Random-Oracle Uninstantiability from Indistinguishability Obfuscation

Christina Brzuska[1], Pooya Farshim[2], and 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
mail@arno-mittelbach.de

Abstract. Assuming the existence of indistinguishability obfuscation (iO), we show that a number of prominent transformations in the random-oracle model are uninstantiable in the standard model. We start by showing that the Encrypt-with-Hash transform of Bellare, Boldyreva and O'Neill (CRYPTO 2007) for converting randomized public-key encryption schemes to deterministic ones is not instantiable in the standard model. To this end, we build on the recent work of Brzuska, Farshim and Mittelbach (CRYPTO 2014) and rely on the existence of iO for Turing machines or for circuits to derive two flavors of uninstantiability. The techniques that we use to establish this result are flexible and lend themselves to a number of other transformations such as the classical Fujisaki–Okamoto transform (CRYPTO 1998) and transformations akin to those by Bellare and Keelveedhi (CRYPTO 2011) and Douceur et al. (ICDCS 2002) for obtaining KDM-secure encryption and de-duplication schemes respectively. Our results call for a re-assessment of scheme design in the random-oracle model and highlight the need for new transforms that do not suffer from iO-based attacks.

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

1 Introduction

1.1 Background

The random-oracle model (ROM) [18] is an idealized model of computation where all parties, honest or otherwise, have oracle access to a uniformly chosen random function. Random oracles model ideal hash functions and have found a plethora of applications in cryptography. They have enabled 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 obfuscators. After designing and analyzing the scheme in the random-oracle model, one then instantiates the oracle via a concrete, possibly keyed, hash function. In this paper we revisit this methodology and 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 early attention to their standard-model instantiations. Canetti, Goldreich and Halevi (CGH) [33] demonstrated a general negative result by constructing 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. Such *uninstantiable* schemes rely on the existence of a compact description for concrete hash functions and lack of one for truly random functions. Roughly speaking, the idea is to take a secure ROM scheme and tweak it slightly so that it behaves securely unless it is run on messages that match the code of the hash function used in the instantiation, in which case it does something "obviously insecure" (e.g., returns the signing key or the message).

A number of other works have further studied uninstantiability problems associated with random oracles. In a follow-up work [34], CGH extend their result to signature schemes which only support short messages. Bellare, Boldyreva and Palacio [8] 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 Kalai [44] study the Fiat–Shamir heuristic and establish uninstantiability results for it. Nielsen [51] gives an uninstantiable cryptographic task, namely that of non-interactive, non-committing encryption, which although achievable in the ROM, is infeasible in the standard model. CGH-type uninstantiability has been adapted to other models of computations such as the ideal-cipher model [21] and the generic-group model [36].

A number of recent works have looked into ROM (un)instantiability in light of the recently proposed candidate for indistinguishability obfuscation (iO) [39]. A secure indistinguishability obfuscator guarantees that the obfuscations of any two functionally equivalent programs (modeled as circuits or Turing machines) are computationally indistinguishable. On the positive side, Hohenberger, Sahai and Waters [46] show how to instantiate the hash function in full-domain hash (FDH) signatures using iO. Bellare, Stepanovs and Tessaro [19] show the first standard-model construction for polynomially many hardcore bits for *any* one-way function. Recently, Brzuska and Mittelbach [31] have shown how to use iO to instantiate certain forms of Universal Computational Extractors (UCEs). UCE is a novel framework of security notions introduced by Bellare, Hoang and Keelveedhi [12] and can be used to generically instantiate random oracles in many protocols.

On the negative side, Brzuska, Farshim and Mittelbach [27] show that under the existence of iO, several security notions in the UCE framework are uninstantiable in the standard model, and proposed fixes to salvage many of the applications. Brzuska and Mittelbach [30] 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 [12,50] and strong multi-output bit point obfuscation [48] exist. Bitansky et al. [20] show that indistinguishability obfuscation rules out the existence of certain types of extractable one-way function families which can be constructed in the random-oracle model [32].

1.3 Our results

Our work continues the study of uninstantiability of random oracles and shows that a number of well-known and widely deployed ROM transforms are provably uninstantiable if indistinguishability obfuscators exist. More specifically, we are interested in ROM transformations T^{RO} that take as input *any* standard-model scheme S which is guaranteed to satisfy a mild form of security, and convert S into a new scheme $\mathsf{T}^{\mathsf{RO}}[\mathsf{S}]$ in the random-oracle model that meets a stronger level of security. A fundamental question for such transforms is their instantiability, 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. We show a number of negative results in this direction, which take the form: there is a mildly secure scheme S^* such that no matter which hash function H is picked, scheme $\mathsf{T}^{\mathsf{H}}[\mathsf{S}^*]$ is provably insecure.

Our results come in two flavors depending on the class of programs that the indistinguishability obfuscator supports. Assuming iO for circuits of a priori bounded size b, we show there is a ROM cryptosystem which is uninstantiable with respect to keyed hash functions of description size at most b. This means that there exists a scheme S_b such that for any hash function H of description size at most b the scheme $\mathsf{T}^{\mathsf{H}}[\mathsf{S}_\mathrm{b}]$ is insecure. This, in particular, yields an uninstantiability result for any fixed and finite set of hash functions. This result, however, does not rule out instantiating the oracle with hash functions which have larger description size and are in some sense "more complex" than the base scheme. By assuming the existence of iO for *Turing machines* we are able to further strengthen this result to one which rules out instantiations with respect to *any*, possibly scheme-dependent, hash function.

Overview of BFM. We build on techniques of Brzuska, Farshim and Mittelbach (BFM) [27] to construct our uninstantiable schemes and briefly recall their technique here. BFM utilize the power of indistinguishability obfuscation to show that a recent notion of security for hash functions known as UCE1 is uninstantiable in the standard model.[1] To this end, BFM construct an adversary

[1] In UCE1 (later renamed to UCE[$\mathcal{S}^{\mathrm{cup}}$]) 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 a random key or the random oracle. It does not get to see the hash key but can leak a message to the second-stage adversary on termination, which additionally gets the hash key and outputs a bit.

which outputs an indistinguishability obfuscation of the Boolean circuit

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

where x is a random domain point and y is the corresponding hash value which could be real or ideal. That is, the circuit $\mathsf{C}[x,y]$ has x and y hard-coded into it and gets as input a hash key hk, computes $\mathsf{H}(\mathsf{hk},x)$ and outputs 1 if and only if this value is equal to y.

BFM need to argue that an indistinguishability obfuscation of this circuit hides x whenever y is truly random (and not computed by applying the hash-function to x). They prove this by a counting argument that establishes that, under appropriate restrictions on the lengths of y and the length of the key hk, the above circuit implements the constant zero circuit with overwhelming probability. They then employ the security of the obfuscator to conclude as the zero circuit is independent of x. The restriction that they require, is that the number of hash keys hk is much smaller than the size of the range $2^{|y|}$, which means that y (with overwhelming probability) is outside the image of the function $\mathsf{H}(\cdot,x)$ that has a fixed x and maps hash-keys hk to $\mathsf{H}(\mathsf{hk},x)$. On the other hand, the above circuit returns 1 when the hash value y is computed as $\mathsf{H}(\mathsf{hk},x)$ and hk as the correct hash key is plugged into $\mathsf{C}[x,y]$.

Techniques. In our uninstantiability results for encryption, we will embed an obfuscated program into the ciphertext.[2] We now describe this program which is a *universal* variant of the BFM circuit. This program takes as input the full description of a hash function H_{hk}, including its key hk if there is one, and returns the result of running the BFM circuit on the input hash-function description. It performs the latter in the standard way by using a universal evaluator UEval, which could be a universal Turing machine or a universal circuit evaluator, depending on the considered model of computation.

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

So, the program $\mathsf{P}[x,y]$ has x and y hard-coded and takes as input a description of H_{hk}, computes $\mathsf{H}_{\mathsf{hk}}(x)$ and checks whether this value is equal to y. In other words, we no longer consider a fixed keyed hash function, but instead (potentially) look at the set of *all* hash functions on a given range and domain.[3] (Similar ideas have been used by Brzuska and Mittelbach [30] to study the feasibility of multi-bit output point function obfuscation in the presence of auxiliary inputs under the iO assumption.) Note that $\mathsf{P}[x,y]$ is either a circuit or a Turing machine depending on the underlying universal evaluator UEval. In adopting this approach, a number of technicalities need to be addressed, which we discuss next.

The second-stage adversary can no longer call the oracle. UCE1 security requires that the leaked message should be such that it does not computationally reveal any of the oracle queries when the oracle is a random function.

[2] We speak of programs which can be modeled either as circuits or as Turing machines.

[3] Alternatively, we are looking at the universal hash function.

Our ultimate goal is to derive a strong result which rules out instantiations (of a transformation) by arbitrary hash functions. This means that program P above should accept inputs of arbitrary length. This, however, lies beyond the powers of the circuit model of computation which current indistinguishability obfuscators support. We address this problem in two incomparable ways. First, we weaken target uninstantiability and under iO for circuits rule out instantiations by a priori bounded-size hash functions. Second, in order to strengthen this result to full uninstantiability, we consider a stronger form of iO which supports *Turing machines*. For our purposes, the crucial difference between iO for circuits and iO fro Turing machines is that an obfuscated Turing machine is still a Turing machine which can process inputs of *arbitrary* length. (Note that the actual Turing machine that we need to obfuscate is a universal Turing machine and has an a priori fixed size.) Our theorem statements will therefore contain two parts to reflect this trade off between the strength of assumptions and the reach of the uninstantiability result obtained.

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

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

This does not affect the success probability of the attack and allows us to argue that x remains hidden as follows: First note that the right-hand side $\mathsf{PRG}(y)$ is a constant that does not depend on the program input and can thus be hard-coded into the program. Now, in a first step we can replace the right hand-side value with a truly random value by the security of the PRG. Note that in this step we *do not* rely on the security of the obfuscator and merely use the indistinguishability of program descriptions. Indeed, the two programs might implement significantly different functionalities. Next, we use the fact that a truly random value is, with overwhelming probability, outside the range of a PRG with sufficiently long stretch. Hence, the obfuscations of the above program are computationally indistinguishable from those of the zero program. 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 [55], the range extension of Matsuda and Hanaoka [49] of a multi-bit point function to obtain shorter point values, the range-extension of a UCE1-secure hash function by Bellare, Hoang and Keelveedhi [14], and the negative result of Brzuska and Mittelbach [30] on multi-bit point-function obfuscation with auxiliary inputs.

Assumptions. Garg et al. [39] construct an indistinguishability obfuscator for $\mathcal{NC}^1$ circuits based on intractability assumptions related to multi-linear maps, and show how to bootstrap it to support all polynomial-time circuits via a fully homomorphic encryption scheme with a decryption circuit in $\mathcal{NC}^1$. The authors validate their multi-linear intractability assumption in a generic model of computation. Recent results show how to improve the assumptions used in constructing indistinguishability obfuscators [52,26,4,3,42], further supporting their plausibility.

Indistinguishability obfuscation for Turing machines has been constructed in the works of Boyle, Chung and Pass [25] and Ananth et al. [2]. 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, Chung and Pass [25] and Ananth et al. [2] show how to build iO for *Turing machines* assuming diO for circuits. The plausibility of differing-inputs obfuscation, however, has become somewhat controversial due to a recent result of Garg et al. [40]. These authors show that the existence of a special-purpose obfuscator for a signature scheme implies that diO with arbitrary auxiliary input cannot exist. Although we currently do not know how to build this special-purpose obfuscator, its existence 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. Indeed, very recently and shortly after the appearance of this work, Koppula, Lewko and Waters [47] have succeeded in constructing iO for Turing machines without relying on diO, and using iO for circuits, one-way functions and injective pseudorandom generators.

Deterministic encryption. Our first result establishes the uninstantiability of the Encrypt-with-Hash (EwH) transform of Bellare, Boldyreva and O'Neill [7], whereby one converts a randomized IND-CPA public-key encryption scheme into a deterministic public-key encryption (D-PKE) scheme $\mathsf{D\text{-}PKE}$ by extracting the randomness needed for encryption via hashing the message and the public key, that is, the encryption algorithm $\mathsf{D\text{-}PKE.Enc}^{\mathcal{RO}(\cdot,\cdot)}(m,(\mathsf{hk},pk))$ first computes random coins $r \leftarrow \mathcal{RO}(\mathsf{hk},pk\|m)$ and then invokes the base encryption algorithm on message m, public key pk and random coins r to generate a ciphertext. 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. Standard-model constructions, on the other hand, achieve weaker levels of security, e.g., security against block sources [10,22] or q-bounded adversaries [38,29]. To this end, we ask if any hash function can be used to instantiate the random oracle within the EwH transform. Assuming iO for circuits/Turing machines, we build an IND-CPA secure encryption scheme such that when the EwH transform is applied to this specially devised encryption scheme together with some (b-bounded) hash-function, the resulting scheme is not PRIV-secure, not even for block-sources or 1-bounded PRIV-security.

Starting with an arbitrary scheme PKE we consider a new scheme PKE^* which includes an indistinguishability obfuscation of the following program as part of its ciphertexts.

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

This program performs a check similar to that of the universal BFM circuit, but instead of returning a Boolean value returns the encrypted messages when the check passes. That is, in $\mathsf{P}[pk, m, r]$, the public-key pk, the message m and the randomness r are parameters, and the program takes as input a hash-function H_{hk} (potentially with some hard-coded key hk), evaluates H_{hk} on $pk||m$ to get some value y. Then, it applies PRG to y and checks whether $\mathsf{PRG}(y)$ is equal to $\mathsf{PRG}(r)$. If this is the case, it returns the message m. Else, it returns 0.

We can use an obfuscation of this program to attack the security of $\mathsf{EwH}^{\mathsf{H}}[\mathsf{PKE}^*]$. The second stage of the adversary runs this program on the description H_{hk} of the hash function that is used in the instantiation (with hard-coded hk) to obtain the encrypted message. A corollary of this result is that under iO, no security assumption (single or multi-staged, falsifiable or not) is strong enough to build D-PKEs via EwH. In particular, a new UCE assumption used to instantiate EwH [15] is uninstantiable assuming iO for Turing machines (and b-bounded uninstantiable assuming iO for circuits). We remark that our results are incomparable to those of Wichs [57] who shows an unconditional unprovability result for D-PKEs using *arbitrary* techniques from single-stage assumptions. (Our results are conditional and show uninstantiability of EwH regardless of the assumptions used.) This result naturally extends to the Randomized-Encrypt-with-Hash [9] transform for building hedged PKEs.

The Fujisaki–Okamoto transform. The above result generalizes to a wider class of (possibly randomized) *admissible* transformations that use their underlying PKE schemes in a structured way and admit recovery algorithms that satisfy certain correctness properties. (We leave the details to the main body.) Somewhat surprisingly, the Fujisaki–Okamoto (FO) transform for converting CPA into CCA security is admissible and thus suffers from uninstantiability. The FO transform, which dates back to the 1990s, is a simple and flexible technique to boost security of various schemes and has been widely used in identity-based encryption [24], its hierarchical and fuzzy variants [43,56], forward-secure encryption [35], and certificateless and certificate-based encryption [1,41] to mention a few. Our results, once again, come in two flavors depending on the strength of the underlying obfuscator. Our techniques can be further tweaked to show that one cannot instantiate the oracle used within the asymmetric component of the FO transform. This means that the POWHF-encryption assumption of Boldyreva and Fischlin [23] used for partial instantiation of the oracles in FO is also uninstantiable if iO/iO for Turing machines exists.

Other constructs. The uninstantiability problems arising from the existence of indistinguishability obfuscators are not limited to deterministic encryptions and its generalizations. We revisit the work of Bellare and Keelveedhi (BK) [16] 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 called RHtE to convert authenticated encryption into one which resists key-dependent-message (KDM) attacks. This is done by hashing the key with a random nonce to derive the actual key used in encryption: one encrypts m as $(N, \mathsf{Enc}(\mathsf{H}(\mathsf{hk}, N\|k), m))$ for a random nonce N. Our iO-based uninstantiability result describes an IND-CPA and INT-CTXT-secure authenticated encryption (AE) scheme whose BK transformation is not KDM secure.

Interestingly, BK require the base scheme to meet a stronger security level than IND-CPA: ciphertexts should be indistinguishable from random strings. BK do not consider this difference to be of major importance; in the abstract of their paper they write that they *present a RO-transform RHtE that endows* any *AE-scheme with this security.* Our result brings this stronger requirement to light, and shows that assuming that ciphertexts are pseudorandom might be a way to circumvent uninstantiability as the current state-of-the-art obfuscators produce programs that are structured and do not look random. Conversely, if an indistinguishability obfuscator can produce obfuscations of the zero circuit that look random,[4] then reverting to the stronger security notion would no longer be of any help.

As a final example we show that the Convergent-Encryption transform of Douceur et al. [37] formalized by Bellare, Keelveedhi and Ristenpart (BKR) [17] for building message-locked encryption is also uninstantiable. Once again, BKR formally rely on pseudorandomness of ciphertexts but similar observation to those given above for BK apply here too.

Comparison with CGH. Recall that Canetti, Goldreich and Halevi (CGH) [33] show the uninstantiability of certain ROM digital signature and encryption schemes without relying on iO. Their technique is to give a (contrived) scheme that is secure in the random oracle model but behaves anomalously on certain inputs that are related to a compact description of the hash function. Our uninstantiability results share these features, that is, neither their nor our uninstantiability results apply to "natural" schemes. For instance, it is not known if Encrypt-with-Hash when used with ElGamal is uninstantiable or not. On the other hand, our results apply to natural transformations.

It is natural to ask if CGH-like techniques can be directly applied here so as to obtain uninstantiability results that do not rely on the iO machinery. For uninstantiability 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 Encrypt-with-Hash. For keyed hash functions, on the other hand, there seems to be an inherent limitation to CGH-like techniques. For

[4] Note that generally, obfuscations of circuits cannot look random, because obfuscation maintains functionality and thus, the obfuscations of the zero circuit would be distinguishable from those of the constant one circuit. This trivial attack, however, does not apply here if we require pseudorandomness only for the zero circuit.

instance, the security model for D-PKEs do *not* allow message distributions to depend on the hash key as this value is included in 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*, which seems to be necessary when applying CGH-like techniques. It might appear that this issue can be easily 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. The caveat, however, is that such an uninstantiable scheme no longer falls 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 it only allows black-box access to the hash function and denies encryption access to the hash key.[5] This observation applies to other transformations as well. For instance, in the FO transformation the message that is asymmetrically encrypted is chosen uniformly at random and thus cannot be set to the description of the hash function. To summarize, although the description of the hash function will be eventually made public, the adversarial scheme never gets to the hash function in full and needs to coordinate the attack with the actual adversary, who sees the hash key, to be successful. Indistinguishability obfuscation allows this distributed attack to be carried out.

Concurrent work. In concurrent and independent work, Green et al. [45] use iO and techniques similar to ours to demonstrate the uninstantiability of random-oracle schemes. Like us, they embed an obfuscated program into schemes in order to make them uninstantiable. Our results, however, rule out the instantiability of (existing) random-oracle *transformations* whereas Green at al. construct uninstantiable *schemes* for primitives which cannot be targeted with CGH-like techniques. For instance bit encryption falls outside the reach of CGH as its input space is too short and cannot be made to behave anomalously on special long inputs. Green et al. show that indistinguishability obfuscation can be used to extended CGH to such constrained primitives.

Primitive design. The shortcomings of ROM primitives that we have identified call for a re-assessment of primitives whose security analyses have only been carried out in idealized models of computation. To highlight the importance of this task, we propose a new transform for building deterministic encryption that is specifically designed to bypass our attacks. In this transform one encrypts two values independently across two invocations of the underlying encryption algorithm to make sure that the information needed for the attack is not available to any of the invocations. (This transform, in particular, is not admissible.) We prove this scheme secure in the ROM, but show that the program that one would need to successfully attack the construction (assuming the availability of all needed information) can be *split* into several programs such that by feeding obfuscations of one program into the obfuscations of another an attack can be

[5] Despite this, CGH-like techniques render Encrypt-with-Hash uninstantiable when stronger notions of security are considered [53].

launched. We leave the characterization of the class of transformations which fall prey to extensions of the iO attack as an interesting open problem.

We believe that the structural soundness of ROM schemes should be further validated by studying if attacks similar to those given in this work can be launched against them. To provably rule out such attacks one needs to reduce security to assumptions, which although strong, are not known to be uninstantiable under existence of (d)iO. Candidate examples include UCEs against statistically and/or strongly unpredictable sources [27,31] and indeed indistinguishability obfuscation itself. We note that recently Bellare and Hoang [11] have proposed a D-PKE transform starting from lossy trapdoor function and statistical UCEs. This approach can be further combined with stronger assumptions on the base schemes (such as pseudorandomness of ciphertexts). Indeed, it would be interesting to derive positive results that circumvent iO-based uninstantiability by merely exploiting the pseudorandomness of ciphertexts, even for somewhat artificial tasks. These would strengthen our confidence in applying the random-oracle methodology 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 $x[i]$. We denote the empty string by ε. 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 $x \leftarrow_\$ X$. For a random variable X we denote the support of X by $[X]$. A real-valued function $\nu(\lambda)$ is negligible if $\nu(\lambda) \in \mathcal{O}(\lambda^{-\omega(1)})$. We denote the set of all negligible functions by negl.

An algorithm is a randomized, stateless Turing machine unless otherwise stated. We call an algorithm efficient or PPT if its runtime on any choice of inputs and random coins is at most a polynomial function 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 generated random coins and write $y \leftarrow_\$ \mathcal{A}(x)$. An adversary is a tuple of stateful PPT algorithms. We omit explicit input and output states to ease notations. When an adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ consists of two stages $\mathcal{A}_1$ and $\mathcal{A}_2$, these two stages are assumed to be distinct algorithms that do *not* share any state, unless explicitly permitted to do so by a game.

Turing machines and circuits. Throughout the paper we consider two models of computation: Turing machines and circuits. Recall that a Turing machine can take inputs of arbitrary length whereas the input length to a circuit is fixed. We denote the runtime of a Turing machine M on input x by $\mathsf{time}_{\mathsf{M}}(x)$ and its description size by $|\mathsf{M}|$. We denote the size (a.k.a. runtime) of a circuit C by $|\mathsf{C}|$. 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 on descriptions of circuits C and inputs x for them. Note that UC only accepts inputs (C, x) of a specific total length, whereas UM can take inputs of arbitrary length. In order to simplify the presentation we use the term *program* to refer to either a Turing machine or a circuit. We may, therefore, speak of a universal program UEval, which denotes either a universal Turing machine UM or a universal circuit UC, and evaluates a program P on some input x. When defining a program, we use the notation $\mathsf{P}[z](\cdot)$ to emphasize the fact that the value z is hard-coded into P.

Indistinguishability obfuscation. We define indistinguishability obfuscation for circuits and Turing machines under a single definition. 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. [6,5] as a potential weakening of the virtual-black-box obfuscation property, for which wide infeasibility results are known. Here we give a game-based definition of indistinguishability obfuscation in the style of [19] with extensions to also cover obfuscation for Turing machines [2]. We only consider the setting where both the sampler and distinguisher are uniform but allow the sampler to output inequivalent programs with negligible probability. This game-based formulation is convenient for use in proofs of security.

A PPT algorithm 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 $\mathsf{P} \in \mathcal{P}_\lambda$, and all $\mathsf{P}' \leftarrow_{\$} \mathsf{iO}(1^\lambda, \mathsf{P})$, the programs P and P' are functionally equivalent. That is, $\mathsf{P}(x) = \mathsf{P}'(x)$ for all input values x.
- Succinctness. There is a polynomial poly such that for all $\lambda \in \mathbb{N}$, all $\mathsf{P} \in \mathcal{P}_\lambda$ and all $\mathsf{P}' \leftarrow_{\$} \mathsf{iO}(1^\lambda, \mathsf{P})$ we have that $|\mathsf{P}'| \in \mathcal{O}(\mathsf{poly}(\lambda + |\mathsf{P}|))$.
- Input-specific runtime. There is a polynomial poly such that for all $\lambda \in \mathbb{N}$, all $\mathsf{P} \in \mathcal{P}_\lambda$ and all $\mathsf{P}' \leftarrow_{\$} \mathsf{iO}(1^\lambda, \mathsf{P})$ and all input values x we have that $\mathsf{Time}_{\mathsf{P}'}(x) \in \mathcal{O}(\mathsf{poly}(\lambda + \mathsf{Time}_{\mathsf{P}}(x)))$.
- Security. For any pair of PPT adversaries $(\mathcal{S}, \mathcal{D})$, where $\mathcal{S}$ is an equivalent sampler, i.e., where

$$\mathsf{Adv}^{\mathrm{eq}}_{\mathcal{S}}(\lambda) := \Pr[\exists x \,\mathsf{s.t.}\, \mathsf{P}_0(x) \neq \mathsf{P}_1(x) \vee \mathsf{Time}_{\mathsf{P}_0}(x) \neq \mathsf{Time}_{\mathsf{P}_1}(x) : \\ (\mathsf{P}_0, \mathsf{P}_1, aux) \leftarrow_{\$} \mathcal{S}(1^\lambda)]$$

is negligible, we have that

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

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

When working with circuits, succinctness and runtime requirements are redundant and follow from the facts that iO is polynomial time and that the size and runtime of a circuit are identical.

Garg et al. [39] prove that under intractability assumptions related to multilinear maps an indistinguishability obfuscator supporting all $\mathcal{NC}^1$ circuits exists. Assuming the existence of a perfectly correct, leveled fully homomorphic encryption scheme and a perfectly sound non-interactive witness-indistinguishable proof system, they also show how to extended this to support all polynomial-size circuits, i.e., the family $\mathcal{C} := \{\mathcal{C}_{\mathrm{b}(\lambda)}\}_{\lambda\in\mathbb{N}}$ where b is a polynomial and

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

Several follow-up works improved the assumptions underlying indistinguishability obfuscators as well as the performance [52,26,3,4,42]. As mentioned above, circuits and obfuscations thereof admit fixed-length inputs only.

Remark. 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. For any bound b, existence of iO for $\mathcal{C}_{\mathrm{b}(\lambda)}$ under our definition is implied by the (non-uniform) definition of Garg et al. [39].

Ananth et al. [2] and Boyle et al. [25] give constructions of indistinguishability obfuscators for Turing machines which admit inputs of arbitrary lengths. Their constructions achieve the stronger notion of *differing-inputs* (a.k.a. extractability) obfuscation, initially also suggested in the work of Barak et al. [6,5]. This type of obfuscation can be regarded as a generalization of indistinguishability obfuscation to programs which are not necessarily functionally equivalent. We recall [2, Theorem 3] and refer the reader to the original works for details and discussion.

Theorem 1 (Ananth et al. [2]). *Under the existence of CPA-secure leveled fully homomorphic encryption, succinct non-interactive arguments of knowledge (SNARKs), differing-inputs obfuscation for all circuits in* $\mathcal{P}/\mathrm{poly}$*, 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*

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

Koppula, Lewko and Waters [47] have succeeded in constructing iO for Turing machines without relying on diO, and using iO for circuits, one-way functions and injective pseudorandom generators.

```
$\mathrm{IO}_{\mathsf{iO}}^{\mathcal{S},\mathcal{D}}(\lambda)$
$(\mathsf{P}_0, \mathsf{P}_1, aux) \leftarrow_{\$} \mathcal{S}(1^\lambda)$
$b \leftarrow_{\$} \{0,1\}$
$\mathsf{P}' \leftarrow_{\$} \mathsf{iO}(1^\lambda, \mathsf{P}_b)$
$b' \leftarrow_{\$} \mathcal{D}(\mathsf{P}', aux)$
$\mathbf{return}\ (b = b')$
```

```
$\mathrm{IND\text{-}CPA}_{\mathsf{PKE}}^{\mathcal{A}}(\lambda)$
$(sk, pk) \leftarrow_{\$} \mathsf{PKE.Kg}(1^\lambda)$
$(m_0, m_1) \leftarrow_{\$} \mathcal{A}(pk)$
$b \leftarrow_{\$} \{0,1\}$
$c \leftarrow_{\$} \mathsf{PKE.Enc}(pk, m_b)$
$b' \leftarrow_{\$} \mathcal{A}(c)$
$\mathbf{return}\ (b = b')$
```

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

Fig. 1. Left: IO game defining the security of an indistinguishability obfuscator. **Middle**: The IND-CPA game for a public-key encryption scheme. **Right**: The IND security game for deterministic PKEs.

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 (sk, pk). 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 $\mathsf{PKE.Dec}(sk, c)$ is given a ciphertext c and secret key sk and outputs a plaintext m or a special symbol $\perp$. We denote the supported message length by $\mathsf{PKE.il}(\lambda)$ and the maximum length of random strings used to encrypt a $\mathsf{PKE.il}(\lambda)$-bit message 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 $\mathcal{A}$ in the IND-CPA game (shown in Figure 1; center) defined by

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

is negligible.

Function families. Following [19], 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 $\mathsf{FF.kl}$, $\mathsf{FF.il}$, and $\mathsf{FF.ol}$ are deterministic and on input 1^λ specify the key, input, and output lengths, respectively. The key-generation algorithm $\mathsf{FF.Kg}$ 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 $\mathsf{FF.Ev}$ 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 $\mathsf{FF.Ev}(1^\lambda, \mathsf{fk}, x) \in \{0,1\}^{\mathsf{FF.ol}(\lambda)}$. We will often refer to function families as hash functions in this work.

PRFs and PRGs. We say that a function family FF is pseudorandom if for any PPT adversary $\mathcal{A}$ we have that

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

In the first term above, 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 $\mathcal{RO}$ with domain $\{0,1\}^{\mathsf{FF.il}}(\lambda)$ and range $\{0,1\}^{\mathsf{FF.ol}(\lambda)}$.

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

$$\begin{aligned}\mathsf{Adv}^{\mathrm{prg}}_{\mathsf{PRG},\mathcal{A}}(\lambda) := \Pr[\mathcal{A}(1^{\lambda}, \mathsf{PRG}(s)) = 1 : s \leftarrow_{\$} \{0,1\}^{\mathsf{PRG.il}(\lambda)}]& \\ - \Pr[\mathcal{A}(1^{\lambda}, y) = 1 : y \leftarrow_{\$} \{0,1\}^{\mathsf{PRG.ol}(\lambda)}]&\end{aligned}$$

is negligible.

Keyed random oracles. Most random-oracle transformations and schemes in the literature are analyzed in the “unkeyed” random-oracle model, and this reflects the fact that a fixed unkeyed hash function will be used in their instantiations. Keyed hash functions, however, are more powerful when it comes to instantiating random oracles and this leaves 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 is rather unclear. For example, if we consider a transformation of symmetric encryption schemes, the hash key could be part of the key-generation process in which case it remains hidden from the adversary, or it could be a parameter generated during set-up, in which case it would be available to the adversary. We therefore use a generalization of the standard random-oracle model whereby all parties get access to a *keyed* random function. More precisely, in the $(\mathsf{kl}, \mathsf{il}, \mathsf{ol})$-ROM, where $(\mathsf{kl}, \mathsf{il}, \mathsf{ol})$ specify various lengths as before, on security parameter λ all parties get access to a random function of the form

$$\mathcal{RO}(\cdot,\cdot) : \{0,1\}^{\mathsf{kl}(\lambda)} \times \{0,1\}^{\mathsf{il}(\lambda)} \longrightarrow \{0,1\}^{\mathsf{ol}(\lambda)} .$$

Note that we recover the standard unkeyed random-oracle model when $\mathsf{kl}(\lambda) = 0$ (there is only one key ε, the empty string). In defining the security of a cryptosystem, the underlying probability space is extended to include a random choice of a keyed function (and choices of random key as specified by the scheme). Whether or not a party gets to see the hash key depends on the specification of the scheme and its security model. For instance, if a keyed ROM scheme includes hash keys under its public keys, an honest or malicious party gets to sees the hash key whenever it gets to see the public key. As our result is a negative result, it suffices to consider weak adversaries that do not get oracle access and/or the hash key in some of their stages, because weaker adversaries correspond to a stronger negative result.

(Un)instantiability. Given a scheme in the keyed ROM, we consider its standard-model instantiations via (concrete) keyed hash functions. Formally, this entails: (1) using a hash function that has key, input and output lengths that are identical to those of the keyed random oracle, (2) running the key-generation algorithm whenever a hash key is generated in the ideal scheme, and (3) calling the evaluation

routine of the hash function whenever an oracle query is placed. Given a keyed ROM scheme and a security model for it, we say that the scheme is *instantiable* if there exists a hash function which when used to instantiate the scheme (and its security model) results in a secure scheme (with respect to the instantiated security model). Conversely, we say that a scheme is *(strongly) uninstantiable* if no hash function can securely instantiate the ideal scheme. Finally, for a polynomial bound p, we call a scheme b-*uninstantiable*, if no hash function of size at most $\mathrm{b}(\lambda)$ can securely instantiate the scheme.

3 Deterministic Encryption

We start by studying the Encrypt-with-Hash (EwH) transform of Bellare, Boldyreva and O'Neill (BBO) [7] for building deterministic encryption from standard (randomized) encryption schemes. We 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. We then prove uninstantiability, and end with two corollaries of this result.

3.1 Definitions

Deterministic public-key encryption. Deterministic public-key encryption was first introduced by Bellare, Boldyreva and O'Neill [7]. The syntax and correctness of a deterministic public-key encryption (D-PKE) 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 deterministic (i.e., $\mathsf{D\text{-}PKE.rl}(\lambda) = 0$). BBO [7] model the security of D-PKEs via a form of simulation-based notion called PRIV. In later works, Bellare et al. [10] and independently Boldyreva, Fehr and O'Neill [22] introduce an indistinguishability-based notion called IND and show that it implies is equivalent to PRIV security. The IND game is formally defined in Figure 1 on the right.[6] 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 dimension that have distinct components and component-wise contain messages of the same length. (Adversary $\mathcal{A}_1$ does not get to see the public key.) 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 $i \in [|\mathbf{m}_b|]$,

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

A key pair $(pk, sk) \leftarrow_{\$} \mathsf{D\text{-}PKE.Kg}(1^\lambda)$ is then chosen, and according to the challenge bit b, one of the two message vectors is encrypted component-wise. The

[6] Bellare et al. [10] allow an additional zeroth-stage adversary to output shared state for adversaries $\mathcal{A}_1$ and $\mathcal{A}_2$. As we prove an impossibility result we choose the weaker definition where this shared state is empty.

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 $\mathsf{D\text{-}PKE}$ by

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

We say that scheme $\mathsf{D\text{-}PKE}$ is IND secure if the advantage of any PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ in the IND game is negligible. The 1-bounded version of this security model demands that the two vectors $(\mathbf{m}_0, \mathbf{m}_1)$ only contain a single message each.

The Encrypt-with-Hash transform. The Encrypt-with-Hash (EwH) transform constructs a deterministic public-key encryption scheme from a (randomized) public-key encryption scheme PKE in the random-oracle model [7]. We present this transform in the keyed ROM, and note that it matches the original transform for singleton key spaces. The keyed RO is assumed to have a range which matches the randomness space of the PKE scheme and a domain which consisting of all bit strings of length the maximum length of public keys plus the length of messages. The EwH transform operates as follows.

The key-generation generates a key pair using the key-generation algorithm of the base PKE scheme. It also generates a hash key $\mathsf{hk} \leftarrow_{\$} \{0,1\}^{\mathsf{kl}(\lambda)}$ and returns $(sk, (\mathsf{hk}, pk))$. Algorithm $\mathsf{D\text{-}PKE.Enc}^{\mathcal{RO}(\cdot,\cdot)}(m, (\mathsf{hk}, pk))$ first computes random coins $r \leftarrow \mathcal{RO}(\mathsf{hk}, pk\|m)$ and then invokes the base encryption algorithm on m and pk and coins r to generate a ciphertext. The decryption routine is identical to that of the underlying scheme (plus a ciphertext re-computation check to ensure non-malleability). EwH results in an IND-secure D-PKE scheme in the keyed ROM when starting from an IND-CPA public-key encryption scheme.

Key access in EwH. With the formalism introduced above, both adversaries $\mathcal{A}_1$ and $\mathcal{A}_2$ get oracle access to $\mathcal{RO}(\cdot,\cdot)$. The first-stage adversary, however, does *not* get to see hk since the hash key is distributed as a component of the public keys. The second-stage adversary, on the other hand, does get to see it. A stronger model where the hash key *is* given out in the first stage can be considered. EwH meets this stronger notion of security, but since our results are negative we use the conventional (and weaker) IND model.

3.2 Uninstantiability of EwH

When the EwH transformation is instantiated with an *unkeyed* random oracle a CGH-style uninstantiability result can be directly established [33]. (This in particular shows that the use of a keyed hash function is necessary to instantiate EwH.) Given an arbitrary PKE scheme PKE, consider a tweaked variant of it PKE' which first interprets parts of the message m as the description of a hash function H (together with its single key) and checks if the provided random coins r match the hash value $\mathsf{H}(pk\|m)$. If so, it returns $0\|m$ and else it returns $1\|\mathsf{PKE.Enc}(pk, m; r)$. Scheme PKE' is still IND-CPA secure because the probability that a truly random value r matches $\mathsf{H}(pk\|m)$ is negligible. On the other hand, when the random coins are generated deterministically by applying a hash function, an IND adversary which asks for encryptions of $m_i\|\mathsf{H}$ for any two high min-entropy messages m_0 and m_1 which differ, say, on their most significant bits can easily win the game.[7] The standard IND game, however, restricts the first-stage adversary not to learn the public key, and thus, it cannot guess the (high min-entropy) hash key.

We show how to use indistinguishability obfuscation to extend the above uninstantiability 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 is assumed to support. Assuming iO for Turing machines we obtain a strong uninstantiability result: there exists an IND-CPA encryption scheme that cannot be securely used in EwH in conjunction with *any* keyed hash function. Assuming the weaker notion of iO for circuits, we get b-uninstantiability: for any polynomial bound b there exists an IND-CPA scheme that cannot be securely used in EwH for any hash function whose description size is at most b. The latter result is still quite strong as, in particular, it means that for any finite set of hash functions (e.g., those which are standardized), we can give a PKE scheme that when used within EwH yields an insecure D-PKE scheme for any choice of hash function from the set. We note that the adversarial PKE scheme that we construct depends only on an upper bound on description sizes and not on their implementation details.

Theorem 2 (Uninstantiability of EwH). *Assuming the existence of indistinguishability obfuscation for Turing machines $\mathcal{M}$ (resp.* b*-bounded circuits $\mathcal{C}_{\mathrm{b}}$), the* EwH *transform is uninstantiable (resp.* b*-uninstantiable) with respect to* IND *security in the standard model.*

We start by giving a high-level description of the proof before presenting the details. We may assume, without loss of generality, that an IND-CPA-secure PKE scheme exists as otherwise uninstantiability trivially holds. This, in turn, implies that we can also assume the existence of a secure pseudorandom generator.

Now given an IND-CPA-secure PKE scheme PKE, we construct a tweaked scheme PKE^* that is also IND-CPA secure but the D-PKE scheme $\mathsf{EwH}^{\mathsf{H}}[\mathsf{PKE}^*]$ fails to be IND secure.

To construct the adversarial scheme PKE^* we follow a similar strategy to CGH. The fundamental difference here is that $\mathsf{PKE}^*.\mathsf{Enc}$ does not have access to the hash key. To overcome this problem, we consider the obfuscation of a program P' that implements a *universal* variant of the BFM circuit [27], i.e., it takes as input the description of a hash function $\mathsf{H}(\mathsf{hk},\cdot)$, with a hard-wired key, runs it on two values m and pk embedded into P', and outputs m if the result matches a third hard-wired value r:

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

[7] This attack generalizes to the setting where the first-stage adversary can guess the hash key with non-negligible probability and in particular, EwH is uninstantiable with respect to the stronger IND model discussed above.

The tweak that we introduce in PKE^* is that the encryption operation appends obfuscations of $\mathsf{P}'[pk, m, r]$ to its ciphertexts, where pk, m and r are the values input to the encryption routine.

We need to argue (1) that this tweak allows an adversary to break the scheme whenever the hash function is instantiated and (2) that outputting such an indistinguishability obfuscation of P' does not hurt the IND-CPA security of PKE^*.

For (1), note that given an obfuscation of $\mathsf{P}'[pk, m, r]$ as well as a description of $\mathsf{H}(\mathsf{hk}, \cdot)$, an adversary can recover m by running the above circuit on $\mathsf{H}(\mathsf{hk}, \cdot)$. Now the second stage of the IND adversary gets the public key and thus the description of the hash-function $\mathsf{H}(\mathsf{hk}, \cdot)$. Furthermore, it also gets a ciphertext which contains an obfuscation of $\mathsf{P}'[pk, m, r]$. Hence, the second-stage adversary has all the information needed to break the IND security of the deterministic encryption scheme $\mathsf{EwH}^{\mathsf{H}}[\mathsf{PKE}^*]$.

Now, intuitively, this insecurity might have nothing to do with the transform because the tweaked scheme PKE^* is already insecure anyway. Hence, we also need to argue that PKE^*, as a randomized encryption scheme, is IND-CPA secure. Following BFM, we try to prove this by showing that the obfuscated circuit is functionally equivalent to the *zero* circuit and hence it does not leak any information about m.

We would like to argue that for a *truly random* r—such an r is used in randomized encryption—P' implements the constant zero program Z. Indeed, if r is sufficiently longer than $|pk| + |m|$ then for any fixed $\mathsf{H}(\mathsf{hk}, \cdot)$, over a random choice of r the check performed by P' would fail with all but negligible probability. This, however, does not necessarily mean that the circuit is functionally equivalent to Z 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 this event 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 consider a scheme which has a much smaller randomness space and instead uses coins that are *pseudorandomly generated.* This ensures that the randomness space used by PKE is sparse within the set of all possible coins, allowing a counting argument to go through. We adapt the program above to cater for the new tweaks:

$$\begin{aligned}\mathsf{P}[pk, m, \mathsf{PRG}(r)]\Big(\mathsf{H}(\mathsf{hk}, \cdot)\Big) := &\text{ if } \mathsf{PRG}(\mathsf{H}(\mathsf{hk}, pk\|m)) = \mathsf{PRG}(r)\\ &\qquad\text{return } m\\ &\quad\text{else return } 0\ .\end{aligned}$$

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 truly random $s \in \{0,1\}^{\mathsf{PRG.ol}(\lambda)}$ the program $\mathsf{P}[pk, m, s]$ has a *description* which is indistinguishable from that of $\mathsf{P}[pk, m, \mathsf{PRG}(r)]$ down to the security of PRG. Furthermore for such an s, this program *can* be shown to be functionally equivalent to the zero circuit with overwhelming probability as s will be outside the range of the PRG with overwhelming probability. These two steps allow us to prove that obfuscations of P leak no information about m, and show that scheme PKE^* is IND-CPA secure.

Finally, notice that obfuscations of P (similarly to those of P') allow an IND adversary to break the resulting EwH-transformed scheme: simply run the obfuscation of P on the description of the hash function used in the instantiation (with a hard-wired key) to recover the encrypted message.

Not that formally program P will use a universal program evaluator to run its input hash-function descriptions. If the (obfuscated) program is a Turing machine, it can be run on arbitrary large descriptions and arbitrarily sized hash functions are ruled out. On the other hand, if the program is a circuit, it has an a priori *fixed* input length, and thus can only be run on hash functions that respect the input-size restrictions. We next formalize this proof intuition.

Proof (of Theorem 2). 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 supporting either Turing machines or circuits. We define a modified PKE scheme PKE^* as follows. The key-generation algorithm is unchanged. The adapted encryption algorithm is defined as shown below by appending an obfuscated program $\overline{\mathsf{P}}$ to its outputs. UEval denotes a universal program evaluator. The modified decryption algorithm ignores the $\overline{\mathsf{P}}$ component and decrypts as in the base scheme.

```
ALGO. $\mathsf{PKE}^*.\mathsf{Enc}(pk, m; r\|r')$
$s \leftarrow \mathsf{PRG}(r)$
$c \leftarrow \mathsf{PKE.Enc}(pk, m; s)$
$\overline{\mathsf{P}} \leftarrow \mathsf{iO}(\mathsf{P}[pk, m, s](\cdot); r')$
return $(c, \overline{\mathsf{P}})$
```

```
PROG. $\mathsf{P}[pk, m, s](\mathsf{H}(\mathsf{hk}, \cdot))$
$r\|r' \leftarrow \mathsf{UEval}(\mathsf{H}(\mathsf{hk}, \cdot), pk\|m)$
$s' \leftarrow \mathsf{PRG}(r)$
if $(s' = s)$ then return $m$
return $0$
```

When we consider the above construction with respect to circuits, we need to specify an extra parameter b that upper-bounds the size of the inputs to the universal circuit evaluator. This maximum size of programs that the universal circuit admits corresponds to the maximum size of the hash functions that our uninstantiability proof applies to. 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 IND-CPA game for the randomized base scheme PKE^* and an arbitrary adversary $\mathcal{A}$.

Game_1**:** In this game the randomness s used in encryption is no longer generated via a PRG call and is sampled uniformly at random.

Game_2**:** In this game the ciphertext component $\overline{\mathsf{P}}$ is generated as an indistinguishability obfuscation of the zero program (that is, Turing machine or circuit) Z padded to the appropriate length (and running time).

$\mathsf{Game}_0(\lambda) \xrightarrow{\mathsf{PRG}}$	$\mathsf{Game}_1(\lambda) \xrightarrow{\mathsf{iO}}$	$\mathsf{Game}_2(\lambda)$
$b \leftarrow_{\$} \{0,1\}$	$b \leftarrow_{\$} \{0,1\}$	$b \leftarrow_{\$} \{0,1\}$
$(sk, pk) \leftarrow_{\$} \mathsf{PKE.Kg}(1^{\lambda})$	$(sk, pk) \leftarrow_{\$} \mathsf{PKE.Kg}(1^{\lambda})$	$(sk, pk) \leftarrow_{\$} \mathsf{PKE.Kg}(1^{\lambda})$
$(m_0, m_1) \leftarrow_{\$} \mathcal{A}(pk)$	$(m_0, m_1) \leftarrow_{\$} \mathcal{A}(pk)$	$(m_0, m_1) \leftarrow_{\$} \mathcal{A}(pk)$
$r \\| r' \leftarrow_{\$} \{0,1\}^{\mathsf{PKE.rl}(\lambda)}$	$r \\| r' \leftarrow_{\$} \{0,1\}^{\mathsf{PKE.rl}(\lambda)}$	$r \\| r' \leftarrow_{\$} \{0,1\}^{\mathsf{PKE.rl}(\lambda)}$
$s \leftarrow \mathsf{PRG}(r)$	$s \leftarrow \{0,1\}^{\mathsf{PRG.ol}(\lambda)}$	$s \leftarrow \{0,1\}^{\mathsf{PRG.ol}(\lambda)}$
$c \leftarrow \mathsf{PKE}(pk, m_b; s)$	$c \leftarrow \mathsf{PKE}(pk, m_b; s)$	$c \leftarrow \mathsf{PKE}(pk, m_b; s)$
$\overline{\mathsf{P}} \leftarrow \mathsf{iO}(\mathsf{P}[pk, m_b, s]; r')$	$\overline{\mathsf{P}} \leftarrow \mathsf{iO}(\mathsf{P}[pk, m_b, s]; r')$	$\overline{\mathsf{P}} \leftarrow \mathsf{iO}(\mathsf{Z}_{\|\mathsf{P}[pk, m_b, s]\|}; r')$
$b' \leftarrow_{\$} \mathcal{A}(c, \overline{\mathsf{P}})$	$b' \leftarrow_{\$} \mathcal{A}(c, \overline{\mathsf{P}})$	$b' \leftarrow_{\$} \mathcal{A}(c, \overline{\mathsf{P}})$
return $(b' = b)$	**return** $(b' = b)$	**return** $(b' = b)$

Fig. 2. Hybrids used in the proof of Theorem 2. The highlighted lines show the changes in game transitions.

We now show that each of the above transitions negligibly changes the game's output with respect to any adversary $\mathcal{A}$.

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 y, a PRG image under a uniformly random seed or a truly uniformly random value, can perfectly simulate games Game_0 and Game_1 for $\mathcal{A}$ by using y in place of s. If y is a PRG image, then Game_0 is run and if y is uniformly random the Game_1 is run:

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

Game_1 *to* Game_2*.* We show that this hop negligibly affects the winning probability of $\mathcal{A}$ down to the security of the indistinguishability obfuscator. We let $\mathcal{S}$ to 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 then sets $\mathsf{P}_0 := \mathsf{P}[pk, m_b, s]$, $\mathsf{P}_1 := \mathsf{Z}_{|\mathsf{P}_0|}$ and aux to be the ciphertext component c and the internal state of the first phase of the IND-CPA adversary. Algorithm $\mathcal{D}$ receives an obfuscation$\overline{\mathsf{P}}$ of either P_0 or P_1, and resumes the second phase of $\mathcal{A}$ on $(c, \overline{\mathsf{P}})$ using the state recovered from aux. When P_0 is obfuscated $\mathcal{A}$ is run according to the rules of Game_1 and when P_1 is obfuscated $\mathcal{A}$ is run according to the rules of Game_2. Hence,

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

We must show that the sampler $\mathcal{S}$ constructed above outputs functionally equivalent circuits with overwhelming probability. Assuming that the stretch of the PRG is 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 functionally inequivalent to the zero circuit is upper bounded by $2^{-\mathsf{PRG.il}(\lambda)}$, that is,

$$\Pr\left[\exists x\, \mathsf{P}_0(x) \neq 0 : (\mathsf{P}_0, \mathsf{P}_1, aux) \leftarrow_{\$} \mathcal{S}(1^\lambda)\right] \leq 2^{-\mathsf{PRG.il}(\lambda)} \ .$$

When working with Turing machines, we also need to ensure that the two programs used above respect the run-time requirements of the definition of a secure indistinguishability obfuscator for Turning machines. Formally, we will implement the Turing machines P and Z *obliviously* as follows. We first consider an oblivious Turing machine which takes in the description of the hash function *and a message* as input and performs exactly the same computation that P does. We then implement P by fixing the message input of this machine to that passed to the encryption algorithm, retaining the machine's oblivious structure. The same strategy will be used in constructing the zero circuit, where the constant zero message (of correct length) is hard-wired in. Since these machines are oblivious, their runtimes depend only on the *sizes* of the message and the hash description and hence coincide.

Game_2*.* We reduce the advantage of $\mathcal{A}$ in Game_2 to the IND-CPA security of scheme PKE. The only difference between this game and the usual IND-CPA game for PKE is that an obfuscation of $\mathsf{Z}_{|\mathsf{P}[pk,m_b,s]|}$ is attached to the ciphertexts. This program has a public description and hence its obfuscations can be perfectly simulated. Hence,

$$2 \cdot \Pr[\,\mathsf{Game}_2(\lambda)] - 1 \leq \mathsf{Adv}^{\text{ind-cpa}}_{\mathsf{PKE}^*,\mathcal{A}}(\lambda) \ .$$

The attack. To conclude the proof, we show there exists an adversary $(\mathcal{A}_1, \mathcal{A}_2)$ that breaks the IND security of $\mathsf{EwH}^{\mathsf{H}}[\mathsf{PKE}^*]$ for any function H that respects the input requirements of P (arbitrary if P is a Turing machine, and b-bounded if a circuit). 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 entropy requirements of admissible IND adversaries. Adversary $\mathcal{A}_2$ gets as input the public key (pk, hk) and a ciphertext $(c, \overline{\mathsf{P}})$. It then evaluates $\overline{\mathsf{P}}$ on the description of hash function $\mathsf{H}(\mathsf{hk}, \cdot)$ with key hk recovered from the public key and hard-coded into the program description. (Note that if we are considering circuits, the description of this circuit must have size at most $\mathrm{b}(\lambda)$.) Adversary $\mathcal{A}_2$ returns the least significant bit of P's output. This adversary and its operation within the IND game is shown in Figure 3. By the correctness of the obfuscator, $(\mathcal{A}_1, \mathcal{A}_2)$ always win IND with probability 1 irrespectively of the message that is encrypted:

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

□

3.3 Consequences for UCEs

We turn to Universal Computational Extractors (UCEs), a novel notion introduced by Bellare, Hoang and Keelveedhi (BHK) [12] to generically instantiate

```
$\mathrm{IND}^{\mathcal{A}_1,\mathcal{A}_2}_{\mathsf{EwH^H[PKE^*]}}(\lambda)$
$b \leftarrow_\$ \{0,1\}$
$(m_0, m_1) \leftarrow_\$$ $\mathcal{A}_1(1^\lambda)$
    $x_0 \leftarrow_\$ \{0,1\}^{\mathsf{PKE.il}(\lambda)-1}$
    $x_1 \leftarrow_\$ \{0,1\}^{\mathsf{PKE.il}(\lambda)-1}$
    return $(x_0\|0, x_1\|1)$
$(sk, pk) \leftarrow_\$ \mathsf{D\text{-}PKE.Kg}(1^\lambda)$
$\mathsf{hk} \leftarrow_\$ \mathsf{HKg}(1^\lambda)$
$(c, \overline{\mathsf{P}}) \leftarrow$ $\mathsf{EwH^H[PKE^*].Enc}((pk, \mathsf{hk}), m_b)$
    $r\|r' \leftarrow \mathsf{H.Ev}(\mathsf{hk}, pk\|m_b)$
    $(c, \overline{\mathsf{P}}) \leftarrow$ $\mathsf{PKE^*.Enc}((pk, \mathsf{hk}), m_b; r\|r')$
        $s \leftarrow \mathsf{PRG}(r)$
        $c \leftarrow \mathsf{PKE.Enc}(pk, m_b; s)$
        $\overline{\mathsf{P}} \leftarrow \mathsf{iO}(\mathsf{P}[pk, m_b, s](\cdot); r')$
        return $(c, \overline{\mathsf{P}})$
    return $(c, \overline{\mathsf{P}})$
$b' \leftarrow$ $\mathcal{A}_2(1^\lambda, (pk, \mathsf{hk}), (c, \overline{\mathsf{P}}))$
    $m_b \leftarrow \overline{\mathsf{P}}(\mathsf{H}(\mathsf{hk}, \cdot))$
    $b' \leftarrow m_b[|m_b|]$
    return $b'$
return $(b = b')$
```

Fig. 3. The IND-security game for scheme $\mathsf{EwH^H[PKE^*]}$ with our adversary $(\mathcal{A}_1, \mathcal{A}_2)$ as constructed in the proof of Theorem 2. The boxed algorithms are to be understood as subroutines.

random oracles across a number of cryptographic protocols. UCEs constitute a set of assumptions that roughly speaking model the strong extractor properties enjoyed by (keyed) random oracles. One application of this new framework has been to the EwH transform. BHK [15] show that if a scheme PKE is IND-CPA secure and a hash function H meets what they call $\mathrm{UCE}[\mathcal{S}^{\mathrm{cup}} \cap \mathcal{S}_{\mathsf{PKE}}]$ security then $\mathsf{EwH^H[PKE]}$ is IND secure. (We refer the reader to the May 2014 version of the paper for the details.) We emphasize that this security definition *depends* on the PKE scheme, because the source class $\mathcal{S}_{\mathsf{PKE}}$ is restricted to those which run the PKE scheme as a subroutine. Our negative results on EwH show that $\mathrm{UCE}[\mathcal{S}^{\mathrm{cup}} \cap \mathcal{S}_{\mathsf{PKE}}]$ security is uninstantiable.

Corollary 1 **($\mathrm{UCE}[\mathcal{S}^{\mathrm{cup}} \cap \mathcal{S}_{\mathsf{PKE}}]$ uninstantiability).** *Assuming the existence of indistinguishability obfuscation for Turing machines $\mathcal{M}$ (resp.* b*-bounded circuits $\mathcal{C}_{\mathrm{b}}$),* $\mathrm{UCE}[\mathcal{S}^{\mathrm{cup}} \cap \mathcal{S}_{\mathsf{PKE}}]$ *security for hash functions is uninstantiable (resp.* b*-uninstantiable) in the standard model.*

We remark that BHK based the security of EwH on other stronger UCE assumptions [12,13]. Our results also show the uninstantiability of these notions assuming indistinguishability obfuscation and in particular imply the negative results of [27]. In particular, we can rule out the instantiatiability of the so-called *bounded paralell sources*[13] by considering sources that internally run an obfuscator. (This translates to D-PKE schemes which run an obfuscator in their encryption routine as we constrcut above.) The results of BFM [27], however, rule out a wider choice of parameters for bounded paralell sources.

3.4 Extension to hedged PKEs

Hedged public-key encryption, introduced by Bellare et al. [9] models the security of public-key encryption schemes where the random coins used in encryption might have low entropy. Indistinguishability under chosen-distribution attacks (IND-CDA) shown in Figure 4 formalizes the security of hedged PKEs. This notion is similar to IND and the only difference is that the adversary additionally to the two message vectors also outputs a randomness vector. 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, called Randomized Encrypt-with-Hash, can be defined for hedged PKEs [9]: hash the message, public key and the randomness to obtain new coins, and use them in encryption. Our uninstantiability result can be immediately adapted to this transform as long as the message space has super-polynomial size:

```
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 input 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 not violate the min-entropy requirements as long as the min-entropy of the messages is sufficiently high. We note that if one strengthens the IND-CDA notion to require the randomness distribution to have super-logarithmic min entropy, our attacks would no longer work. This in particular is the case if the message space of the scheme is small.

3.5 Other uninstantiability results

In the full version of the paper [28] we show that our uninstantiability results can be further leveraged to rule out standard-model instantiations of a number of other known transformations. We generalize the iO attack to what we call *admissible transformations*, and show that the classical and widely deployed

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

Fig. 4. The IND-CDA security game for hedged public-key encryption 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 [54].

Fujisaki–Okamoto transformation [FO99] falls under it. We also show that a generic approach to building secure symmetric encryption in the presence of key-dependent messages, and another one for building de-duplication schemes are uninstantiable.

In the full version, we also explore new classes of D-PKE transformations that lie beyond those captured by admissible transformations. We present a candidate transformation that is specifically designed to foil our iO attack. We first show that this transformation is structurally sound by proving it secure in the ROM. We then show how to extend our techniques to this (and potentially other classes of) transformations. Our goal is to illustrate the flexibility of our main technique and show that it can be tweaked and extended in many ways.

4 Concluding Remarks

The uninstantiability results presented in this paper (and the generalization presented in the full version [28]) demonstrate 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. It is also an interesting and nontrivial 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) [15]. For instance, Bellare, Hoang and Keelveedhi [15] 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 [27]. Very recently, Bellare and Hoang [11] have proposed a similar transform for D-PKE starting from lossy trapdoor functions.

Alternatively, one could switch to schemes that meet stronger notions of security. For instance, IND$-type security notions that require the ciphertexts to be indistinguishable from random do not lend themselves to out attacks as it is unclear if obfuscation schemes can provide circuits which are indistinguishable from random strings.

Acknowledgments

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

References

1. Al-Riyami, S.S., Paterson, K.G.: Certificateless public key cryptography. In: Laih, C.S. (ed.) Advances in Cryptology – ASIACRYPT 2003. Lecture Notes in Computer Science, vol. 2894, pp. 452–473. Springer, Berlin, Germany, Taipei, Taiwan (Nov 30 – Dec 4, 2003)
2. Ananth, P., Boneh, D., Garg, S., Sahai, A., Zhandry, M.: Differing-inputs obfuscation and applications. Cryptology ePrint Archive, Report 2013/689 (2013), http://eprint.iacr.org/2013/689
3. Ananth, P.V., Gupta, D., Ishai, Y., Sahai, A.: Optimizing obfuscation: Avoiding barrington's theorem. In: Ahn, G.J., Yung, M., Li, N. (eds.) ACM CCS 14: 21st Conference on Computer and Communications Security. pp. 646–658. ACM Press, Scottsdale, AZ, USA (Nov 3–7, 2014)
4. Barak, B., Garg, S., Kalai, Y.T., Paneth, O., Sahai, A.: Protecting obfuscation against algebraic attacks. In: Nguyen, P.Q., Oswald, E. (eds.) Advances in Cryptology – EUROCRYPT 2014. Lecture Notes in Computer Science, vol. 8441, pp. 221–238. Springer, Berlin, Germany, Copenhagen, Denmark (May 11–15, 2014)
5. Barak, B., Goldreich, O., Impagliazzo, R., Rudich, S., Sahai, A., Vadhan, S., Yang, K.: On the (im)possibility of obfuscating programs. J. ACM 59(2), 6:1–6:48 (May 2012), http://doi.acm.org/10.1145/2160158.2160159
6. Barak, B., Goldreich, O., Impagliazzo, R., Rudich, S., Sahai, A., Vadhan, S.P., Yang, K.: On the (im)possibility of obfuscating programs. In: Kilian, J. (ed.) Advances in Cryptology – CRYPTO 2001. Lecture Notes in Computer Science, vol. 2139, pp. 1–18. Springer, Berlin, Germany, Santa Barbara, CA, USA (Aug 19–23, 2001)
7. Bellare, M., Boldyreva, A., O'Neill, A.: Deterministic and efficiently searchable encryption. In: Menezes, A. (ed.) Advances in Cryptology – CRYPTO 2007. Lecture Notes in Computer Science, vol. 4622, pp. 535–552. Springer, Berlin, Germany, Santa Barbara, CA, USA (Aug 19–23, 2007)
8. Bellare, M., Boldyreva, A., Palacio, A.: An uninstantiable random-oracle-model scheme for a hybrid-encryption problem. In: Cachin, C., Camenisch, J. (eds.) Advances in Cryptology – EUROCRYPT 2004. Lecture Notes in Computer Science, vol. 3027, pp. 171–188. Springer, Berlin, Germany, Interlaken, Switzerland (May 2–6, 2004)

9. Bellare, M., Brakerski, Z., Naor, M., Ristenpart, T., Segev, G., Shacham, H., Yilek, S.: Hedged public-key encryption: How to protect against bad randomness. In: Matsui, M. (ed.) Advances in Cryptology – ASIACRYPT 2009. Lecture Notes in Computer Science, vol. 5912, pp. 232–249. Springer, Berlin, Germany, Tokyo, Japan (Dec 6–10, 2009)
10. Bellare, M., Fischlin, M., O'Neill, A., Ristenpart, T.: Deterministic encryption: Definitional equivalences and constructions without random oracles. In: Wagner, D. (ed.) Advances in Cryptology – CRYPTO 2008. Lecture Notes in Computer Science, vol. 5157, pp. 360–378. Springer, Berlin, Germany, Santa Barbara, CA, USA (Aug 17–21, 2008)
11. Bellare, M., Hoang, V.T.: UCE+LTDFs: Efficient, subversion-resistant PKE in the standard model. Cryptology ePrint Archive, Report 2014/876 (2014), http://eprint.iacr.org/2014/876
12. Bellare, M., Hoang, V.T., Keelveedhi, S.: Instantiating random oracles via UCEs. In: Canetti, R., Garay, J.A. (eds.) Advances in Cryptology – CRYPTO 2013, Part II. Lecture Notes in Computer Science, vol. 8043, pp. 398–415. Springer, Berlin, Germany, Santa Barbara, CA, USA (Aug 18–22, 2013)
13. Bellare, M., Hoang, V.T., Keelveedhi, S.: 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)
14. Bellare, M., Hoang, V.T., Keelveedhi, S.: Personal communication (Sep, 2013)
15. Bellare, M., Hoang, V.T., Keelveedhi, S.: 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
16. Bellare, M., Keelveedhi, S.: Authenticated and misuse-resistant encryption of key-dependent data. In: Rogaway, P. (ed.) Advances in Cryptology – CRYPTO 2011. Lecture Notes in Computer Science, vol. 6841, pp. 610–629. Springer, Berlin, Germany, Santa Barbara, CA, USA (Aug 14–18, 2011)
17. Bellare, M., Keelveedhi, S., Ristenpart, T.: Message-locked encryption and secure deduplication. In: Johansson, T., Nguyen, P.Q. (eds.) Advances in Cryptology – EUROCRYPT 2013. Lecture Notes in Computer Science, vol. 7881, pp. 296–312. Springer, Berlin, Germany, Athens, Greece (May 26–30, 2013)
18. Bellare, M., Rogaway, P.: Random oracles are practical: A paradigm for designing efficient protocols. In: Ashby, V. (ed.) ACM CCS 93: 1st Conference on Computer and Communications Security. pp. 62–73. ACM Press, Fairfax, Virginia, USA (Nov 3–5, 1993)
19. Bellare, M., Stepanovs, I., Tessaro, S.: Poly-many hardcore bits for any one-way function and a framework for differing-inputs obfuscation. In: Sarkar, P., Iwata, T. (eds.) Advances in Cryptology – ASIACRYPT 2014, Part II. Lecture Notes in Computer Science, vol. 8874, pp. 102–121. Springer, Berlin, Germany, Kaoshiung, Taiwan, R.O.C. (Dec 7–11, 2014)
20. Bitansky, N., Canetti, R., Paneth, O., Rosen, A.: On the existence of extractable one-way functions. In: Shmoys, D.B. (ed.) 46th Annual ACM Symposium on Theory of Computing. pp. 505–514. ACM Press, New York, NY, USA (May 31 – Jun 3, 2014)
21. Black, J.: The ideal-cipher model, revisited: An uninstantiable blockcipher-based hash function. In: Robshaw, M.J.B. (ed.) Fast Software Encryption – FSE 2006. Lecture Notes in Computer Science, vol. 4047, pp. 328–340. Springer, Berlin, Germany, Graz, Austria (Mar 15–17, 2006)

22. Boldyreva, A., Fehr, S., O'Neill, A.: On notions of security for deterministic encryption, and efficient constructions without random oracles. In: Wagner, D. (ed.) Advances in Cryptology – CRYPTO 2008. Lecture Notes in Computer Science, vol. 5157, pp. 335–359. Springer, Berlin, Germany, Santa Barbara, CA, USA (Aug 17–21, 2008)
23. Boldyreva, A., Fischlin, M.: Analysis of random oracle instantiation scenarios for OAEP and other practical schemes. In: Shoup, V. (ed.) Advances in Cryptology – CRYPTO 2005. Lecture Notes in Computer Science, vol. 3621, pp. 412–429. Springer, Berlin, Germany, Santa Barbara, CA, USA (Aug 14–18, 2005)
24. Boneh, D., Franklin, M.K.: Identity-based encryption from the Weil pairing. In: Kilian, J. (ed.) Advances in Cryptology – CRYPTO 2001. Lecture Notes in Computer Science, vol. 2139, pp. 213–229. Springer, Berlin, Germany, Santa Barbara, CA, USA (Aug 19–23, 2001)
25. Boyle, E., Chung, K.M., Pass, R.: On extractability obfuscation. In: Lindell, Y. (ed.) TCC 2014: 11th Theory of Cryptography Conference. Lecture Notes in Computer Science, vol. 8349, pp. 52–73. Springer, Berlin, Germany, San Diego, CA, USA (Feb 24–26, 2014)
26. Brakerski, Z., Rothblum, G.N.: Virtual black-box obfuscation for all circuits via generic graded encoding. In: Lindell, Y. (ed.) TCC 2014: 11th Theory of Cryptography Conference. Lecture Notes in Computer Science, vol. 8349, pp. 1–25. Springer, Berlin, Germany, San Diego, CA, USA (Feb 24–26, 2014)
27. Brzuska, C., Farshim, P., Mittelbach, A.: Indistinguishability obfuscation and UCEs: The case of computationally unpredictable sources. In: Garay, J.A., Gennaro, R. (eds.) Advances in Cryptology – CRYPTO 2014, Part I. Lecture Notes in Computer Science, vol. 8616, pp. 188–205. Springer, Berlin, Germany, Santa Barbara, CA, USA (Aug 17–21, 2014)
28. Brzuska, C., Farshim, P., Mittelbach, A.: Random oracle uninstantiability from indistinguishability obfuscation. Cryptology ePrint Archive, Report 2014/867 (2014), http://eprint.iacr.org/2014/867
29. Brzuska, C., Mittelbach, A.: Deterministic public-key encryption from indistinguishability obfuscation and point obfuscation (Sep, 2014)
30. Brzuska, C., Mittelbach, A.: Indistinguishability obfuscation versus multi-bit point obfuscation with auxiliary input. In: Sarkar, P., Iwata, T. (eds.) Advances in Cryptology – ASIACRYPT 2014, Part II. Lecture Notes in Computer Science, vol. 8874, pp. 142–161. Springer, Berlin, Germany, Kaoshiung, Taiwan, R.O.C. (Dec 7–11, 2014)
31. Brzuska, C., Mittelbach, A.: Using indistinguishability obfuscation via UCEs. In: Sarkar, P., Iwata, T. (eds.) Advances in Cryptology – ASIACRYPT 2014, Part II. Lecture Notes in Computer Science, vol. 8874, pp. 122–141. Springer, Berlin, Germany, Kaoshiung, Taiwan, R.O.C. (Dec 7–11, 2014)
32. Canetti, R., Dakdouk, R.R.: Extractable perfectly one-way functions. In: Aceto, L., Damgård, I., Goldberg, L.A., Halldórsson, M.M., Ingólfsdóttir, A., Walukiewicz, I. (eds.) ICALP 2008: 35th International Colloquium on Automata, Languages and Programming, Part II. Lecture Notes in Computer Science, vol. 5126, pp. 449–460. Springer, Berlin, Germany, Reykjavik, Iceland (Jul 7–11, 2008)
33. Canetti, R., Goldreich, O., Halevi, S.: The random oracle methodology, revisited (preliminary version). In: 30th Annual ACM Symposium on Theory of Computing. pp. 209–218. ACM Press, Dallas, Texas, USA (May 23–26, 1998)
34. Canetti, R., Goldreich, O., Halevi, S.: 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

35. Canetti, R., Halevi, S., Katz, J.: A forward-secure public-key encryption scheme. In: Biham, E. (ed.) Advances in Cryptology – EUROCRYPT 2003. Lecture Notes in Computer Science, vol. 2656, pp. 255–271. Springer, Berlin, Germany, Warsaw, Poland (May 4–8, 2003)
36. Dent, A.W.: Adapting the weaknesses of the random oracle model to the generic group model. In: Zheng, Y. (ed.) Advances in Cryptology – ASIACRYPT 2002. Lecture Notes in Computer Science, vol. 2501, pp. 100–109. Springer, Berlin, Germany, Queenstown, New Zealand (Dec 1–5, 2002)
37. Douceur, J.R., Adya, A., Bolosky, W.J., Simon, D., Theimer, M.: Reclaiming space from duplicate files in a serverless distributed file system. In: International Conference on Distributed Computing Systems. pp. 617–624 (2002)
38. Fuller, B., O'Neill, A., Reyzin, L.: A unified approach to deterministic encryption: New constructions and a connection to computational entropy. In: Cramer, R. (ed.) TCC 2012: 9th Theory of Cryptography Conference. Lecture Notes in Computer Science, vol. 7194, pp. 582–599. Springer, Berlin, Germany, Taormina, Sicily, Italy (Mar 19–21, 2012)
39. Garg, S., Gentry, C., Halevi, S., Raykova, M., Sahai, A., Waters, B.: Candidate indistinguishability obfuscation and functional encryption for all circuits. In: 54th Annual Symposium on Foundations of Computer Science. pp. 40–49. IEEE Computer Society Press, Berkeley, CA, USA (Oct 26–29, 2013)
40. Garg, S., Gentry, C., Halevi, S., Wichs, D.: On the implausibility of differing-inputs obfuscation and extractable witness encryption with auxiliary input. In: Garay, J.A., Gennaro, R. (eds.) Advances in Cryptology – CRYPTO 2014, Part I. Lecture Notes in Computer Science, vol. 8616, pp. 518–535. Springer, Berlin, Germany, Santa Barbara, CA, USA (Aug 17–21, 2014)
41. Gentry, C.: Certificate-based encryption and the certificate revocation problem. In: Biham, E. (ed.) Advances in Cryptology – EUROCRYPT 2003. Lecture Notes in Computer Science, vol. 2656, pp. 272–293. Springer, Berlin, Germany, Warsaw, Poland (May 4–8, 2003)
42. Gentry, C., Lewko, A., Sahai, A., Waters, B.: Indistinguishability obfuscation from the multilinear subgroup elimination assumption. Cryptology ePrint Archive, Report 2014/309 (2014), http://eprint.iacr.org/2014/309
43. Gentry, C., Silverberg, A.: Hierarchical ID-based cryptography. In: Zheng, Y. (ed.) Advances in Cryptology – ASIACRYPT 2002. Lecture Notes in Computer Science, vol. 2501, pp. 548–566. Springer, Berlin, Germany, Queenstown, New Zealand (Dec 1–5, 2002)
44. Goldwasser, S., Kalai, Y.T.: On the (in)security of the Fiat-Shamir paradigm. In: 44th Annual Symposium on Foundations of Computer Science. pp. 102–115. IEEE Computer Society Press, Cambridge, Massachusetts, USA (Oct 11–14, 2003)
45. Green, M.D., Katz, J., Malozemoff, A.J., Zhou, H.S.: A unified approach to idealized model separations via indistinguishability obfuscation. Cryptology ePrint Archive, Report 2014/863 (2014), http://eprint.iacr.org/2014/863
46. Hohenberger, S., Sahai, A., Waters, B.: Replacing a random oracle: Full domain hash from indistinguishability obfuscation. In: Nguyen, P.Q., Oswald, E. (eds.) Advances in Cryptology – EUROCRYPT 2014. Lecture Notes in Computer Science, vol. 8441, pp. 201–220. Springer, Berlin, Germany, Copenhagen, Denmark (May 11–15, 2014)
47. Koppula, V., Lewko, A.B., Waters, B.: Indistinguishability obfuscation for turing machines with unbounded memory. Cryptology ePrint Archive, Report 2014/925 (2014), http://eprint.iacr.org/2014/925

48. Lynn, B., Prabhakaran, M., Sahai, A.: Positive results and techniques for obfuscation. In: Cachin, C., Camenisch, J. (eds.) Advances in Cryptology – EUROCRYPT 2004. Lecture Notes in Computer Science, vol. 3027, pp. 20–39. Springer, Berlin, Germany, Interlaken, Switzerland (May 2–6, 2004)
49. Matsuda, T., Hanaoka, G.: Chosen ciphertext security via point obfuscation. In: Lindell, Y. (ed.) TCC 2014: 11th Theory of Cryptography Conference. Lecture Notes in Computer Science, vol. 8349, pp. 95–120. Springer, Berlin, Germany, San Diego, CA, USA (Feb 24–26, 2014)
50. Mittelbach, A.: Salvaging indifferentiability in a multi-stage setting. In: Nguyen, P.Q., Oswald, E. (eds.) Advances in Cryptology – EUROCRYPT 2014. Lecture Notes in Computer Science, vol. 8441, pp. 603–621. Springer, Berlin, Germany, Copenhagen, Denmark (May 11–15, 2014)
51. Nielsen, J.B.: Separating random oracle proofs from complexity theoretic proofs: The non-committing encryption case. In: Yung, M. (ed.) Advances in Cryptology – CRYPTO 2002. Lecture Notes in Computer Science, vol. 2442, pp. 111–126. Springer, Berlin, Germany, Santa Barbara, CA, USA (Aug 18–22, 2002)
52. Pass, R., Seth, K., Telang, S.: Indistinguishability obfuscation from semantically-secure multilinear encodings. In: Garay, J.A., Gennaro, R. (eds.) Advances in Cryptology – CRYPTO 2014, Part I. Lecture Notes in Computer Science, vol. 8616, pp. 500–517. Springer, Berlin, Germany, Santa Barbara, CA, USA (Aug 17–21, 2014)
53. Raghunathan, A., Segev, G., Vadhan, S.P.: Deterministic public-key encryption for adaptively chosen plaintext distributions. In: Johansson, T., Nguyen, P.Q. (eds.) Advances in Cryptology – EUROCRYPT 2013. Lecture Notes in Computer Science, vol. 7881, pp. 93–110. Springer, Berlin, Germany, Athens, Greece (May 26–30, 2013)
54. Ristenpart, T., Shacham, H., Shrimpton, T.: Careful with composition: Limitations of the indifferentiability framework. In: Paterson, K.G. (ed.) Advances in Cryptology – EUROCRYPT 2011. Lecture Notes in Computer Science, vol. 6632, pp. 487–506. Springer, Berlin, Germany, Tallinn, Estonia (May 15–19, 2011)
55. Sahai, A., Waters, B.: How to use indistinguishability obfuscation: deniable encryption, and more. In: Shmoys, D.B. (ed.) 46th Annual ACM Symposium on Theory of Computing. pp. 475–484. ACM Press, New York, NY, USA (May 31 – Jun 3, 2014)
56. Sahai, A., Waters, B.R.: Fuzzy identity-based encryption. In: Cramer, R. (ed.) Advances in Cryptology – EUROCRYPT 2005. Lecture Notes in Computer Science, vol. 3494, pp. 457–473. Springer, Berlin, Germany, Aarhus, Denmark (May 22–26, 2005)
57. Wichs, D.: Barriers in cryptography with weak, correlated and leaky sources. In: Kleinberg, R.D. (ed.) ITCS 2013: 4th Innovations in Theoretical Computer Science. pp. 111–126. Association for Computing Machinery, Berkeley, CA, USA (Jan 9–12, 2013)