_1'' \leftarrow \mathsf{G}_1(pk\|m)$, $r_2||r_2'||r_2'' \leftarrow \mathsf{G}_2(pk\|m)$
if $(\mathsf{PRG}(r_1) \neq s_1)$ **then return** 0
if $(\mathsf{PRG}(r_2) \neq s_2)$ **then return** 0
return m

This program is the analogue of the EwH program adapted to HD-EwH. Indeed, had we access to both x_1, s_1 and x_2, s_2 in one of the runs of the encrypt algorithm, we could have directly attacked the scheme by obfuscating $\overline{\mathsf{P}}$. Since this access is (by design) denied to the scheme, we instead emulate the effect of the above program by constructing two obfuscated programs each having access to only one of x_1, s_1 or x_2, s_2. As before, the above program returns the message m when run on correct hash descriptions and the last component of ciphertext. Hence, by our choice of challenge messages, returning the least significant bit of the output message would match the hidden bit with probability one.

6 Concluding Remarks

The uninstantiability result presented in the previous section for HD-EwH serves as an example of the applicability of our techniques to a more general class of transforms beyond those captured by admissible transformations. It seems an intricate task to characterize the class of transformations which are subject to our iO-based attacks (e.g., consider extending our generalized result in Section 4 to multiple, possibly cascaded, encryptions). It is an interesting and non-trivial question to propose a D-PKE transformation that is not subject to our uninstantiability result.

One promising avenue is to build schemes based on assumptions from the framework of Universal Computational Extractors (UCEs) [BHK14]. For instance, Bellare, Hoang and Keelveedhi [BHK14] show that message-locked encryption can be based on UCE[$\mathcal{S}^{\text{sup}}$], that is, UCEs with statistically unpredictable sources. This result, however, is not generic with respect to symmetric encryption schemes but rather fixes the base symmetric scheme. Note also that iO is not known to contradict statistical UCEs [BFM14].

Alternatively, one could switch to base schemes that meet stronger notions of security. For instance, IND$-type security notions require that ciphertexts are indistinguishable from random bit strings, and to see that our results do not readily extend to this setting, note that it is unclear if obfuscation schemes can provide circuits which are indistinguishable from random strings (see also Section A.2).

Acknowledgements

Part of this work was done while Christina Brzuska was a post-doctoral researcher at Tel-Aviv University and supported by the Israel Science Foundation (grant 1076/11 and 1155/11), the Israel Ministry of Science and Technology grant 3-9094), and the German-Israeli Foundation for Scientific Research and Development (grant 1152/2011). Pooya Farshim's research was supported by EPSRC research grant

[7] Alternatively, it can also run $\overline{\mathsf{P}_1}'[pk, x_2, s_2]$ on the obfuscated program $\overline{\mathsf{P}_2}[pk, x_1, s_2]$ and hash descriptions. In either case, the modified PKE scheme must contain obfuscations of both P1 and P2.

EP/L018543/1. Arno Mittelbach was supported by CASED (www.cased.de) and the German Research Foundation (DFG) SPP 1736.

References

[ABG+13] Prabhanjan Ananth, Dan Boneh, Sanjam Garg, Amit Sahai, and Mark Zhandry. Differing-inputs obfuscation and applications. Cryptology ePrint Archive, Report 2013/689, 2013. http://eprint.iacr.org/2013/689. (Cited on pages 5, 6, 8, 9, and 10.)

[AGIS14a] Prabhanjan Ananth, Divya Gupta, Yuval Ishai, and Amit Sahai. Optimizing obfuscation: Avoiding barrington's theorem. Cryptology ePrint Archive, Report 2014/222, 2014. http://eprint.iacr.org/. (Cited on page 5.)

[AGIS14b] Prabhanjan Ananth, Divya Gupta, Yuval Ishai, and Amit Sahai. Optimizing obfuscation: Avoiding barrington's theorem. Cryptology ePrint Archive, Report 2014/222, 2014. http://eprint.iacr.org/2014/222. (Cited on page 9.)

[ARP03] Sattam S. Al-Riyami and Kenneth G. Paterson. Certificateless public key cryptography. In Chi-Sung Laih, editor, *Advances in Cryptology – ASIACRYPT 2003*, volume 2894 of *Lecture Notes in Computer Science*, pages 452–473, Taipei, Taiwan, November 30 – December 4, 2003. Springer, Berlin, Germany. (Cited on page 6.)

[BBN+09] Mihir Bellare, Zvika Brakerski, Moni Naor, Thomas Ristenpart, Gil Segev, Hovav Shacham, and Scott Yilek. Hedged public-key encryption: How to protect against bad randomness. In Mitsuru Matsui, editor, *Advances in Cryptology – ASIACRYPT 2009*, volume 5912 of *Lecture Notes in Computer Science*, pages 232–249, Tokyo, Japan, December 6–10, 2009. Springer, Berlin, Germany. (Cited on pages 6 and 18.)

[BBO07] Mihir Bellare, Alexandra Boldyreva, and Adam O'Neill. Deterministic and efficiently searchable encryption. In Alfred Menezes, editor, *Advances in Cryptology – CRYPTO 2007*, volume 4622 of *Lecture Notes in Computer Science*, pages 535–552, Santa Barbara, CA, USA, August 19–23, 2007. Springer, Berlin, Germany. (Cited on pages 6, 12, and 13.)

[BBP04] Mihir Bellare, Alexandra Boldyreva, and Adriana Palacio. An uninstantiable random-oracle-model scheme for a hybrid-encryption problem. In Christian Cachin and Jan Camenisch, editors, *Advances in Cryptology – EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 171–188, Interlaken, Switzerland, May 2–6, 2004. Springer, Berlin, Germany. (Cited on page 3.)

[BCC+14] Nir Bitansky, Ran Canetti, Henry Cohn, Shafi Goldwasser, Yael Tauman Kalai, Omer Paneth, and Alon Rosen. The impossibility of obfuscation with auxiliary input or a universal simulator. In Juan A. Garay and Rosario Gennaro, editors, *Advances in Cryptology – CRYPTO 2014, Part II*, volume 8617 of *Lecture Notes in Computer Science*, pages 71–89, Santa Barbara, CA, USA, August 17–21, 2014. Springer, Berlin, Germany. (Cited on page 4.)

[BCP13] Elette Boyle, Kai-Min Chung, and Rafael Pass. On extractability (a.k.a. differing-inputs) obfuscation. Cryptology ePrint Archive, Report 2013/650, 2013. http://eprint.iacr.org/. (Cited on page 10.)

[BCP14] Elette Boyle, Kai-Min Chung, and Rafael Pass. On extractability obfuscation. In Yehuda Lindell, editor, *TCC 2014: 11th Theory of Cryptography Conference*, volume 8349 of *Lecture Notes in Computer Science*, pages 52–73, San Diego, CA, USA, February 24–26, 2014. Springer, Berlin, Germany. (Cited on pages 5, 6, and 8.)

[BCPR14] Nir Bitansky, Ran Canetti, Omer Paneth, and Alon Rosen. On the existence of extractable one-way functions. In David B. Shmoys, editor, *46th Annual ACM Symposium on Theory of Computing*, pages 505–514, New York, NY, USA, May 31 – June 3, 2014. ACM Press. (Cited on page 4.)

[BDJR97] Mihir Bellare, Anand Desai, Eric Jokipii, and Phillip Rogaway. A concrete security treatment of symmetric encryption. In *38th Annual Symposium on Foundations of Computer Science*, pages 394–403, Miami Beach, Florida, October 19–22, 1997. IEEE Computer Society Press.

[BF01] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the Weil pairing. In Joe Kilian, editor, *Advances in Cryptology – CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 213–229, Santa Barbara, CA, USA, August 19–23, 2001. Springer, Berlin, Germany. (Cited on page 6.)

[BF05] Alexandra Boldyreva and Marc Fischlin. Analysis of random oracle instantiation scenarios for OAEP and other practical schemes. In Victor Shoup, editor, *Advances in Cryptology – CRYPTO 2005*, volume 3621 of *Lecture Notes in Computer Science*, pages 412–429, Santa Barbara, CA, USA, August 14–18, 2005. Springer, Berlin, Germany. (Cited on pages 7 and 23.)

[BFM14] Christina Brzuska, Pooya Farshim, and Arno Mittelbach. Indistinguishability obfuscation and UCEs: The case of computationally unpredictable sources. In Juan A. Garay and Rosario Gennaro, editors, *Advances in Cryptology – CRYPTO 2014, Part I*, volume 8616 of *Lecture Notes in Computer Science*, pages 188–205, Santa Barbara, CA, USA, August 17–21, 2014. Springer, Berlin, Germany. (Cited on pages 3, 4, 8, 14, 18, and 26.)

[BFO08] Alexandra Boldyreva, Serge Fehr, and Adam O'Neill. On notions of security for deterministic encryption, and efficient constructions without random oracles. In David Wagner, editor, *Advances in Cryptology – CRYPTO 2008*, volume 5157 of *Lecture Notes in Computer Science*, pages 335–359, Santa Barbara, CA, USA, August 17–21, 2008. Springer, Berlin, Germany. (Cited on pages 6 and 12.)

[BFOR08] Mihir Bellare, Marc Fischlin, Adam O'Neill, and Thomas Ristenpart. Deterministic encryption: Definitional equivalences and constructions without random oracles. In David Wagner, editor, *Advances in Cryptology – CRYPTO 2008*, volume 5157 of *Lecture Notes in Computer Science*, pages 360–378, Santa Barbara, CA, USA, August 17–21, 2008. Springer, Berlin, Germany. (Cited on pages 6 and 12.)

[BGI+01] 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, *Advances in Cryptology – CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 1–18, Santa Barbara, CA, USA, August 19–23, 2001. Springer, Berlin, Germany. (Cited on pages 9 and 10.)

[BGI+12] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. *J. ACM*, 59(2):6:1–6:48, May 2012. (Cited on pages 9 and 10.)

[BGK+14] Boaz Barak, Sanjam Garg, Yael Tauman Kalai, Omer Paneth, and Amit Sahai. Protecting obfuscation against algebraic attacks (to appear in EUROCRYPT 2014), 2014. (Cited on pages 5 and 9.)

[BHK13a] Mihir Bellare, Viet Tung Hoang, and Sriram Keelveedhi. Instantiating random oracles via UCEs. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology – CRYPTO 2013,*

Part II, volume 8043 of *Lecture Notes in Computer Science*, pages 398–415, Santa Barbara, CA, USA, August 18–22, 2013. Springer, Berlin, Germany. (Cited on pages 3, 4, 17, and 18.)

[BHK13b] Mihir Bellare, Viet Tung Hoang, and Sriram Keelveedhi. Instantiating random oracles via UCEs. Cryptology ePrint Archive, Report 2013/424., Sep 22, 2013. (Version after initial BFM attack.) `http://eprint.iacr.org/2013/424/20130924:163256`). (Cited on pages 8 and 18.)

[BHK13c] Mihir Bellare, Viet Tung Hoang, and Sriram Keelveedhi. Personal communication. Sep, 2013. (Cited on page 5.)

[BHK14] Mihir Bellare, Viet Tung Hoang, and Sriram Keelveedhi. Instantiating random oracles via UCEs. Cryptology ePrint Archive, Report 2013/424., May 20, 2014. (Latest version at the time of writing.) `http://eprint.iacr.org/2013/424`. (Cited on pages 6, 18, and 26.)

[BK11] Mihir Bellare and Sriram Keelveedhi. Authenticated and misuse-resistant encryption of key-dependent data. In Phillip Rogaway, editor, *Advances in Cryptology – CRYPTO 2011*, volume 6841 of *Lecture Notes in Computer Science*, pages 610–629, Santa Barbara, CA, USA, August 14–18, 2011. Springer, Berlin, Germany. (Cited on pages 7, 12, 32, 33, 34, and 36.)

[BKR13] Mihir Bellare, Sriram Keelveedhi, and Thomas Ristenpart. Message-locked encryption and secure deduplication. In Thomas Johansson and Phong Q. Nguyen, editors, *Advances in Cryptology – EUROCRYPT 2013*, volume 7881 of *Lecture Notes in Computer Science*, pages 296–312, Athens, Greece, May 26–30, 2013. Springer, Berlin, Germany. (Cited on pages 7, 37, and 38.)

[Bla06] John Black. The ideal-cipher model, revisited: An uninstantiable blockcipher-based hash function. In Matthew J. B. Robshaw, editor, *Fast Software Encryption – FSE 2006*, volume 4047 of *Lecture Notes in Computer Science*, pages 328–340, Graz, Austria, March 15–17, 2006. Springer, Berlin, Germany. (Cited on page 3.)

[BM14a] Christina Brzuska and Arno Mittelbach. Deterministic public-key encryption from indistinguishability obfuscation and point obfuscation. Sep, 2014. (Cited on page 6.)

[BM14b] Christina Brzuska and Arno Mittelbach. Indistinguishability obfuscation versus multi-bit point obfuscation with auxiliary input. In Palash Sarkar and Tetsu Iwata, editors, *Advances in Cryptology – ASIACRYPT 2014*, Lecture Notes in Computer Science, pages ??–??, Kaohsiung, Taiwan, December 7–11, 2014. Springer, Berlin, Germany. (Cited on pages 3 and 5.)

[BM14c] Christina Brzuska and Arno Mittelbach. Using indistinguishability obfuscation via uces. In Palash Sarkar and Tetsu Iwata, editors, *Advances in Cryptology – ASIACRYPT 2014*, Lecture Notes in Computer Science, pages ??–??, Kaohsiung, Taiwan, December 7–11, 2014. Springer, Berlin, Germany. (Cited on pages 3 and 8.)

[BR93] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In V. Ashby, editor, *ACM CCS 93: 1st Conference on Computer and Communications Security*, pages 62–73, Fairfax, Virginia, USA, November 3–5, 1993. ACM Press. (Cited on page 3.)

[BR14] Zvika Brakerski and Guy N. Rothblum. Virtual black-box obfuscation for all circuits via generic graded encoding. In Yehuda Lindell, editor, *TCC 2014: 11th Theory of Cryptography Conference*, volume 8349 of *Lecture Notes in Computer Science*, pages 1–25, San Diego, CA, USA, February 24–26, 2014. Springer, Berlin, Germany. (Cited on pages 5 and 9.)

[BST14] Mihir Bellare, Igors Stepanovs, and Stefano Tessaro. Poly-many hardcore bits for any one-way function. In Palash Sarkar and Tetsu Iwata, editors, *Advances in Cryptology – ASIACRYPT 2014*, Lecture Notes in Computer Science, pages ??–??, Kaohsiung, Taiwan, December 7–11, 2014. Springer, Berlin, Germany. (Cited on pages 3, 9, and 10.)

[Can97] Ran Canetti. Towards realizing random oracles: Hash functions that hide all partial information. In Burton S. Kaliski Jr., editor, *Advances in Cryptology – CRYPTO'97*, volume 1294 of *Lecture Notes in Computer Science*, pages 455–469, Santa Barbara, CA, USA, August 17–21, 1997. Springer, Berlin, Germany. (Cited on page 23.)

[CD08] Ran Canetti and RonnyRamzi Dakdouk. Extractable perfectly one-way functions. In Luca Aceto, Ivan Damgrd, LeslieAnn Goldberg, MagnsM. Halldrsson, Anna Inglfsdttir, and Igor Walukiewicz, editors, *Automata, Languages and Programming*, volume 5126 of *Lecture Notes in Computer Science*, pages 449–460. Springer Berlin Heidelberg, 2008. (Cited on page 4.)

[CGH98] Ran Canetti, Oded Goldreich, and Shai Halevi. The random oracle methodology, revisited (preliminary version). In *30th Annual ACM Symposium on Theory of Computing*, pages 209–218, Dallas, Texas, USA, May 23–26, 1998. ACM Press. (Cited on pages 3 and 13.)

[CGH03] Ran Canetti, Oded Goldreich, and Shai Halevi. On the random-oracle methodology as applied to length-restricted signature schemes. Cryptology ePrint Archive, Report 2003/150, 2003. `http://eprint.iacr.org/2003/150`. (Cited on page 3.)

[CHK03] Ran Canetti, Shai Halevi, and Jonathan Katz. A forward-secure public-key encryption scheme. In Eli Biham, editor, *Advances in Cryptology – EUROCRYPT 2003*, volume 2656 of *Lecture Notes in Computer Science*, pages 255–271, Warsaw, Poland, May 4–8, 2003. Springer, Berlin, Germany. (Cited on page 6.)

[DAB+02] John R. Douceur, Atul Adya, William J. Bolosky, Dan Simon, and Marvin Theimer. Reclaiming space from duplicate files in a serverless distributed file system. In *International Conference on Distributed Computing Systems*, pages 617–624, 2002. (Cited on pages 7 and 37.)

[Den02] Alexander W. Dent. Adapting the weaknesses of the random oracle model to the generic group model. In Yuliang Zheng, editor, *Advances in Cryptology – ASIACRYPT 2002*, volume 2501 of *Lecture Notes in Computer Science*, pages 100–109, Queenstown, New Zealand, December 1–5, 2002. Springer, Berlin, Germany. (Cited on page 3.)

[FO99] Eiichiro Fujisaki and Tatsuaki Okamoto. Secure integration of asymmetric and symmetric encryption schemes. In Michael J. Wiener, editor, *Advances in Cryptology – CRYPTO'99*, volume 1666 of *Lecture Notes in Computer Science*, pages 537–554, Santa Barbara, CA, USA, August 15–19, 1999. Springer, Berlin, Germany. (Cited on pages 12, 18, 19, 20, and 22.)

[FOR12] Benjamin Fuller, Adam O'Neill, and Leonid Reyzin. A unified approach to deterministic encryption: New constructions and a connection to computational entropy. In Ronald Cramer, editor, *TCC 2012: 9th Theory of Cryptography Conference*, volume 7194 of *Lecture Notes in Computer Science*, pages 582–599, Taormina, Sicily, Italy, March 19–21, 2012. Springer, Berlin, Germany. (Cited on page 6.)

[Gen03] Craig Gentry. Certificate-based encryption and the certificate revocation problem. In Eli Biham, editor, *Advances in Cryptology – EUROCRYPT 2003*, volume 2656 of *Lecture Notes in Computer Science*, pages 272–293, Warsaw, Poland, May 4–8, 2003. Springer, Berlin, Germany. (Cited on page 6.)

[GGH+13] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *54th Annual Symposium on Foundations of Computer Science*, pages 40–49, Berkeley, CA, USA, October 26–29, 2013. IEEE Computer Society Press. (Cited on pages 3, 5, 8, and 9.)

[GGHW14] Sanjam Garg, Craig Gentry, Shai Halevi, and Daniel Wichs. On the implausibility of differing-inputs obfuscation and extractable witness encryption with auxiliary input. In Juan A. Garay and Rosario Gennaro, editors, *Advances in Cryptology – CRYPTO 2014, Part I*, volume 8616 of *Lecture Notes in Computer Science*, pages 518–535, Santa Barbara, CA, USA, August 17–21, 2014. Springer, Berlin, Germany. (Cited on page 6.)

[GK03] Shafi Goldwasser and Yael Tauman Kalai. On the (in)security of the Fiat-Shamir paradigm. In *44th Annual Symposium on Foundations of Computer Science*, pages 102–115, Cambridge, Massachusetts, USA, October 11–14, 2003. IEEE Computer Society Press. (Cited on page 3.)

[GLSW14a] Craig Gentry, Allison Lewko, Amit Sahai, and Brent Waters. Indistinguishability obfuscation from the multilinear subgroup elimination assumption. Cryptology ePrint Archive, Report 2014/309, 2014. `http://eprint.iacr.org/2014/309`. (Cited on page 5.)

[GLSW14b] Craig Gentry, Allison Lewko, Amit Sahai, and Brent Waters. Indistinguishability obfuscation from the multilinear subgroup elimination assumption. Cryptology ePrint Archive, Report 2014/309, 2014. `http://eprint.iacr.org/`. (Cited on page 9.)

[GS02] Craig Gentry and Alice Silverberg. Hierarchical ID-based cryptography. In Yuliang Zheng, editor, *Advances in Cryptology – ASIACRYPT 2002*, volume 2501 of *Lecture Notes in Computer Science*, pages 548–566, Queenstown, New Zealand, December 1–5, 2002. Springer, Berlin, Germany. (Cited on page 6.)

[HSW14] Susan Hohenberger, Amit Sahai, and Brent Waters. Replacing a random oracle: Full domain hash from indistinguishability obfuscation (to appear in EUROCRYPT 2014), 2014. (Cited on page 3.)

[KRW13] Venkata Koppula, Kim Ramchen, and Brent Waters. Separations in circular security for arbitrary length key cycles. Cryptology ePrint Archive, Report 2013/683, 2013. `http://eprint.iacr.org/2013/683`. (Cited on page 4.)

[LPS04] Ben Lynn, Manoj Prabhakaran, and Amit Sahai. Positive results and techniques for obfuscation. In Christian Cachin and Jan Camenisch, editors, *Advances in Cryptology – EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 20–39, Interlaken, Switzerland, May 2–6, 2004. Springer, Berlin, Germany. (Cited on page 4.)

[MH14] Takahiro Matsuda and Goichiro Hanaoka. Chosen ciphertext security via point obfuscation. In Yehuda Lindell, editor, *TCC 2014: 11th Theory of Cryptography Conference*, volume 8349 of *Lecture Notes in Computer Science*, pages 95–120, San Diego, CA, USA, February 24–26, 2014. Springer, Berlin, Germany. (Cited on page 5.)

[Mit14] Arno Mittelbach. Salvaging indifferentiability in a multi-stage setting. In Phong Q. Nguyen and Elisabeth Oswald, editors, *Advances in Cryptology – EUROCRYPT 2014*, volume 8441 of *Lecture Notes in Computer Science*, pages 603–621, Copenhagen, Denmark, May 11–15, 2014. Springer, Berlin, Germany. (Cited on page 4.)

[MO14] Antonio Marcedone and Claudio Orlandi. Obfuscation $\rightarrow$ (IND-CPA security $\not\rightarrow$ circular security). In Michel Abdalla and Roberto De Prisco, editors, *SCN 14: 9th International*

Conference on Security in Communication Networks, volume 8642 of *Lecture Notes in Computer Science*, pages 77–90, Amalfi, Italy, September 3–5, 2014. Springer, Berlin, Germany. (Cited on page 4.)

[Nie02] Jesper Buus Nielsen. Separating random oracle proofs from complexity theoretic proofs: The non-committing encryption case. In Moti Yung, editor, *Advances in Cryptology – CRYPTO 2002*, volume 2442 of *Lecture Notes in Computer Science*, pages 111–126, Santa Barbara, CA, USA, August 18–22, 2002. Springer, Berlin, Germany. (Cited on page 3.)

[PST13] Rafael Pass, Karn Seth, and Sidharth Telang. Indistinguishability obfuscation from semantically-secure multi-linear encodings. Cryptology ePrint Archive, Report 2013/781, 2013. `http://eprint.iacr.org/2013/781`. (Cited on page 9.)

[PST14] Rafael Pass, Karn Seth, and Sidharth Telang. Indistinguishability obfuscation from semantically-secure multilinear encodings. In Juan A. Garay and Rosario Gennaro, editors, *Advances in Cryptology – CRYPTO 2014, Part I*, volume 8616 of *Lecture Notes in Computer Science*, pages 500–517, Santa Barbara, CA, USA, August 17–21, 2014. Springer, Berlin, Germany. (Cited on page 5.)

[RSS11] Thomas Ristenpart, Hovav Shacham, and Thomas Shrimpton. Careful with composition: Limitations of the indifferentiability framework. In Kenneth G. Paterson, editor, *Advances in Cryptology – EUROCRYPT 2011*, volume 6632 of *Lecture Notes in Computer Science*, pages 487–506, Tallinn, Estonia, May 15–19, 2011. Springer, Berlin, Germany. (Cited on page 19.)

[RSV13] Ananth Raghunathan, Gil Segev, and Salil P. Vadhan. Deterministic public-key encryption for adaptively chosen plaintext distributions. In Thomas Johansson and Phong Q. Nguyen, editors, *Advances in Cryptology – EUROCRYPT 2013*, volume 7881 of *Lecture Notes in Computer Science*, pages 93–110, Athens, Greece, May 26–30, 2013. Springer, Berlin, Germany. (Cited on pages 8 and 21.)

[SW05] Amit Sahai and Brent R. Waters. Fuzzy identity-based encryption. In Ronald Cramer, editor, *Advances in Cryptology – EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 457–473, Aarhus, Denmark, May 22–26, 2005. Springer, Berlin, Germany. (Cited on page 6.)

[SW14] Amit Sahai and Brent Waters. How to use indistinguishability obfuscation: deniable encryption, and more. In David B. Shmoys, editor, *46th Annual ACM Symposium on Theory of Computing*, pages 475–484, New York, NY, USA, May 31 – June 3, 2014. ACM Press. (Cited on page 5.)

[Wic13] Daniel Wichs. Barriers in cryptography with weak, correlated and leaky sources. In Robert D. Kleinberg, editor, *ITCS 2013: 4th Innovations in Theoretical Computer Science*, pages 111–126, Berkeley, CA, USA, January 9–12, 2013. Association for Computing Machinery. (Cited on page 6.)

A Key-Dependent Security

In this section, we consider uninstantiability results for generic ROM constructions with the purpose of achieving secure encryption even if the encrypted message depends on the secret key, i.e., we consider constructions that are key-dependent message (KDM) secure. The transformations that we consider apply to symmetric encryption schemes. In line with [BK11] we consider an extended notion of (deterministic)

$\underline{\mathrm{KIAE}^{\mathcal{A}}_{\mathsf{SE}}(\lambda)}$

$K \leftarrow\!\!\$\ \mathsf{SE.Kg}(1^\lambda)$
$S \leftarrow \emptyset$
$b \leftarrow\!\!\$\ \{0,1\}$
$b' \leftarrow\!\!\$\ \mathcal{A}^{\textsc{Enc},\textsc{Dec}}(1^\lambda)$
return $(b = b')$

$\underline{\textsc{Enc}(m)}$

$N \leftarrow\!\!\$\ \{0,1\}^{\mathsf{SE.nl}(\lambda)}$
$c_1 \leftarrow \mathsf{SE.enc}(K, N, m)$
$m' \leftarrow\!\!\$\ \{0,1\}^{|m|}$
$c_0 \leftarrow \mathsf{SE.enc}(K, N, m)$
$S \leftarrow S \cup \{(N, c_b)\}$
return (N, c_b)

$\underline{\textsc{Dec}(N, c)}$

if $(N, c) \in S$ **then return** $\perp$
$m \leftarrow \perp$
if $b = 1$ **then**
  $m \leftarrow \mathsf{SE.dec}(K, N, c)$
return $(m \neq \perp)$

Figure 6: The KIAE security game for authenticated encryption [BK11]. Note that [BK11] consider a slightly stronger variant where the encryption oracle responds with random strings rather than random encryptions. We denote this stronger notion by \$-KIAE.

symmetric encryption schemes that encompasses nonces. For simplicity we do not introduce the additional header field introduced in [BK11].

A symmetric encryption scheme $\mathsf{SE} = (\mathsf{SE.Kg}, \mathsf{SE.enc}, \mathsf{SE.dec})$ uses keys $k \in \{0,1\}^{\mathsf{SE.kl}(\lambda)}$ generated by the probabilistic key generation algorithm $\mathsf{SE.Kg}(1^\lambda)$. The encryption algorithm $\mathsf{SE.enc}$ takes as input a key k, a nonce $N \in \{0,1\}^{\mathsf{SE.nl}(\lambda)}$, and a message $m \in \{0,1\}^{\mathsf{SE.il}(\lambda)}$. We require that the scheme is correct, that is, for all choices of the key k, message m and nonce R it must hold that $\mathsf{SE.dec}(k, N, \mathsf{SE.enc}(k, N, m)) = m$. In general it is up to the application to ensure transport of the nonce, but for simplicity we assume that the nonce is part of the ciphertext, that is a ciphertext is a pair (N, c), where c is the actual encrypted message.

Key-independent authenticated encryption (KIAE). We consider symmetric encryption in the authenticated encryption setting as defined via the key-independent authenticated encryption (KIAE) security game in Figure 6. The game chooses a key K and gives the adversary access to two oracles Enc and Dec which are parameterized with key K and allow adversary $\mathcal{A}$ to encrypt messages of its choice and to test whether (mauled) ciphertexts are well-formed. Depending on a hidden bit b the decrypt oracle always returns $\perp$ (if $b = 0$) or if $b = 1$, it checks whether the supplied ciphertext is "fresh," decrypts it, and responds with a Boolean value indicating if decryption succeeded.[8] For the encryption oracle, according to the hidden bit b, either an encryption of the supplied message (with a fresh random nonce) or an encryption of a random plaintext (of appropriate length) is returned. We define the advantage of an adversary $\mathcal{A}$ in the KIAE game against a symmetric encryption scheme SE by

$$\mathsf{Adv}^{\mathrm{kiae}}_{\mathsf{SE},\mathcal{A}}(\lambda) := 2 \cdot \Pr\left[\mathrm{KIAE}^{\mathcal{A}}_{\mathsf{SE}}(\lambda)\right] - 1 \ .$$

We call a symmetric encryption scheme SE a secure key-independent authenticated encryption scheme (KIAE) if the advantage $\mathsf{Adv}^{\mathrm{kiae}}_{\mathsf{SE},\mathcal{A}}$ of any PPT adversary $\mathcal{A}$ is negligible. Note that we consider the nonce to be outside the control of the adversary; that is, the nonce is chosen afresh on every encryption and is appended to the output.

Key-dependent authenticated encryption (KDAE). We consider a strengthening of KIAE to the key-dependent setting, and allow the adversary to obtain encryptions of messages which are derived from the key in an adversarially specified manner. Following [BK11] we formally define the KDAE security of a symmetric encryption scheme in Figure 7. We define the advantage of an adversary in the KDAE game with a symmetric encryption scheme SE via

$$\mathsf{Adv}^{\mathrm{kdae}}_{\mathsf{SE},\mathcal{A}_1,\mathcal{A}_2}(\lambda) := 2 \cdot \Pr\left[\mathrm{KDAE}^{\mathcal{A}}_{\mathsf{SE}}(\lambda) = 1\right] - 1 \ .$$

[8] The decryption oracle, thus, models unforgeability of ciphertexts, since an adversary which manages to place a successful decryption query (i.e., one where the oracle does not return false), can detect that the bit b is set to 1.

$\underline{\text{KDAE}^{\mathcal{A}}_{\mathsf{SE},w}(\lambda)}$

for $j = 1, \ldots, w$ **do**
  $K_j \leftarrow\!\!\$\ \mathsf{SE.Kg}(1^\lambda)$
  $S_j \leftarrow\!\!\$\ \emptyset$
$b \leftarrow\!\!\$\ \{0,1\}$
$b' \leftarrow\!\!\$\ \mathcal{A}^{\text{Enc},\text{Dec}}(1^\lambda)$
return $(b = b')$

$\underline{\text{Enc}(j, \phi)}$

$m \leftarrow \phi(K_1, \ldots, K_w)$
$N \leftarrow\!\!\$\ \{0,1\}^{\mathsf{SE.nl}(\lambda)}$
$c_1 \leftarrow \mathsf{SE.enc}(K_j, N, m)$
$m' \leftarrow\!\!\$\ \{0,1\}^{|m|}$
$c_0 \leftarrow\!\!\$\ \mathsf{SE.enc}(K_j, N, m')$
$S_j \leftarrow S_j \cup \{(N, c_b)\}$
return (N, c_b)

$\underline{\text{Dec}(j, N, c)}$

if $(N, c) \in S_j$ **then return** $\perp$
$m \leftarrow \perp$
if $b = 1$ **then**
  $m \leftarrow \mathsf{SE.dec}(K_j, N, c)$
return $(m \neq \perp)$

Figure 7: The KDAE security game with procedures Enc and Dec. Note that we consider an authenticated setting and thus give the adversary access to a decryption oracle with checks ciphertext for well-formedness.

We call a scheme KDAE secure if the advantage of any PPT adversary $\mathcal{A}$ in game KDAE is negligible. Note that the KDAE game is parameterized by w which specifies the number of keys in the system. Note also that for the encryption oracle, the adversary specifies a key index as well as a function ϕ, which is applied to the keys in the system to obtain plaintext m.

The Randomized-Hash-then-Encrypt transform. Bellare and Keelveedhi [BK11] introduce the *Randomized Hash-then-Encrypt* transform (RHtE) as a means to convert a KIAE symmetric encryption scheme to one which is KDAE secure. This transformation $\mathsf{RHtE}^{\mathsf{RO}}[\mathsf{SE}]$ builds on a symmetric encryption scheme SE in the random-oracle model and the key for the hash function is chosen during setup time and assumed to be public. We will later discuss the case when the hash function key is kept private and modeled as being part of the key generation process. In the standard model the random oracle is instantiated by a keyed hash function H which yields scheme $\mathsf{RHtE}[\mathsf{SE}, \mathsf{H}]$. Encryption and decryption are defined in the following form, where the nonce underlying scheme SE is fixed and thus omitted from the description. Also note that the nonce N is outside the control of the adversary and assumed to be uniformly random for each encryption. For consistency with [BK11] we, however, add the nonce as input to the encryption operation.

$\underline{\mathsf{RHtE}[\mathsf{SE}, \mathsf{H}].\mathsf{enc}((k, \mathsf{hk}), N, m)}$

$K \leftarrow \mathsf{H.Ev}(\mathsf{hk}, N \| k)$
$c \leftarrow \mathsf{SE.enc}(K, m)$
return (c, N)

$\underline{\mathsf{RHtE}[\mathsf{SE}, \mathsf{H}].\mathsf{dec}((k, \mathsf{hk}), N, c)}$

$K \leftarrow \mathsf{H.Ev}(\mathsf{hk}, N \| k)$
$m \leftarrow \mathsf{SE.dec}(K, c)$
return m

Bellare and Keelveedhi (BK; [BK11]) show that the RHtE transform yields a \$-KDAE secure scheme, when starting from a one-time secure symmetric encryption scheme, that is, a scheme which requires indistinguishability of encryptions when only one ciphertext (encrypting one of two adversarially chosen messages) is available. Loosely speaking, one-time security is sufficient as in the RHtE transform a fresh key for the symmetric scheme SE is chosen for every new encryption. Let us note that BK consider slightly stronger variants of KIAE and KDAE that we denote by \$-KIAE and \$-KDAE where ciphertexts are indistinguishable from a random string rather than from an encryption of a random string. We note that it is not clear whether the results by BK can be extended to also hold for the case of KIAE and KDAE (rather than for \$-KIAE and \$-KDAE). We elaborate on the distinction and on the interpretation of our result in Section A.2.

Uninstantiability of RHtE. We show how to tweak any (one-time KIAE secure) scheme SE to one which is still one-time KIAE secure, but which yields an insecure scheme when used within RHtE for standard-model hash functions. As before our result comes in two flavors assuming indistinguishability obfuscation for Turing machines and circuits respectively.

Theorem A.1 (RHtE uninstantiability)**.** *Assuming the existence of indistinguishability obfuscation for Turing machines $\mathcal{M}$ (resp.* p*-bounded circuits $\mathcal{C}_{\mathrm{p}}$), the* RHtE *transform is uninstantiable (resp.* p*-uninstantiable) with respect to KDAE security in the standard model.*

Proof. First note that we cannot use our generalized result as we are now considering a transformation of a (deterministic) symmetric encryption scheme rather than a transformation of a randomized public-key encryption scheme. The overall proof strategy will however be similar to before.

Let SE be a one-time KIAE-secure symmetric-key encryption scheme with a key generation algorithm that samples a key uniformly at random from $\{0,1\}^{\mathsf{SE.kl}(\lambda)}$. Let PRG be a pseudorandom generator of appropriate stretch. The encryption routine generates a ciphertext c that consists of three components $c_1, \overline{\mathsf{P}}$ and τ where c_1 is the ciphertext generated with the underlying scheme SE and $\overline{\mathsf{P}}$ is an indistinguishability obfuscation of program $\mathsf{P}[s, N, m]$, that is, the program defined on the right which depends on the extended key s, the encrypted message m and nonce N. Since we need to ensure that the modified scheme remains (one-time) secure in the presence of a decryption oracle, an unforgeable MAC of $(c_1, \overline{\mathsf{P}})$ as tag τ is appended to the outputs. For technical reasons that we will explain later, we use a PRF instead of a MAC. Note that a PRF with long output is always a MAC, but that a MAC is not necessarily a PRF. Decryption first checks if the tag value is correct by computing a PRF of $(c_1, \overline{\mathsf{P}})$. If not it returns $\perp$. It then calls $\mathsf{SE.dec}$ on the (adapted) secret key and ciphertext c_1 to learn m.

Algo. $\mathsf{SE}^*.\mathsf{enc}(k\|\|k'\|r, N, m)$	Algo. $\mathsf{SE}^*.\mathsf{dec}(k\|\|k'\|r, c_1, \overline{\mathsf{P}}, N, \tau)$	Prog. $\mathsf{P}[s, N, m](\mathsf{H})$
$s \leftarrow \mathsf{PRG}(k)$	$\tau' \leftarrow \mathsf{PRF}(k', c_1 \| \overline{\mathsf{P}} \| N)$	$k\|\|k'\|r \leftarrow \mathsf{UEval}(\mathsf{H}, N \| m)$
$c_1 \leftarrow \mathsf{SE.enc}(s, N, m)$	**if** $(\tau \neq \tau')$ **then return** $\perp$	$s' \leftarrow \mathsf{PRG}(k)$
$\overline{\mathsf{P}} \leftarrow\!\!\$\ \mathsf{iO}(\mathsf{P}[s, N, m](\cdot); r)$	$s \leftarrow \mathsf{PRG}(k)$	**if** $(s' = s)$ **then return** m
$\tau \leftarrow \mathsf{PRF}(k', c_1 \| \overline{\mathsf{P}} \| N)$	$m \leftarrow \mathsf{SE.dec}(s, c_1, N)$	**return** 0
return $(c_1, \overline{\mathsf{P}}, N, \tau)$	**return** m	

Note that since Bellare and Keelveedhi only need one-time security, we only consider one-time security, too. As the scheme is deterministic, we will interpret parts of the key as randomness needed for obfuscation.

To reduce the KIAE security of the adapted scheme to that of the underlying scheme, we first note that the decryption oracle can be simulated by answering with $\perp$ for all queries down to the hardness of predicting PRF-values. Indeed, an adversary would need to come up with a new ciphertext $(c_1, \overline{\mathsf{P}}, N)$ and a valid tag τ on it as otherwise, by the rules of the game, the oracle will return $\perp$. Such a query can be directly used to break the pseudorandomness of the PRF. In order to show that the scheme is KIAE-secure, we follow a strategy similar to that given for EwH. Indeed, it can shown following similar steps as in proof of Theorem 3.1 that the obfuscation of P is indistinguishable from an obfuscation of the all zero functionality and hence, does not have any adverse effects on KIAE security.

We now show how to attack the KDAE security of RHtE when it is instantiated with some hash function H and scheme SE^* as constructed above. We construct an adversary $\mathcal{A}$ that uses a single key (i.e., $w = 1$) and needs a single encryption query. Adversary $\mathcal{A}$ sets ϕ to the identity function and calls $\textsc{Enc}(1, \phi)$ to receive nonce and ciphertext (N, c). It parses the ciphertext as $(c_1, \overline{\mathsf{P}}, \tau)$, interprets the second component $\overline{\mathsf{P}}$ as a program, and runs it on (an encoding of) the hash function H (with key hk hardcoded). Note that by assumption the adversary is aware of the hash key hk (we shortly discuss the case when hk is kept private). When the program returns a message m, $\mathcal{A}$ interprets this value as a key K and attempts to decrypt the challenge ciphertext. If it decrypts successfully, $\mathcal{A}$ returns 1. Otherwise, it returns 0.

Assuming the hidden bit b is 1, the ciphertext is honestly generated for message $m \leftarrow \phi(K)$. Let K be the key for scheme SE. Since ϕ has been chosen to be the identity map, message $m = K$ will be encrypted. Hence, in the RHtE-transformed scheme, key $k\|\|k'\|r \leftarrow \mathsf{H}(\mathsf{hk}, N \| K)$ will be used during

encryption. This means the first part of the ciphertext will be generated as $c_1 \leftarrow \mathsf{SE.enc}(\mathsf{PRG}(k), K)$ and the second component of the ciphertext contains an obfuscation of the program $\mathsf{P}[\mathsf{PRG}(k), N, K]$. Recalling the definition of P, this program returns the key K if the description of hash function which is equivalent to that used in the institution is passed to it. (It returns 0 otherwise.) This is because the s and s' variables used in the check are computed in exactly the same way.

In case $b = 0$, message m is set to a uniformly chosen random string, and hence the probability that m will correctly decrypt the challenge ciphertext is negligible. Hence, in this case, the adversary outputs 0 with overwhelming probability. □

A.1 Secret hash keys

In the above proof we relied upon the hash key being publicly known. An alternative way to model the instantiation of the RHtE scheme in the standard model is to model the hash key generation as part of the key generation process. That is a (secret) key (k, hk) is generated as $k \leftarrow\!\!\$\ \mathsf{SE.Kg}(1^\lambda)$ and $\mathsf{hk} \leftarrow\!\!\$\ \mathsf{H.Kg}(1^\lambda)$.

Our attack can also be mounted in this setting. For this note, that when the adversary $\mathcal{A}_1$ chooses function ϕ as the identity map, the message then becomes (k, hk). All that remains is to adapt the program to extract the hash key and plug it into the description of the hash function. In this case the adversary can thus simply send a description of the hash function without the hash key embedded.

A.2 Real-or-random security

The KIAE and KDAE definitions considered by Bellare and Keelveedhi [BK11] require that ciphertexts are indistinguishable from a *random string* rather than an encryption of a random string. We refer to these stronger variants as \$-KIAE and \$-KDAE. Bellare and Keelveedhi [BK11] show that RHtE yields a random-nonce KDAE secure scheme, if the underlying symmetric scheme is one-time \$-KIAE secure.

This raises the question if our results also apply when starting with a \$-KDAE-secure scheme. Our uninstantiability result crucially depends on embedding an obfuscation of a circuit into the ciphertext. Such an obfuscation is, however, heavily structured and it is not clear if indistinguishability obfuscation schemes exist that have an obfuscation which is indistinguishable from a uniformly random bit string. One straight forward distinguishing attack against an obfuscation scheme would be to simply execute the code. In particular, it cannot be the case, that an indistinguishability obfuscation of the all-zero circuit looks like a random string and also, the indistinguishability obfuscation of the all-one circuit looks like a random string, because clearly, the all-zero circuit and the all-one circuit are efficiently distinguishable.

However, potentially, there could be indistinguishability obfuscation schemes where obfuscations of the zero circuit look like random strings and random strings can also be "run" (and would describe a circuit that looks like the all zero circuit). In this case, the symmetric encryption scheme that we constructed would also by \$-KIAE-secure.

According to the current state-of-the-art, it is not clear whether such an indistinguishability obfuscation scheme exists. In particular, we do not understand very well whether assuming real-or-random indistinguishability obfuscation is a plausible assumption or not and we leave the study of such random looking obfuscators for future work.

Despite this, if the base scheme is \$-KIAE-secure, the tweaked scheme can be shown to have *simulatable* ciphertexts in the sense that it is possible to extend the ciphertexts to those which look indistinguishable from real tweaked ciphertexts. This is the reason why we used a PRF instead of a MAC. An inspection of the proof given by Bellare and Keelveedhi [BK11, page 25] reveals that this property suffices to extend their proof to the generalized setting, where only simulatability is required. Put differently, our uninstantiability result shows that the \$-KIAE assumption cannot be weakened to simulatable ciphertexts when the hash function is instantiated in the standard model.

$\underline{\text{PRV-CDA}^{\mathcal{A}}_{\mathsf{MLE}}(\lambda)}$

$\mathsf{prm} \leftarrow\!\!\$\ \mathsf{MLE.Pg}(\lambda)$
$b \leftarrow\!\!\$\ \{0,1\}$
$(\mathbf{m}_0, \mathbf{m}_1) \leftarrow\!\!\$\ \mathcal{A}_1(1^\lambda)$
for $i = 1 \ldots |\mathbf{m}_0|$ **do**
  $k \leftarrow \mathsf{MLE.Kg}(\mathsf{prm}, \mathbf{m}_b[i])$
  $\mathbf{c}[i] \leftarrow \mathsf{MLE.enc}(k, \mathbf{m}_b[i])$
$b' \leftarrow\!\!\$\ \mathcal{A}_2(\mathsf{prm}, \mathbf{c})$
return $(b = b')$

$\underline{\text{PRV\$-CDA}^{\mathcal{A}}_{\mathsf{MLE}}(\lambda)}$

$\mathsf{prm} \leftarrow\!\!\$\ \mathsf{MLE.Pg}(\lambda)$
$b \leftarrow\!\!\$\ \{0,1\}$
$\mathbf{m} \leftarrow\!\!\$\ \mathcal{A}_1(1^\lambda)$
for $i = 1 \ldots |\mathbf{m}|$ **do**
  $k \leftarrow \mathsf{MLE.Kg}(\mathsf{prm}, \mathbf{m}[i])$
  $\mathbf{c}_1[i] \leftarrow \mathsf{MLE.enc}(k, \mathbf{m}[i])$
  $\mathbf{c}_0[i] \leftarrow\!\!\$\ \{0,1\}^{|\mathbf{c}_1[i]|}$
$b' \leftarrow\!\!\$\ \mathcal{A}_2(\mathsf{prm}, \mathbf{c}_b)$
return $(b = b')$

Figure 8: The MLE security games PRV-CDA$_{\mathsf{MLE}}$ and PRV\$-CDA$_{\mathsf{MLE}}$ of BKR [BKR13]. We here give a slightly simpler variation where the adversaries are not allowed to communicate in the clear. As we give an impossibility result, this strengthens our result as it also rules the weaker notion.

B Message-Locked Encryption

Message-locked encryption (MLE) is a form of deterministic symmetric encryption where the encryption key is derived deterministically from the message that is to be encrypted. This mechanism ensures that encryptions of identical plaintexts produce identical ciphertexts, thereby allowing secure (cloud) storage providers to keep a single copy of the encrypted data. MLE was first formalized by Bellare, Keelveedhi and Ristenpart (BKR) [BKR13], who defined appropriate security models and constructed schemes that meet these definitions both in the random-oracle and standard models.

BKR propose several security notions for MLEs. One is called PRV-CDA and is similar to the IND notion for deterministic PKE (see Figure 8 on the left). In place of the public key, the public parameters P are now outside the reach of the first phase of the attack adversary. These parameters are used for encryption to derive the actual encryption key. Also here, in order to rule out trivial plainchecking attacks via re-encryption, each component of the message vectors, similarly to the IND notion, need to have high min-entropy.

Convergent encryption. One transformation which is formally studied by BKR and originally proposed by Douceur et al. [DAB+02] is called convergent encryption. The convergent encryption transformation constructs an encryption scheme from a one-time secure deterministic symmetric encryption scheme SE in the random-oracle model or by implementing the random oracle RO by a hash function family H. (Note that, in contrast to Section A we now consider SE schemes that only take a key and a message but not an additional nonce.) In the random-oracle model, the convergent encryption scheme chooses the encryption key k for a message m and public parameters prm—parameters prm are chosen during setup time as a uniformly random string—as $k \leftarrow \mathsf{RO}(\mathsf{prm}\|m)$; that is the key is simply the hash of parameters and the message. In the standard model the public parameters are assumed to contain the hash key and there key k is generated as $k \leftarrow \mathsf{H.Ev}(\mathsf{hk}, m)$. For encryption and decryption the algorithms of SE are used directly without change. Note that the range of the hash function must be a subset of the scheme's key space. This is without loss of generality, because one can always interpret the randomness of the key-generation algorithm as the scheme's key. We denote the resulting scheme in the random-oracle model by $\mathsf{CE}^{\mathsf{RO}}[\mathsf{SE}]$ and by $\mathsf{CE}[\mathsf{SE}, \mathsf{H}]$ if it is in the standard model with hash function H.

Attacking PRV-CDA$_{\mathsf{MLE}}$. We now show how to apply our uninstantiability techniques to the convergent encryption transformation. In other words, we will show that this transform may yield insecure schemes when starting from a one-time, key-recovery and IND-CPA-secure scheme SE. One-time key recovery requires that in presence of at most one ciphertext, no adversary can guess the entire key but with negligible probability. One-time IND-CPA security is analogous and requires indistinguishability

when only one ciphertext (encrypting one of two adversarially chosen messages) is available. The reason that it suffices to consider one-time secure symmetric encryption schemes instead of full IND-CPA secure schemes lies in the nature of message-locked encryption where for each encryption a fresh key is chosen. Especially, when considering the random-oracle model, then for each message a uniformly random fresh key is selected by hashing the message and public parameters. Under one-time key recovery and a slightly stronger variant of IND-CPA security is the assumption under which the CE transformation is proved PRV-CDA$_{\mathsf{MLE}}$ secure in the random oracle model [BKR13]. Namely, the assumption is that ciphertexts are indistinguishable from random strings. For a discussion on the feasability of an uninstantiability result in this case, see the discussion in Appendix A.2 on the analogous topic for KDM-security.

We get the following result.

Theorem B.1 (Uninstantiability of Convergent Encryption)**.** *Assuming the existence of a one-time* IND-CPA *and key-recovery secure SE scheme* SE *and the existence indistinguishability obfuscation for Turing machines* $\mathcal{M}$ *(resp.* p*-bounded circuits* $\mathcal{C}_{\mathsf{p}}$*), the convergent encryption transform* CE *is uninstantiable (resp.* p*-uninstantiable) with respect to* PRV-CDA$_{\mathsf{MLE}}$*-security in the standard model.*

Again note, that we cannot use our generalized result as we are considering the transformation of a (deterministic) symmetric encryption scheme as for KDM in Appendix A.

We next present our construction of a one-time IND-CPA secure symmetric encryption scheme that breaks $\mathsf{CE}[\mathsf{SE},\mathsf{H}]$, that is, $\mathsf{CE}[\mathsf{SE},\mathsf{H}]$ will not be PRV-CDA$_{\mathsf{MLE}}$-secure. The idea, as before, is to append an obfuscated circuited to the ciphertext. Since in MLE the scheme is not randomized, we obtain the necessary randomness for obfuscation directly from the secret key. Let SE be a one-time IND-CPA and one-time key-recovery secure symmetric-key encryption scheme with a key generation algorithm that samples a key uniformly at random in $\{0,1\}^{\mathsf{SE.kl}(\lambda)}$ and let PRG be a pseudorandom generator of appropriate stretch. We construct SE^* as follows. Key generation will be adopted and we define encryption as:

ALGO. $\mathsf{SE}^*.\mathsf{enc}(k\|k',m)$

$s \leftarrow \mathsf{PRG}(k)$

$c_1 \leftarrow \mathsf{SE.enc}(s,m)$

$\overline{\mathsf{P}} \leftarrow\!\$\ \mathsf{iO}(\mathsf{P}[m,s](\cdot);k')$

return $(c_1,\overline{\mathsf{P}})$

PROG. $\mathsf{P}[m,s](\mathsf{H})$

$k\|k' \leftarrow \mathsf{UEval}(\mathsf{H},m)$

$s' \leftarrow \mathsf{PRG}(k)$

if $(s'=s)$ **then return** m

return 0

Proving that the above modifications do not affect the one-time IND-CPA and one-time key-recovery security of SE is analogous to the proof of IND-CPA security of the construction for deterministic public-key encryption. For this note that since we consider one-time security, each encryption is run on a freshly sampled random key. This means the key can take the role of randomness used in the proof for D-PKEs. The attack against PRV-CDA$_{\mathsf{MLE}}$ security of the modified scheme also works analogously to the D-PKE case. We present the pseudocode outline of the attack in Figure 9.

REMARK. In our adapted scheme SE^* we abuse the fact that the scheme only needs to be one-time secure, because we can get arbitrary randomness from the key. It is an interesting question whether our attack can also be mounted if the symmetric-encryption scheme satisfies stronger security notions. Note that removing the one-time restriction is delicate, because one would need to introduce fresh randomness to avoid trivial attacks (or introduce a more complex security model). Then, the syntax would be different and the transformation would not apply anymore.

Another, simpler avenue to circumvent our uninstantiability result that we also discuss in Appendix A.2 in a similar context is to make the stronger requirement that ciphertexts look random. This notion called PRV\$-CDA$_{\mathsf{MLE}}$ was also proposed by BKR for MLEs. This definition can be seen as a "real-or-random" analogue of the PRV-CDA$_{\mathsf{MLE}}$ game, and is formally defined in Figure 8 on the right. Since our tweaked symmetric encryption embeds an obfuscated circuit as part of ciphertext, it is unclear if our results

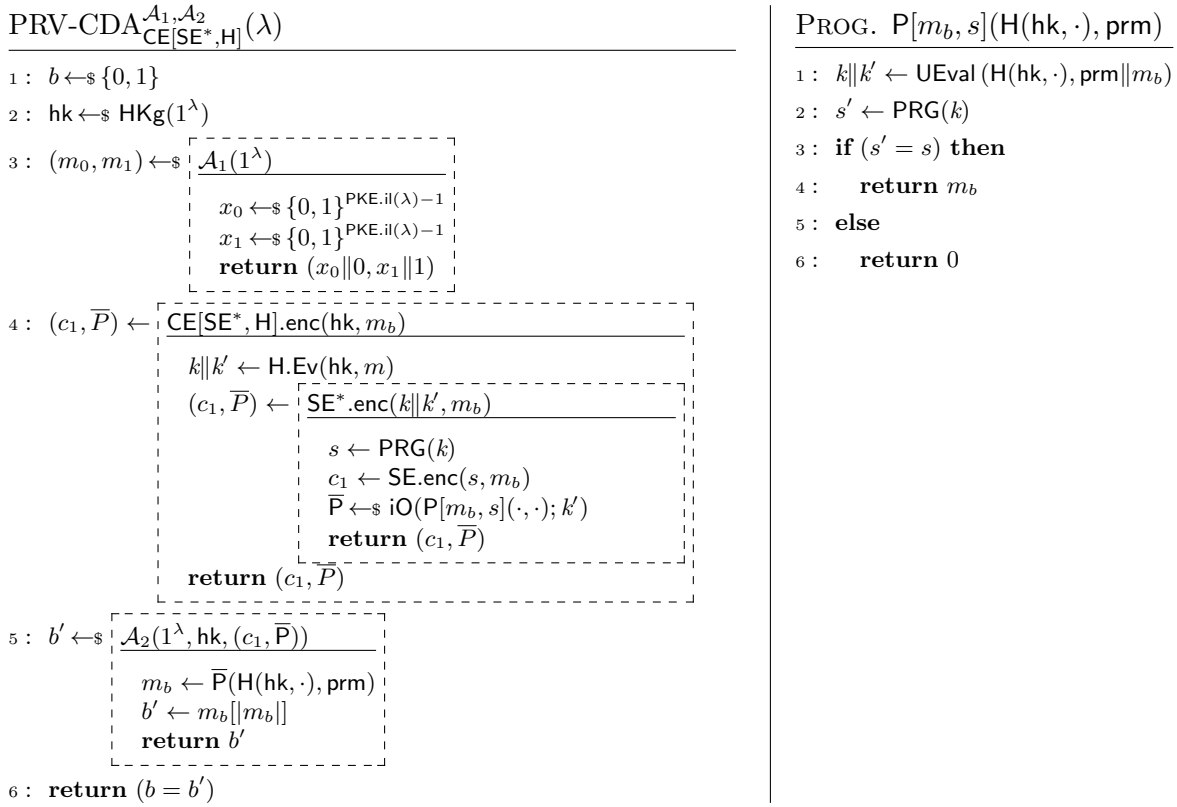

Figure 9: The PRV-CDA$_{\mathsf{MLE}}$-security game for scheme $\mathsf{CE}[\mathsf{SE}^*, \mathsf{H}]$ with our adversary $(\mathcal{A}_1, \mathcal{A}_2)$ as constructed in the proof of Theorem B.1. The boxed algorithms are to be understood as subroutines. Program P that is obfuscated as part of ciphertexts is given on the right. Note that the hash key has taken over the role of public parameters prm.

carry over to this stronger setting. Also see the discussion in Appendix A.2. Indeed, to prove a similar uninstantiability result in this case one would need to have a stronger obfuscator where obfuscations of the constant zero circuit look indistinguishable from random (See also the discussion in Section A.2).

C IND Security of HD-EwH in ROM

Before we present the formal proof of Theorem 5.1, we give some intuition. The idea in the upcoming proof is to turn an IND adversary into an IND-CPA adversary, that is, we are going to construct an IND-CPA adversary $\mathcal{B}$ that will use an IND adversary $(\mathcal{A}_1, \mathcal{A}_2)$ as a subroutine and simulate the random oracles H_1, H_2, G_1 and G_2.

Let us make the simplifying assumption that the second part of the IND adversary $\mathcal{A}_2$ does not make any random oracle calls. Then, the adversary $\mathcal{A}_2$ expects as input ciphertexts of the form

$$\mathsf{PKE.enc}\left(pk, \mathsf{H}_1(pk\|m); \mathsf{G}_1(pk\|m)\right), \mathsf{PKE.enc}\left(pk, \mathsf{H}_2(pk\|m); \mathsf{G}_2(pk\|m)\right), m \oplus \mathsf{H}_1^2(pk\|m) \oplus \mathsf{H}_2^2(pk\|m)\ ,$$

where the first two values are distributed as encryptions of two uniformly random values and the last component is a uniformly random value that is information-theoretically independent from the message m.

Now, let us consider what could be the first oracle query of the adversary. As m is drawn from a high-entropy distribution and as the random values are independent from m, the first query of the adversary will not contain m as a substring. However, it could be that the adversary breaks the encryption and recovers $\mathsf{H}_1(pk\|m)$ or $\mathsf{H}_2(pk\|m)$. Note that, unless the adversary makes *both* of these queries, the third component of the ciphertext is still a uniformly random string that is independent of the message. Thus, if the adversary wants to recover m, he has to make these two queries.

We can exploit this fact in the design of adversary $\mathcal{B}$, that is, $\mathcal{B}$ can choose random looking ciphertexts and embed random-looking "markers" into $\mathsf{H}_1(pk\|m)$ and $\mathsf{H}_2(pk\|m)$ to recognize which message got encrypted. We now make this intuition formal.

Proof (of Theorem 5.1). Assume there exists an adversary $(\mathcal{A}_1, \mathcal{A}_2)$ in the random-oracle model, against the IND security of $\mathsf{HD\text{-}EwH}^{\mathsf{H}_1,\mathsf{H}_2,\mathsf{G}_1,\mathsf{G}_2}[\mathsf{PKE}]$. Based on $(\mathcal{A}_1, \mathcal{A}_2)$, we construct an adversary $\mathcal{B}$ against the IND-CPA security of PKE as follows.

Adversary $\mathcal{B}_1$ gets as input a public key pk and runs $\mathcal{A}_1$ on input the security parameter, simulating random oracle queries (for oracles H and G) via lazy sampling in order to learn the equality pattern on the messages as output by $\mathcal{A}_1$. That is, when $\mathcal{A}_1$ terminates with outputs $(\mathbf{m}_0, \mathbf{m}_1)$, adversary $\mathcal{B}_1$ then samples vectors $\mathbf{m}'_0$ and $\mathbf{m}'_1$ as

$$\mathbf{m}'_0[i] \leftarrow\!\!\$\ \{0,1\}^{\mathsf{H.ol}(\lambda)} \qquad \mathbf{m}'_1[i] \leftarrow\!\!\$\ \{0,1\}^{\mathsf{H.ol}(\lambda)}$$

for all $i = 1, \ldots, |\mathbf{m}_0|$ while observing the equality pattern of $(\mathbf{m}_0, \mathbf{m}_1)$, that is, if

$$\mathbf{m}_b[i] = \mathbf{m}_b[j]$$

for some $j \neq i$ and $b \in \{0,1\}$ then $\mathbf{m}'_b[j]$ is set to $\mathbf{m}'_b[i]$. (Note that the equality pattern must be identical for $b = 0$ and $b = 1$.) Adversary $\mathcal{B}$ outputs message vectors $\mathbf{m}'_0, \mathbf{m}'_1$.

The second phase of adversary $\mathcal{B}$ then receives as input a ciphertext vector $\mathbf{c}$ which is either an encryption of message vector $\mathbf{m}'_0$ or of vector $\mathbf{m}'_1$. It then adjusts ciphertext vector $\mathbf{c}$ according to the equality of $(\mathbf{m}_0, \mathbf{m}_1)$. That is, if for some $j > i$ and $b \in \{0,1\}$,

$$\mathbf{m}_b[i] = \mathbf{m}_b[j]$$

it sets $\mathbf{c}[j] \leftarrow \mathbf{c}[i]$. Adversary $\mathcal{B}$ then constructs two additional vectors $\mathbf{c}'$ and $\mathbf{c}''$ of the same length as $\mathbf{c}$. Vector $\mathbf{c}'$ is constructed as encryptions under pk of uniformly random messages (adhering to the same equality pattern as $\mathbf{m}_b$ and vector $\mathbf{c}''$ is sampled uniformly at random.

Adversary $\mathcal{B}$ then calls adversary $\mathcal{A}_2$ on input $(pk, (\mathbf{c}, \mathbf{c}', \mathbf{c}''))$. Note that for easier notation we assume that $\mathcal{A}_2$ takes three ciphertext vectors, the first two corresponding to the public-key parts and the last one corresponding to the symmetric key part. Adversary $\mathcal{B}$ then continues answering random oracle queries using lazy sampling. If there is an oracle query q to H_1 by $\mathcal{A}_2$ such that there exists a $b \in \{0,1\}$ and an $i \in [1, \ldots, |\mathbf{m}'_b|]$ such that $q = \mathbf{m}'_b[i]$, then $\mathcal{B}$ stops and outputs b. If b is not uniquely specified, or there is no such query, or if $\mathcal{A}_2$ stops, then adversary $\mathcal{B}$ flips a bit and outputs the result.

Let us consider an adversary $(\mathcal{A}_1, \mathcal{A}_2)$ in the IND security game. Adversary $\mathcal{A}_2$ expects to receive as input the public key pk and ciphertexts of the form

$$\Big(\mathsf{PKE.enc}(pk, \mathsf{H}_1(pk\|\mathbf{m}_b); \mathsf{G}_1(pk\|\mathbf{m}_b)), \mathsf{PKE.enc}(pk, \mathsf{H}_2(pk\|\mathbf{m}_b); \mathsf{G}_2(pk\|\mathbf{m}_b)), \mathbf{m}_b \oplus \mathsf{H}_1^2(pk\|\mathbf{m}_b) \oplus \mathsf{H}_2^2(pk\|\mathbf{m}_b)\Big)$$

where the vector notation denotes that it receives a vector where the i-th entry is

$$\begin{gathered}\mathsf{PKE.enc}(pk, \mathsf{H}_1(pk\|\mathbf{m}_b[i]); \mathsf{G}_1(pk\|\mathbf{m}_b[i])),\\ \mathsf{PKE.enc}(pk, \mathsf{H}_2(pk\|\mathbf{m}_b[i]); \mathsf{G}_2(pk\|\mathbf{m}_b[i])),\\ \mathbf{m}_b[i] \oplus \mathsf{H}_1^2(pk\|\mathbf{m}_b[i]) \oplus \mathsf{H}_2^2(pk\|\mathbf{m}_b[i])\ .\end{gathered}$$

First let us argue that the simulation of our adversary $\mathcal{B}$ from the point of view of $(\mathcal{A}_1, \mathcal{A}_2)$ is perfect but for negligible probability. By assumption the public key cannot be guessed with noticeable probability and thus we have that $\mathcal{A}_1$ will not query any of the random oracles H_1, H_2, G_1, or G_2 on input $pk\|\mathbf{m}_b[i]$ for any $i \in [|\mathbf{m}_b|]$ and $b \in \{0,1\}$ with overwhelming probability. Thus, an encryption of

$\mathsf{H}_1(pk\|\mathbf{m}_b[i])$ (resp. for H_2, G_1, G_2) is distributed identically to an encryption of a uniformly random value generated with independent randomness.

Recall that $\mathcal{B}_1$ chooses its messages $\mathbf{m}'_0$ and $\mathbf{m}'_1$ uniformly at random and constructs $\mathbf{c}'$ as encryptions of uniformly random messages.

Furthermore, $\mathcal{A}_2$ expects that the last ciphertext component

$$\mathbf{m}_b[i] \oplus \mathsf{H}_1(\mathsf{H}_1(pk\|\mathbf{m}_b[i])) \oplus \mathsf{H}_2(\mathsf{H}_2(pk\|\mathbf{m}_b[i]))$$

is distributed uniformly as it is a one-time pad as long as adversary $\mathcal{A}_2$ does not query the random oracle on values $\mathsf{H}_1(\mathsf{H}_1(pk\|\mathbf{m}_b[i]))$ and $\mathsf{H}_2(\mathsf{H}_2(pk\|\mathbf{m}_b[i]))$.

Note that this also matches the simulation by $\mathcal{B}_2$. Thus, unless $\mathcal{A}_2$ queries H_1 on input $\mathsf{H}_1(pk\|\mathbf{m}_b[i])$ (which we programmed to be $\mathbf{m}'_b[i]$) the simulation is indistinguishable for adversary $\mathcal{A}_2$ and furthermore it information theoretically hides b. Thus, if $(\mathcal{A}_1, \mathcal{A}_2)$ has noticeable probability in winning the IND-security game the probability that $\mathcal{A}_2$ queries H_1 on $\mathsf{H}_1(pk\|\mathbf{m}_b[i])$ (that is, on $\mathbf{m}'_b[i]$) is also noticeable. However, if $\mathcal{A}_2$ queries H_1 on input $\mathbf{m}'_b[i]$ then $\mathcal{B}$ stops and outputs b. As $\mathbf{m}'_b$ contains no information about $\mathbf{m}'_{1-b}$ we have that $\mathcal{A}_2$ queries H_1 on any $\mathbf{m}'_{1-b}[i]$ only with negligible probability and thus if $\mathcal{A}$ wins with noticeable probability in the IND game, then so does $\mathcal{B}$ in the IND-CPA game. □