Random-Oracle Uninstantiability from Indistinguishability Obfuscation

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

[1]Microsoft Research Cambridge, UK
[2]Queen's University Belfast, UK
[3]Darmstadt University of Technology, Germany
christina.brzuska@gmail.com pooya.farshim@gmail.com arno.mittelbach@cased.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 circuits or iO for Turing machines to derive uninstantiability for hash functions of a priori bounded polynomial size and arbitrary polynomial size, respectively. 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, hedged PKE, Fujisaki–Okamoto, KDM security, message-locked encryption, UCE.

Contents

1 Introduction

1.1 Background

The random-oracle model (ROM) [BR93] is an idealized model of computation where all parties, honest or otherwise, have oracle access to a uniformly chosen random function. Random oracles (ROs) 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. Once a scheme is designed and analyzed in the random-oracle model, one instantiates the oracle via a concrete hash function, tacitly assuming that it has a "RO-like" behavior. In this paper we revisit this random-oracle methodology and show that a number of prominent RO-model transforms cannot be securely instantiated with any hash function 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) [CGH98] 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 the 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 something "obviously insecure," is done (e.g., the signing key or the message is returned).

A number of other works have further studied uninstantiability problems associated with random oracles. In a follow-up work CGH [CGH03] extend their result to signature schemes which only support short messages. Bellare, Boldyreva and Palacio [BBP04] show that no instantiation of the hashed ElGamal key-encapsulation mechanism composes well with symmetric schemes, even though it enjoys this property in the ROM. Goldwasser and Kalai [GK03] study the Fiat–Shamir heuristic and establish uninstantiability results for it. Nielsen [Nie02] 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 [Bla06] and the generic-group model [Den02].

A number of recent works have looked into ROM (un)instantiability in light of the recently proposed candidate for indistinguishability obfuscation (iO) [GGH$^+$13]. 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 [HSW14] show how to instantiate the hash function in full-domain hash (FDH) signatures using iO. Bellare, Stepanovs and Tessaro [BST14] give the first standard-model construction for polynomially many hard-core bits for *any* one-way function. Recently, Brzuska and Mittelbach [BM14c] 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 [BHK13a] and can be used to generically instantiate random oracles in many protocols.

On the negative side, Brzuska, Farshim and Mittelbach [BFM14] 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 [BM14b] show that assuming iO, multi-bit output point-function obfuscation secure in the presence of auxiliary information cannot be realized. Both results can be interpreted as (conditional) uninstantiability results, as ROM constructions for both UCEs [BHK13a, Mit14] and strong multi-output bit point obfuscation [LPS04] exist. Bitansky et al. [BCPR14] show that indistinguishability obfuscation rules out the existence of certain types of extractable one-way function families which can be constructed in the random-oracle model [CD08].

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 all mildly secure S. We show a number of negative results in this direction, which take the following 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 p, we show there is a ROM cryptosystem which is uninstantiable with respect to keyed hash functions of description size at most p. This means that there exists a scheme S_{p} such that for any hash function H of description size at most p the scheme $\mathsf{T}^{\mathsf{H}}[\mathsf{S}_{\mathrm{p}}]$ 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) [BFM14] 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. (See Section 3.3 for a brief description of UCEs.) To this end, BFM construct an adversary 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. 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 a random y (with overwhelming probability over the choice of y) is outside the range of the function $\mathsf{H}(\cdot, x)$ that maps hash keys hk to $\mathsf{H}(\mathsf{hk}, x)$ for a fixed x. On the other hand, the above circuit returns 1 when the hash value y is computed as $\mathsf{H}(\mathsf{hk}, x)$ and 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. We use programs for either circuits or Turing machines throughout the paper. 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 underlying model of computation:

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

That is, program $\mathsf{P}[x, y]$ has x and y hard-coded in, takes as input a description of H_{hk}, computes $\mathsf{H}_{\mathsf{hk}}(x)$ and checks whether this value is equal to y. We no longer consider a fixed keyed hash function here, but instead (potentially) consider the set of *all* hash functions on a given range and domain.[1] (Similar ideas have been used by Brzuska and Mittelbach [BM14b] to study the feasibility of multi-bit output point function obfuscation in the presence of auxiliary inputs under the iO assumption.) 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.

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 standard indistinguishability obfuscators support. We address this problem in two incomparable ways. First, we weaken the target uninstantiability and under iO for circuits rule out instantiations by a priori bounded-size hash functions. Second, in order to obtain 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 for Turing machines is that an obfuscated Turing machine is still a Turing machine and 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 the program

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

This does not affect the success probability of the attack and allows us to argue that x remains hidden as follows. First, note that on the right-hand side of the check $\mathsf{PRG}(y)$ is a constant that does not depend on the program input and can thus be hard-coded into P. 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 with overwhelming probability is outside the range of a PRG that has a sufficiently long stretch. Hence, by iO security, the obfuscation of the above program is computationally indistinguishable from an obfuscation 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 [SW14], the range extension of Matsuda and Hanaoka [MH14] of a multi-bit point function to obtain shorter point values, the range-extension of a UCE1-secure hash function by Bellare, Hoang and Keelveedhi [BHK13c], and the negative result of Brzuska and Mittelbach [BM14b] on multi-bit point-function obfuscation with auxiliary inputs.

Assumptions. Garg et al. [GGH$^+$13] 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

[1] Alternatively, we can consider the universal hash function.

support all polynomial-size circuits via a fully homomorphic encryption scheme which has an $\mathcal{NC}^1$ decryption circuit. 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 [PST14, BR14, BGK$^+$14, AGIS14, GLSW14], further supporting their plausibility.

Indistinguishability obfuscation for Turing machines has been constructed in the works of Boyle, Chung and Pass [BCP14] and Ananth et al. [ABG$^+$13]. The authors study a stronger primitive called *extractability* or *differing-inputs* obfuscation (diO) which extends iO to circuits (and Turing machines) that are not necessarily functionally equivalent. Security of diO requires that any adversary that can break the indistinguishability property can be converted to an extractor that can output a point on which the two circuits differ. Boyle, Chung and Pass [BCP14] and Ananth et al. [ABG$^+$13] 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. [GGHW14]. They 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, and hence more likely.[2] 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 [KLW14] 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 [BBO07], 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 $\mathsf{D\text{-}PKE.Enc}^{\mathcal{RO}}(m, (\mathsf{hk}, pk))$ 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 in the ROM if the underlying encryption scheme is IND-CPA secure. Roughly speaking, a deterministic public-key encryption is IND-secure if no adversary can distinguish the encryptions of two high-entropy and pk-independent messages. Known standard-model constructions, on the other hand, achieve weaker levels of security, e.g., security against block sources [BFOR08, BFO08]. Deterministic public-key encryption secure against q-bounded adversaries can be build from correlation-secure trapdoor functions [FOR12] or iO [BM14a].

We ask if any hash function can be used to instantiate the random oracle within the EwH transform to achieve IND security. Assuming iO for circuits/Turing machines, we build an IND-CPA-secure encryption scheme such that when the EwH transform is applied to it together with some (p-bounded) hash function, the resulting scheme is *not* secure, not even for block-sources or 1-bounded adversaries.

In more details, starting with an arbitrary scheme PKE we consider a new scheme PKE^* which includes as part of its ciphertexts an indistinguishability obfuscation of the program

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

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

[2] Instantiating this special-purpose obfuscator from more standard assumptions is an open problem.

We can use an obfuscation of this program to attack the security of $\mathsf{EwH}^{\mathsf{H}}[\mathsf{PKE}^*]$. 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. (Note that this (second-stage) adversary gets to see the public and hence hk.) 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 [BHK14] is uninstantiable assuming iO for Turing machines (and p-bounded uninstantiable assuming iO for circuits). We remark that our results are incomparable to those of Wichs [Wic13] 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 transform for building hedged PKEs [BBN$^+$09].

The Fujisaki–Okamoto transform. The above result generalizes to a wider class of (possibly randomized) transformations that use their underlying PKE schemes in a structured way and admit recovery algorithms that satisfy certain correctness properties. We call these transformation *admissible.* (See Section 4 for the details.) 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 [BF01], its hierarchical and fuzzy variants [GS02, SW05], forward-secure encryption [CHK03], and certificateless and certificate-based encryption [ARP03, Gen03] 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 in particular means that the POWHF-encryption assumption of Boldyreva and Fischlin [BF05] used for partial instantiation of the oracles in FO is also uninstantiable if iO for Turing machines/circuits 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) [BK11] on authenticated and misuse-resistant encryption of key-dependent data and show that it too suffers from uninstantiability problems. Roughly speaking, BK give a transformation 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 seem to consider this difference to be of major importance: in the abstract of their paper [BK11] they write that they "present a RO-based 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,[3] then reverting to the stronger security notion would no longer be of any help.

As a final application of our techniques, we show that the Convergent-Encryption transform of Douceur et al. [DAB$^+$02] formalized by Bellare, Keelveedhi and Ristenpart (BKR) [BKR13] for building

[3] 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 only.

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) [CGH98] 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 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 a *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.[4] 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 to be successful it needs to coordinate the attack with the actual adversary (who does see the hash key). Indistinguishability obfuscation allows this distributed attack to be carried out.

Concurrent work. In concurrent and independent work, Green et al. [GKMZ14] 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 et 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. 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

[4] Despite this, CGH-like techniques render Encrypt-with-Hash uninstantiable when stronger notions of security are considered [RSV13].

programs such that by feeding obfuscations of one program into the obfuscations of another an attack can be 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 [BFM14, BM14c] and indeed indistinguishability obfuscation itself. For instance, recently Bellare and Hoang [BH14] 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 find a transformation from encryption to some other security notion that suffers from iO-based uninstantiability when the encryption scheme is CPA-secure but becomes instantiable when the encryption scheme has ciphertexts that are indistinguishable from random strings. Such a result would be interesting even for somewhat artificial tasks. These, in turn, 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 empty string by ε, 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 ith bit of a string x is indicated by $x[i]$. A vector of strings $\mathbf{x}$ is written in boldface, and $\mathbf{x}[i]$ denotes its ith 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 this is explicitly permitted 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 P, 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. [BGI+01, BGI+12] 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 [BST14] with extensions to also cover obfuscation for Turing machines [ABG+13]. 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 $|\mathsf{P}'| \in \mathcal{O}\left(\mathsf{poly}(\lambda + |\mathsf{P}|)\right)$.

- 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

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

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. [GGH+13] prove that under intractability assumptions related to multi-linear maps an indistinguishability obfuscator supporting all $\mathcal{NC}^1$ circuits exists. Assuming the existence of a perfectly correct leveled fully homomorphic encryption with $\mathcal{NC}^1$ decryption 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 $\mathcal{C} := \{\mathcal{C}_{\mathrm{p}(\lambda)}\}_{\lambda\in\mathbb{N}}$ where p is a polynomial and

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

For any bound p, existence of iO for $\mathcal{C}_{\mathrm{p}(\lambda)}$ under our definition is implied by the (non-uniform) definition of Garg et al. [GGH+13].

Several follow-up works improved the assumptions underlying indistinguishability obfuscators as well as the performance [PST14, BR14, AGIS14, BGK+14, GLSW14]. As mentioned above, circuits and obfuscations thereof admit fixed-length inputs only.

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

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

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

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

Ananth et al. [ABG$^+$13] and Boyle et al. [BCP14] give constructions of indistinguishability obfuscators for Turing machines which admit inputs of arbitrary lengths. Their constructions achieve the stronger notion of *differing-inputs* (a.k.a. extractability) obfuscation, initially also suggested in the work of Barak et al. [BGI$^+$01, BGI$^+$12]. This type of obfuscation can be regarded as a generalization of indistinguishability obfuscation to programs which are not necessarily functionally equivalent. We recall [ABG$^+$13, Theorem 3] here and refer the reader to the original works for details and discussion.

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

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

Koppula, Lewko and Waters [KLW14] have recently succeeded in constructing iO for Turing machines without relying on diO and using iO for circuits, one-way functions and injective pseudorandom generators only.

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 public key pk, a message m and possibly some explicit random coins r and outputs a ciphertext c. The deterministic decryption algorithm $\mathsf{PKE.Dec}(sk, c)$ is given a secret key sk and a ciphertext c, 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

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

is negligible.

Function families. Following [BST14], we define a function family FF as a five tuple of PPT algorithms $(\mathsf{FF.Kg}, \mathsf{FF.Ev}, \mathsf{FF.kl}, \mathsf{FF.il}, \mathsf{FF.ol})$ where the algorithms $\mathsf{FF.kl}$, $\mathsf{FF.il}$, 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 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

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

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

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

is negligible.

Keyed random oracles. 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 generated and who gets it, 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). 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 hash keys 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 results are mostly negative, it suffices to consider weak adversaries that do not get oracle access and/or the hash key in some of their stages.

(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 p-*uninstantiable*, if no hash function of size at most $\mathrm{p}(\lambda)$ 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) [BBO07] 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 [BBO07]. 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., $\mathsf{D\text{-}PKE.rl}(\lambda) = 0$ for all λ). BBO [BBO07] model the security of D-PKEs via a form of simulation-based notion called PRIV. In later works, Bellare et al. [BFOR08] and independently Boldyreva, Fehr and O'Neill [BFO08] introduce an indistinguishability-based notion called IND and show that it is equivalent to PRIV security. The IND game is formally defined in Figure 1 on the right.[5] 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 $(sk, pk) \leftarrow\!\!\$\; \mathsf{D\text{-}PKE.Kg}(1^\lambda)$ is then chosen, and according to the challenge bit b, one of the two message vectors is component-wise encrypted. The second-stage adversary $\mathcal{A}_2$ is run on the resulting vector of ciphertexts and the public key, and wins the game if it correctly guesses the hidden bit b. We define the advantage of an adversary $\mathcal{A}$ in the IND game (see Figure 1) against scheme $\mathsf{D\text{-}PKE}$ 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 q-bounded version of the model demands that $(\mathbf{m}_0, \mathbf{m}_1)$ contain at most q message each. All the adversaries that we consider in our uninstantiability results will be 1-bounded (and thus also use block sources).

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 [BBO07]. 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 any IND-CPA public-key encryption scheme.

[5] Bellare et al. [BFOR08] 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.

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 also 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 [CGH98]. (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.[6] 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 p-uninstantiability: for any polynomial bound p there exists an IND-CPA scheme that cannot be securely used in EwH for any hash function whose description size is at most p. The latter result is also 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 this 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 3.1 (Uninstantiability of EwH). *Assuming the existence of indistinguishability obfuscation for Turing machines $\mathcal{M}$ (resp. p-bounded circuits $\mathcal{C}_{\mathrm{p}}$), the* EwH *transform is uninstantiable (resp. p-uninstantiable) with respect to* IND *security and* IND-CPA *base schemes 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 [BFM14],

[6] 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 [RSV13].

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:

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

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

For (1), note that given an obfuscation of $\mathsf{P}'[pk, m, r]$ and 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 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.

In other words, 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:

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

At this point it might appear that no progress has been made as the above program, for reasons similar to those given above, is not functionally equivalent to Z. We note, however, that for a 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.

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 formalize this proof intuition next.

Proof (of Theorem 3.1). 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

$$\begin{tikzcd} \mathsf{Game}_0(\lambda) \arrow[r, bend left, "\mathsf{PRG}"] & \mathsf{Game}_1(\lambda) \arrow[r, bend left, "\mathsf{iO}"] & \mathsf{Game}_2(\lambda) \end{tikzcd}$$

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

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

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

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

Figure 2: Hybrids used in the proof of Theorem 3.1 (left) and the program P obfuscated in the first two games (right). The highlighted lines show the changes in game transitions.

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 p 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).

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 with overwhelming probability outputs functionally equivalent circuits. 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 Turing 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 depends 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 $\mathsf{D\text{-}PKE}^* := \mathsf{EwH}^{\mathsf{H}}[\mathsf{PKE}^*]$ for any function H that respects the input requirements of P (arbitrary if P is a Turing machine, and p-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 (hk, pk) 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 $\mathrm{p}(\lambda)$.) 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:

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

□

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

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

Figure 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 3.1. The boxed algorithms are to be understood as subroutines. Program P that is obfuscated as part of ciphertexts is given on the right.

3.3 Consequences for UCEs

We turn to Universal Computational Extractors (UCEs), a novel notion introduced by Bellare, Hoang and Keelveedhi (BHK) [BHK13a] to generically instantiate 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. Roughly speaking, in UCE1 security (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. The latter additionally gets the hash key, can no longer call the oracle, and outputs a bit. 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.

One application of this new framework has been to the EwH transform. BHK [BHK14] show that if a scheme PKE is IND-CPA secure and a hash function H meets what they call 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 UCE[$\mathcal{S}^{\mathrm{cup}} \cap \mathcal{S}_{\mathsf{PKE}}$] security is uninstantiable.

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

We remark that in previous versions of their work [BHK13a, BHK13b], BHK based the security of

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

Figure 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 [RSS11].

EwH on other stronger UCE assumptions. Our results also show the uninstantiability of these notions assuming indistinguishability obfuscation and in particular imply the negative results of [BFM14] for UCE[$\mathcal{S}^{\text{cup}}$] security and *bounded parallel sources*, a notion introduced in [BHK13b], as both of these notions imply a secure instantiation of EwH. The idea behind bounded parallel sources is to circumvent iO attacks by potentially exploiting the efficiency of public-key encryption compared to obfuscation. As our counterexample encryption scheme uses an obfuscator as a subroutine, the scheme is at least as complex as the obfuscator and this intuition does not apply anymore. Note, however, that BFM [BFM14] rule out the instantiability of bounded parallel sources for a wider choice of parameters, even for extremely efficient public-key encryption schemes.

3.4 Extension to hedged PKEs

Hedged public-key encryption, introduced by Bellare et al. [BBN$^+$09] models the security of public-key encryption schemes where the random coins used in encryption might have low entropy. Indistinguishability under chosen-distribution attacks (IND-CDA) shown in Figure 4 formalizes the security of hedged PKEs. This notion is similar to IND and the only difference is that the adversary additionally to the two message vectors also outputs a randomness vector. The high min-entropy restriction is spread over the message and randomness vectors. When the length of the randomness entries is 0, one recovers the IND model for D-PKEs. A transform similar to EwH, called Randomized Encrypt-with-Hash, can be defined for hedged PKEs [BBN$^+$09]: 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. $\mathsf{P}[pk, m, s](\mathsf{H}(\mathsf{hk}, \cdot), \rho)$
$r \leftarrow \mathsf{UEval}(\mathsf{H}(\mathsf{hk}, \cdot), pk\|m\|\rho)$
$s' \leftarrow \mathsf{PRG}(r)$
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.

4 Beyond Encrypt-with-Hash

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 of the previous section to what we call *admissible* transformations, and show that the classical and widely used 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.

4.1 Generalizing the attack

Let $\mathsf{GT}^{\mathcal{RO}}[\mathsf{PKE}]$ be a ROM transformation mapping PKE schemes to PKE schemes. Without loss of generality we assume that there is a single random oracle as multiple random oracles can be simulated via domain separation. When the oracle in $\mathsf{GT}^{\mathcal{RO}}[\mathsf{PKE}]$ is instantiated with a hash function H we write $\mathsf{GT}^{\mathsf{H}}[\mathsf{PKE}]$. We say that transform GT is *structured* if it takes the following form for a (possibly randomized) oracle PPT machine $\mathsf{T}^{\mathcal{RO}}$ and a public-key encryption scheme PKE.

```
$\mathsf{GT}^{\mathcal{RO}}[\mathsf{PKE}].\mathsf{Enc}(pk, m; r)$
$(pk', m', r', c') \leftarrow \mathsf{T}^{\mathcal{RO}}(pk, m; r)$
$c \leftarrow \mathsf{PKE.Enc}(pk', m'; r')$
return $(c, c')$
```

Note that in order to obtain a deterministic encryption routine T needs to be deterministic (i.e., $r = \varepsilon$). For the sake of generality, however, we allow T to be randomized.

In addition to the above structural requirement on $\mathsf{GT}^{\mathcal{RO}}$, we require the existence of a deterministic *recovery* algorithm $\mathsf{Rec}^{\mathcal{RO}}$ such that experiment Recover returns true with probability 1 for any valid choice of message m.[7]

```
Exp. $\mathrm{Recover}_{\mathsf{PKE},\mathsf{T},\mathsf{Rec},m}(\lambda)$
$\mathcal{RO} \leftarrow\$ \mathrm{Func}(\mathsf{kl}(\lambda), \mathsf{il}(\lambda), \mathsf{ol}(\lambda))$
$r \leftarrow\$ \{0,1\}^{\mathsf{PKE.rl}(\lambda)}; (sk, pk) \leftarrow\$ \mathsf{PKE.Kg}(1^\lambda)$
$(pk', m', r', c') \leftarrow \mathsf{T}^{\mathcal{RO}}(pk, m; r)$
$c \leftarrow \mathsf{PKE.Enc}(pk', m'; r')$
$(m^*, r^*) \leftarrow \mathsf{Rec}^{\mathcal{RO}}(pk', m', c, c')$
return $(m^* = m \wedge r^* = r')$.
```

Note that $\mathsf{Rec}^{\mathcal{RO}}$ gets to see the adapted public key pk', the adapted message m' and the ciphertext (c, c') as computed by a transformation $\mathsf{GT}^{\mathcal{RO}}[\mathsf{PKE}]$ but it does *not* get to see the secret key sk nor the original randomness r (if present). We call a transformation *admissible* if it is structured and admits a recovery algorithm.

As an example, let us check that the Encrypt-with-Hash transformation is admissible. The encryption operation of Encrypt-with-Hash is given by

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

This can be re-written in the above structured form as follows.

[7] For our results it is, in fact, sufficient if the recovery algorithm succeeds only with noticeable probability.

```
$\mathsf{EwH}^{\mathcal{RO}}[\mathsf{PKE}].\mathsf{Enc}(pk, m; \varepsilon)$
$(pk', m', r', c') \leftarrow$ ┌ $\mathsf{T}^{\mathcal{RO}}(pk, m; \varepsilon)$
                               │   $r' \leftarrow \mathcal{RO}(pk \| m)$
                               │   return $(pk, m, r', \varepsilon)$
                               └
$c \leftarrow \mathsf{PKE.Enc}(pk', m'; r')$
return $(c, c')$
```

Note that EwH is deterministic, and so we set $r = \varepsilon$. The recovery algorithm $\mathsf{Rec}^{\mathcal{RO}}$ on input (pk', m', c, c') outputs

$$\Big(m', \mathcal{RO}(pk' \| m')\Big) \ .$$

As $m' = m$ and $pk' = pk$ this is the required output in order to succeed in experiment Recover.

We now give our generalized uninstantiability result. Since we consider both randomized and deterministic transformations, in the former case we show that the scheme resulting from applying the transformation to an adversarial scheme is not IND-CPA secure, and in the latter case we show this for IND security.

Theorem 4.1 (Uninstantiability of admissible transforms)**.** *Let* $\mathsf{GT}^{\mathcal{RO}}$ *be an admissible transformation. Assuming the existence of indistinguishability obfuscation for Turing machines* $\mathcal{M}$ *(resp.* p*-bounded circuits* $\mathcal{C}_{\mathrm{p}}$*), the* $\mathsf{GT}^{\mathcal{RO}}$ *transform is uninstantiable (resp.* p*-uninstantiable) with respect to* IND *security and* IND-CPA *base schemes if* GT *is deterministic, and uninstantiable (resp.* p*-uninstantiable) with respect to* IND-CPA *security and* IND-CPA *base schemes if* GT *is randomized.*

Proof. Similarly to EwH, we modify a scheme PKE to a tweaked variant PKE^* by attaching obfuscations of a program that can be used to win the IND game for $\mathsf{GT}^{\mathsf{H}}[\mathsf{PKE}]$ in case $\mathsf{GT}^{\mathcal{RO}}$ yields a D-PKE scheme, or the IND-CPA game in case it yields a standard PKE scheme. The program that PKE^* obfuscates depends on the recovery algorithm Rec. Roughly speaking, the program uses the recovery algorithm to recompute the randomness in application of the underlying PKE scheme and to check its well-formedness. If this check passes the program outputs the original message m. Otherwise it outputs 0. The important difference here is that P on top of a hash-function description also takes a *ciphertext component* as input. In other words, this program allows the adversary to also exploit the ciphertext that it has as its disposal.

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

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

IND-CPA PRESERVATION. The tweaked scheme remains IND-CPA secure. The proof is analogous to that of Theorem 3.1. First we replace s with a truly random value in the generation of P. This is indistinguishable down to the security of the pseudorandom generator. Next we note that program P would only output a non-zero value in case s lies within the range of the pseudorandom generator. This occurs with only a negligible probability via the union bound. The proof then follows from the security of the indistinguishability obfuscator.

It remains to show that $\mathsf{GT}^{\mathsf{H}}[\mathsf{PKE}^*]$ is IND insecure assuming that T is deterministic and in case T is randomized that the instantiated scheme is not IND-CPA secure.

```
$\mathrm{IND}^{\mathcal{A}_1,\mathcal{A}_2}_{\mathsf{GT}^{\mathsf{H}}[\mathsf{PKE}^*]}(\lambda)$
$b \leftarrow\!\$\ \{0,1\}$
$(m_0, m_1) \leftarrow\!\$\ \mathcal{A}_1(1^\lambda)$
    $x_0 \leftarrow\!\$\ \{0,1\}^{\mathsf{PKE.il}(\lambda)-1}$
    $x_1 \leftarrow\!\$\ \{0,1\}^{\mathsf{PKE.il}(\lambda)-1}$
    return $(x_0\|0, x_1\|1)$
$(sk, pk) \leftarrow\!\$\ \mathsf{D\text{-}PKE.Kg}(1^\lambda)$
$\mathsf{hk} \leftarrow\!\$\ \mathsf{H.Kg}(1^\lambda)$
$(c, \overline{\mathsf{P}}, c') \leftarrow \mathsf{GT}^{\mathsf{H}}[\mathsf{PKE}^*].\mathsf{Enc}((\mathsf{hk}, pk), m_b; \varepsilon)$
    $(pk', m', (r'\|r''), c') \leftarrow \mathsf{T}^{\mathsf{H}}(pk, m_b; \varepsilon)$
    $(c, \overline{\mathsf{P}}) \leftarrow \mathsf{PKE}^*.\mathsf{Enc}(pk', m'; r'\|r'')$
        $s \leftarrow \mathsf{PRG}(r')$
        $c \leftarrow \mathsf{PKE.Enc}(pk', m'; s)$
        $\overline{\mathsf{P}} \leftarrow \mathsf{iO}(\mathsf{P}[pk', m', s, c](\cdot,\cdot); r'')$
        return $(c, \overline{\mathsf{P}})$
    return $(c, \overline{\mathsf{P}}, c')$
$b' \leftarrow \mathcal{A}_2(1^\lambda, (\mathsf{hk}, pk), (c, \overline{\mathsf{P}}, c'))$
    $m_b \leftarrow \overline{\mathsf{P}}(\mathsf{H}(\mathsf{hk},\cdot), c')$
    $b' \leftarrow m_b[|m_b|]$
    return $b'$
return $(b = b')$
```

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

Figure 5: The IND game with respect to $\mathsf{GT}^{\mathsf{H}}[\mathsf{PKE}^*]$ and adversary $(\mathcal{A}_1, \mathcal{A}_2)$ as constructed in the proof of Theorem 4.1. For IND we assume that $\mathsf{GT}^{\mathsf{H}}[\mathsf{PKE}^*]$ is deterministic and thus $r = \varepsilon$.

Breaking IND. The attack is similar to that for EwH. We define $\mathcal{A}_1$ to output two random messages that end in 0 and 1 respectively. The second-stage adversary $\mathcal{A}_2$ gets as input $(pk, (c, \overline{\mathsf{P}}), c')$, that is, the public key pk, the ciphertext part $(c, \overline{\mathsf{P}})$ coming from our adapted scheme PKE^* and the second ciphertext part c' from the transformation GT (see definition of a structured transformation as described at the start of Section 4.1). Adversary $\mathcal{A}_2$ constructs a description of the hash function H with the hash key hk hard-coded in[8], and runs $\overline{\mathsf{P}}$ on (H, c'). It terminates by returning the least significant bit of $\overline{\mathsf{P}}$'s output. We give the pseudocode of the attack in Figure 5.

When m_b is encrypted, the adversary gets an obfuscation of $\mathsf{P}[pk', m', s, c]$ where pk' and m' are generated deterministically via T using input $(pk, m_b; \varepsilon)$. (Note that we are currently considering deterministic transformations.) On input (H, c') program $\mathsf{P}[pk', m', s, c]$ runs the recovery algorithm Rec on input (pk', c, c', m'), giving it oracle access to H. The recovery algorithm outputs $(m_b, r\|r')$, where $r\|r'$ were the coins given to PKE^*. Program P then recomputes $\mathsf{PRG}(r)$. This matches s by the definition of GT, and hence the program outputs m_b. Thus the adversary recovers the least significant bit of the encrypted message with probability 1.

Breaking IND-CPA. In this case, we consider an IND-CPA adversary $\mathcal{A}$ that outputs $m_0 := 0$ and $m_1 := 1$ as its chosen plaintexts and in its second phase launches the second stage of the IND attack above and checks which messages is recovered. The analysis is identical to the above case, noting that the recovery algorithm recovers the encrypted message as well as the random coins given to the PKE encryption operation.[9] □

[8] Note that the description of the hash function is public and the key hk is either public or part of the public key pk.

[9] Alternatively, simply note that IND implies IND-CPA security for pk-independent messages.

Strong IND. The IND security game for deterministic PKEs only applies to public-key-independent distributions. Raghunathan, Segev and Vadhan [RSV13] strengthen this definition to allow the adversaries to adaptively choose messages after learning some information about the public key. They then give constructions in the standard model and present two new transformations in the random-oracle model. The first scheme generates an additional random value u as part of the public key and sets the required randomness to $\mathcal{RO}(\mathsf{hk}, m\|u)$; that is, the entire public key is *not* used for randomness generation but only a specific part of it. The second scheme is parameterized by a polynomial Q and generates randomness as $\oplus_{i=1}^{\mathrm{Q}+1}\mathcal{RO}(\mathsf{hk}, m\|i)$. Both of these schemes fall prey to our iO-based attacks as they can be shown admissible similarly to the EwH transformation.

4.2 The Fujisaki–Okamoto transformation

The Fujisaki–Okamoto (FO) transformation [FO99] is a ROM technique to convert weak public-key encryption schemes, e.g., those which are indistinguishable (or even one-way) against chosen-plaintext attacks into strong ones which resist chosen-ciphertext attacks (i.e., are IND-CCA secure). In this transform a public-key encryption scheme PKE, a (deterministic) symmetric encryption scheme SE and two independent random oracles $\mathcal{RO}_1$ and $\mathcal{RO}_2$ are used. Under the FO transform, a ciphertext for a message m is generated by picking a fresh random value σ—FO is randomized—which will be hashed and then used as key for the symmetric scheme which in turn is used to encrypt the actual message m. The asymmetric scheme PKE is then used to encrypt σ in a checkable way: the randomness used to encrypt can be derived from message m and value σ:

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

In the standard model the random oracles are instantiated with keyed hash functions H and G. The hash keys are assumed to be a part of the public key. We denote such a standard-model instantiation by $\mathsf{FO}^{\mathsf{H},\mathsf{G}}[\mathsf{PKE},\mathsf{SE}]$. Similarly to EwH, the FO transformation is admissible and Theorem 4.1 implies its uninstantiability. We show that FO is uninstantiable even with respect to the weaker IND-CPA (rather than IND-CCA) security.

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

Proof. We show that $\mathsf{FO}^{\mathcal{RO}_1,\mathcal{RO}_2}[\mathsf{PKE},\mathsf{SE}]$ is admissible. The result would then follow from Theorem 4.1. The required transformation T and the recovery algorithm Rec are shown below.

```
ALGO. $\mathsf{FO}^{\mathcal{RO}_1,\mathcal{RO}_2}[\mathsf{PKE},\mathsf{SE}].\mathsf{Enc}(pk, m; \sigma)$
$(pk', m', r', c') \leftarrow$ $\mathsf{T}^{\mathcal{RO}_1,\mathcal{RO}_2}(pk, m; \sigma)$
                              $m' \leftarrow \sigma$
                              $r' \leftarrow \mathcal{RO}_1(\sigma\|m)$
                              $c \leftarrow \mathsf{SE}.\mathsf{Enc}(\mathcal{RO}_2(\sigma), m)$
                              return $(pk, m', r', c')$
$c \leftarrow \mathsf{PKE}.\mathsf{Enc}(pk', m'; r')$
return $(c, c')$
```

```
ALGO. $\mathsf{Rec}^{\mathcal{RO}_1,\mathcal{RO}_2}[\mathsf{PKE},\mathsf{SE}](pk', m', c, c')$
    $\sigma \leftarrow m'$
    $m \leftarrow \mathsf{SE}.\mathsf{Dec}(\mathcal{RO}_2(\sigma), c')$
    $r' \leftarrow \mathcal{RO}_1(\sigma\|m)$
    return $(m, r')$
```

□

Partial instantiations of FO. Boldyreva and Fischlin [BF05] study the security of the FO transformation when only *one* of the two random oracles in the construction is instantiated.[10] They consider perfectly one-way hash functions (POWHF) [Can97], which, on a high-level, hide all information about pre-images even given the hash key. Boldyreva and Fischlin [BF05] show that under an assumption which they call *POWHF encryption* one can securely instantiate the $\mathcal{RO}_1$ oracle. The POWHF-encryption assumption asks that for any efficient message distribution $\mathcal{M}$ the following two distributions are computationally indistinguishable.

```
$(sk, pk) \leftarrow_\$ \mathsf{PKE.Kg}(1^\lambda)$
$k \leftarrow_\$ \mathsf{POWHF.Kg}(1^\lambda)$
$r \leftarrow_\$ \mathsf{POWHF.coins}(\lambda)$
$(m, aux) \leftarrow_\$ \mathcal{M}(pk, k, r)$
$\sigma \leftarrow_\$ \{0,1\}^\lambda$
$\omega \leftarrow \mathsf{POWHF.Ev}(k, \sigma \| m; r)$
$c \leftarrow \mathsf{PKE.Enc}(pk, \sigma; \omega)$
return $(pk, k, r, c, aux)$
```

$$\approx$$

```
$(sk, pk) \leftarrow_\$ \mathsf{PKE.Kg}(1^\lambda)$
$k \leftarrow_\$ \mathsf{POWHF.Kg}(1^\lambda)$
$r \leftarrow_\$ \mathsf{POWHF.coins}(\lambda)$
$(m, aux) \leftarrow_\$ \mathcal{M}(pk, k, r)$
$\sigma \leftarrow_\$ \{0,1\}^\lambda$
$\omega \leftarrow_\$ \{0,1\}^{\mathsf{PKE.rl}(\lambda)}$
$c \leftarrow \mathsf{PKE.Enc}(pk, \sigma; \omega)$
return $(pk, k, r, c, aux)$
```

Looking at the proof of Corollary 4.2, and the obfuscated program P in particular (appearing in Theorem 4.1), we see that P uses both of its random oracles; that is, the recovery algorithm Rec, which is a subroutine of P, first decrypts a ciphertext component using $\mathcal{RO}_2$ and then recomputes the randomness using $\mathcal{RO}_1$. We can modify this algorithm and obtain an impossibility result for partial instantiations as follows. Instead of using the decryption routine, we hard-wire the message $m_1 := 1$ into the circuit. For this, it is crucial that we operate in the setting of IND-CPA security where the adversary can always submit the same two messages, say $m_0 := 0$ and $m_1 := 1$. (In contrast, for IND security as considered before, the messages were required to be unpredictable.) We use a program similar to that used in the proof of Theorem 4.2 and use the following subroutine Rec as follows.

```
Algo. $\mathsf{Rec}^{\mathcal{RO}_1}[\mathsf{PKE}, \mathsf{SE}, m_1](pk', m', c, c')$
$\sigma \leftarrow m'$
$r' \leftarrow \mathcal{RO}_1(\sigma \| m_1)$
return $(m_1, r')$
```

The second phase of IND-CPA adversary $\mathcal{A}$, as before, runs $\overline{\mathsf{P}}$ and outputs (the last bit of) the result. Through these modifications we have removed the dependency on the second random oracle altogether. We can restate our result as follows.

Corollary 4.3 (Partial uninstantiability of FO). *Assuming the existence of indistinguishability obfuscation for Turing machines $\mathcal{M}$ (resp.* p*-bounded circuits $\mathcal{C}_\mathrm{p}$), the first random oracle in the* FO *transformation is uninstantiable (resp.* p*-uninstantiable) with respect to* IND-CPA *security and* IND-CPA *base schemes in the standard model. In particular the POWFH-encryption assumption is uninstantiable (resp.* p*-uninstantiable) with respect to* IND-CPA *schemes in the standard model assuming iO for Turing machines (resp. for $\mathcal{C}_\mathrm{p}$).*

4.3 KDM security and message-locked encryption

So far we applied our techniques to transformations that operate on randomized public-key encryption schemes. In Appendices A and B we show that our techniques can be also applied in the symmetric setting and even to deterministic schemes. We give a brief overview of the results here.

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

Bellare and Keelveedhi (BK) [BK11] give a transformation to convert an authenticated encryption scheme into a KDM-secure one. This is done by hashing the key with a random nonce N to derive a new key, which will be then used for encryption. In other words, the encryption of m is $(N, \mathsf{SE.Enc}(\mathsf{H}(\mathsf{hk}, N\|k), m))$. We can construct a program P which breaks this scheme when $m = k$ is passed to it and leaves it intact otherwise. Roughly speaking, this program checks if m equals k by first hashing $N\|k$ (via the code provided to it) and comparing the result with a correct valued embedded into P upon encryption (encryption has access to the secret key). See Appendix A for the details and a discussion of security requirements from the base scheme.

Message-locked encryption (MLE) is a form of deterministic symmetric encryption where the encryption key is deterministically derived from the message that is to be encrypted. We show that the Convergent-Encryption transform of Douceur et al. [DAB$^+$02] formalized and proved secure by Bellare, Keelveedhi and Ristenpart (BKR) [BKR13] in the ROM for building message-locked encryption is also uninstantiable. Roughly speaking, in this transformation one encrypts m via $\mathsf{SE.Enc}(\mathsf{H}(\mathsf{hk}, m), m))$. The program that we use to break this transform follows a similar design pattern: when run on input a hash function it first computes a key by hashing a message hard-coded into it and checks if it matches the correct key hard-coded into it. See Appendix B for the details.

5 Careful with Conversion

In this section we 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.

The underlying idea behind this new transformation, which we term Hybrid and Double Encrypt-with-Hash (HD-EwH), is to fix the symmetric encryption scheme to one-time pad (so that it cannot be modified adversarially) and "share out" the randomness and message given to the public-key scheme among two independent invocations so that the necessary information needed for an iO attack is not available for any single invocation. Formally, we define $\mathsf{HD\text{-}EwH}^{\mathcal{RO}}[\mathsf{PKE}]$ as follows. Key generation creates $(sk, pk) \leftarrow_{\$} \mathsf{PKE.Kg}(1^\lambda)$ as well as four keys $\mathsf{hk}_1, \mathsf{hk}_2, \mathsf{gk}_1, \mathsf{gk}_2 \leftarrow_{\$} \{0,1\}^{\mathsf{kl}(\lambda)}$. It returns $(sk, (\mathsf{hk}_1, \mathsf{hk}_2, \mathsf{gk}_1, \mathsf{gk}_2, pk))$. An encryption of a message m consists of the following three components

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

where $\mathcal{RO}^2(\mathsf{hk}, x) := \mathcal{RO}(\mathsf{hk}, \mathcal{RO}(\mathsf{hk}, x))$. The decryption algorithm decrypts the asymmetric components of the ciphertext to get $\mathcal{RO}(\mathsf{hk}_1, pk\|m)$ and $\mathcal{RO}(\mathsf{hk}_2, pk\|m)$, hashes them under keys hk_1 and hk_2 respectively and xors them to calculate the symmetric mask, and uses this to recover the message.

In Appendix C we establish the soundness of the above transform by showing that it indeed results in a secure D-PKE in the random-oracle model. In this model, we can safely ignore the dependence on keys, and treat the four invocations of the oracle as independent (unkeyed) random oracles.

Proposition 5.1 (ROM security of HD-EwH). *Let* PKE *be an* IND-CPA*-secure public-key encryption scheme. Then, in the random-oracle model, scheme* $\mathsf{HD\text{-}EwH}^{\mathcal{RO}_1,\ldots,\mathcal{RO}_4}[\mathsf{PKE}]$ *is an* IND*-secure D-PKE scheme assuming that the probability of correctly guessing a public-key as generated by* $\mathsf{PKE.Kg}$ *on uniformly random coins is negligible.*

It is easy to seen that this transformation falls outside the realm of our generalized result in Section 4, and this opens up the possibility of its standard-model instantiability. We show that our techniques can be extended to also cover HD-EwH. We will prove this for a slight generalization of HD-EwH where a fifth random oracle is used to generate a one-time-pad key for the ciphertext component:

$$m \oplus \mathcal{RO}\left(\mathsf{fk}, (\mathcal{RO}(\mathsf{hk}_1, pk\|m), \mathcal{RO}(\mathsf{hk}_2, pk\|m))\right) .$$

If the original scheme is instantiable, then so is this scheme: we first instantiate the first four oracles, and then replace the fifth one by the hash function $\mathsf{F.Ev}(\mathsf{fk}, (x_1, x_2)) := \mathsf{H}_1.\mathsf{Ev}(\mathsf{hk}_1, x_1) \oplus \mathsf{H}_2.\mathsf{Ev}(\mathsf{hk}_2, x_2)$, where $\mathsf{fk} := (\mathsf{hk}_1, \mathsf{hk}_2)$.

As before, we construct an adversarial PKE scheme PKE^* which outputs obfuscations as part of its ciphertext. In this case, however, the scheme will output the obfuscations of two programs P1 and P2 as shown below. We denote the messages passed to the two instances of the public-key encryption scheme by x and x', reserve m for the actual message encrypted under $\mathsf{HD\text{-}EwH}^{\mathsf{H}_1,\mathsf{H}_2,\mathsf{G}_1,\mathsf{G}_2,\mathsf{F}}[\mathsf{PKE}^*]$.

```
$\mathsf{PKE}^*.\mathsf{Enc}(x, pk; r\|r_1\|r_2)$
$s \leftarrow \mathsf{PRG}(r)$
$c \leftarrow \mathsf{PKE.Enc}(pk, x; s)$
$\overline{\mathsf{P1}} \leftarrow \mathsf{iO}(\mathsf{P1}[pk, x, s](\cdot); r_1)$
$\overline{\mathsf{P2}} \leftarrow \mathsf{iO}(\mathsf{P2}[pk, x, s](\cdot); r_2)$
return $(c, \overline{\mathsf{P1}}, \overline{\mathsf{P2}})$
```

```
$\mathsf{P1}[pk, x, s](\mathsf{G}_1, \mathsf{G}_2, \mathsf{F}, c, \overline{\mathsf{P2}})$
$m \leftarrow \mathsf{UEval}(\overline{\mathsf{P2}}, (\mathsf{G}_2, \mathsf{F}, x, c))$
$r\|r_1\|r_2 \leftarrow \mathsf{G}_1(pk\|m)$
if $(\mathsf{PRG}(r) = s)$ then return $m$
return 0
```

```
$\mathsf{P2}[pk, x, s](\mathsf{G}_2, \mathsf{F}, x', c)$
$m \leftarrow c \oplus \mathsf{UEval}(\mathsf{F}, (x, x'))$
$r\|r_1\|r_2 \leftarrow \mathsf{G}_2(pk\|m)$
if $(\mathsf{PRG}(r) = s)$ then return $m$
return 0
```

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

We now show that using scheme PKE^* in the HD-EwH transform yields an insecure scheme for any choice of hash functions H_1, H_2, G_1, G_2 and F for the five random oracles. Let us see how the adversarial scheme when plugged into HD-EwH with these hash functions looks likes.

```
ALGO. $\mathsf{HD\text{-}EwH}^{\mathsf{H}_1,\mathsf{H}_2,\mathsf{G}_1,\mathsf{G}_2,\mathsf{F}}[\mathsf{PKE}](pk, m)$
$r_1\|r_1'\|r_1'' \leftarrow \mathsf{G}_1.\mathsf{Ev}(\mathsf{gk}_1, pk\|m)$,      $r_2\|r_2'\|r_2'' \leftarrow \mathsf{G}_2.\mathsf{Ev}(\mathsf{gk}_2, pk\|m)$
$s_1 \leftarrow \mathsf{PRG}(r_1)$,                                        $s_2 \leftarrow \mathsf{PRG}(r_2)$
$x_1 \leftarrow \mathsf{H}_1.\mathsf{Ev}(\mathsf{hk}_1, pk\|m)$,           $x_2 \leftarrow \mathsf{H}_2.\mathsf{Ev}(\mathsf{hk}_2, pk\|m)$
$c_1 \leftarrow \mathsf{PKE.Enc}(pk, x_1; s_1)$,                          $c_1' \leftarrow \mathsf{PKE.Enc}(pk, x_2; s_2)$
$\overline{\mathsf{P1}} \leftarrow \mathsf{iO}(\mathsf{P1}[pk, x_1, s_1](\cdot); r_1')$,     $\overline{\mathsf{P1}}' \leftarrow \mathsf{iO}(\mathsf{P1}[pk, x_2, s_2](\cdot); r_2')$
$\overline{\mathsf{P2}} \leftarrow \mathsf{iO}(\mathsf{P2}[pk, x_1, s_1](\cdot); r_1'')$,    $\overline{\mathsf{P2}}' \leftarrow \mathsf{iO}(\mathsf{P2}[pk, x_2, s_2](\cdot); r_2'')$
$c \leftarrow m \oplus \mathsf{F.Ev}(\mathsf{fk}, x_1, x_2)$
return $(c_1, \overline{\mathsf{P1}}, \overline{\mathsf{P2}}, c_1', \overline{\mathsf{P1}}', \overline{\mathsf{P2}}', c)$
```

We construct an adversary $(\mathcal{A}_1, \mathcal{A}_2)$ against the IND security of our transformed scheme as follows. The first adversary $\mathcal{A}_1$ chooses two uniformly random values $d_0, d_1 \leftarrow_{\$} \{0,1\}^{\mathsf{PKE.il}(\lambda)-1}$ and outputs messages $m_0 := d_0\|0$ and $m_1 := d_1\|1$. The second adversary $\mathcal{A}_2$ then receives as input a ciphertext $(c_1, \overline{\mathsf{P1}}, \overline{\mathsf{P2}}, c_1', \overline{\mathsf{P1}}', \overline{\mathsf{P2}}', c)$, where components $\overline{\mathsf{P1}}$ and $\overline{\mathsf{P1}}'$ are obfuscations of $\mathsf{P1}[pk, x_1, s_1]$ and $\mathsf{P1}'[pk, x_2, s_2]$ respectively, and $\overline{\mathsf{P2}}$ and $\overline{\mathsf{P2}}'$ are obfuscations of $\mathsf{P2}[pk, x_1, s_1]$ and $\mathsf{P2}'[pk, x_2, s_2]$ respectively. Adversary $\mathcal{A}_2$ then runs $\overline{\mathsf{P1}}[pk, x_1, s_1]$ on input the descriptions of functions $\mathsf{G}_1(\mathsf{gk}_1, \cdot)$ and

$\mathsf{G}_2(\mathsf{gk}_2, \cdot)$, a description of function $\mathsf{F}(\mathsf{fk}, \cdot)$, the ciphertext component c and the obfuscated program $\overline{\mathsf{P2}}'[pk, x_{,}s_2](\cdot)$. Note that the attack is running an obfuscated circuit on another obfuscated circuit here.[11] It returns the least significant bit of the output as its guess.

To see that this attack is successful, observe that the program consisting of the composition of P1 with P2 as run by the adversary is, with overwhelming probability, functionally equivalent to program P^* below.

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

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

6 Concluding Remarks

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

One promising avenue is to build schemes based on assumptions from the framework of Universal Computational Extractors (UCEs) [BHK14]. For instance, Bellare, Hoang and Keelveedhi [BHK14] show that message-locked encryption can be based on UCE[$\mathcal{S}^{\mathrm{sup}}$], that is, UCEs with statistically unpredictable sources. iO is not known to contradict statistical UCEs [BFM14]. This result, however, is not generic with respect to symmetric encryption schemes and fixes the base symmetric scheme. Very recently, Bellare and Hoang [BH14] have proposed a similar transform for D-PKE starting from lossy trapdoor functions and statistical UCEs.

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 (see also comments in Appendix A.3).

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

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

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

[ABG$^+$13] Prabhanjan Ananth, Dan Boneh, Sanjam Garg, Amit Sahai, and Mark Zhandry. Differing-inputs obfuscation and applications. Cryptology ePrint Archive, Report 2013/689, 2013. `http://eprint.iacr.org/2013/689`. (Cited on pages 6, 10, and 11.)

[AGIS14] Prabhanjan Vijendra Ananth, Divya Gupta, Yuval Ishai, and Amit Sahai. Optimizing obfuscation: Avoiding barrington's theorem. In Gail-Joon Ahn, Moti Yung, and Ninghui Li, editors, *ACM CCS 14: 21st Conference on Computer and Communications Security*, pages 646–658, Scottsdale, AZ, USA, November 3–7, 2014. ACM Press. (Cited on pages 6 and 10.)

[ARP03] Sattam S. Al-Riyami and Kenneth G. Paterson. Certificateless public key cryptography. In Chi-Sung Laih, editor, *Advances in Cryptology – ASIACRYPT 2003*, volume 2894 of *Lecture Notes in Computer Science*, pages 452–473, Taipei, Taiwan, November 30 – December 4, 2003. Springer, Berlin, Germany. (Cited on page 7.)

[BBN$^+$09] Mihir Bellare, Zvika Brakerski, Moni Naor, Thomas Ristenpart, Gil Segev, Hovav Shacham, and Scott Yilek. Hedged public-key encryption: How to protect against bad randomness. In Mitsuru Matsui, editor, *Advances in Cryptology – ASIACRYPT 2009*, volume 5912 of *Lecture Notes in Computer Science*, pages 232–249, Tokyo, Japan, December 6–10, 2009. Springer, Berlin, Germany. (Cited on pages 7 and 19.)

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

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

[BCP14] Elette Boyle, Kai-Min Chung, and Rafael Pass. On extractability obfuscation. In Yehuda Lindell, editor, *TCC 2014: 11th Theory of Cryptography Conference*, volume 8349 of *Lecture Notes in Computer Science*, pages 52–73, San Diego, CA, USA, February 24–26, 2014. Springer, Berlin, Germany. (Cited on pages 6 and 10.)

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

[BF01] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the Weil pairing. In Joe Kilian, editor, *Advances in Cryptology – CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 213–229, Santa Barbara, CA, USA, August 19–23, 2001. Springer, Berlin, Germany. (Cited on page 7.)

[BF05] Alexandra Boldyreva and Marc Fischlin. Analysis of random oracle instantiation scenarios for OAEP and other practical schemes. In Victor Shoup, editor, *Advances in Cryptology – CRYPTO 2005*, volume 3621 of *Lecture Notes in Computer Science*, pages 412–429, Santa Barbara, CA, USA, August 14–18, 2005. Springer, Berlin, Germany. (Cited on pages 7 and 24.)

[BFM14] Christina Brzuska, Pooya Farshim, and Arno Mittelbach. Indistinguishability obfuscation and UCEs: The case of computationally unpredictable sources. In Juan A. Garay and Rosario Gennaro, editors, *Advances in Cryptology – CRYPTO 2014, Part I*, volume 8616 of *Lecture Notes in Computer Science*, pages 188–205, Santa Barbara, CA, USA, August 17–21, 2014. Springer, Berlin, Germany. (Cited on pages 3, 4, 9, 14, 19, and 27.)

[BFO08] Alexandra Boldyreva, Serge Fehr, and Adam O'Neill. On notions of security for deterministic encryption, and efficient constructions without random oracles. In David Wagner, editor, *Advances in Cryptology – CRYPTO 2008*, volume 5157 of *Lecture Notes in Computer Science*, pages 335–359, Santa Barbara, CA, USA, August 17–21, 2008. Springer, Berlin, Germany. (Cited on pages 6 and 13.)

[BFOR08] Mihir Bellare, Marc Fischlin, Adam O'Neill, and Thomas Ristenpart. Deterministic encryption: Definitional equivalences and constructions without random oracles. In David Wagner, editor, *Advances in Cryptology – CRYPTO 2008*, volume 5157 of *Lecture Notes in Computer Science*, pages 360–378, Santa Barbara, CA, USA, August 17–21, 2008. Springer, Berlin, Germany. (Cited on pages 6 and 13.)

[BGI$^+$01] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. In Joe Kilian, editor, *Advances in Cryptology – CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 1–18, Santa Barbara, CA, USA, August 19–23, 2001. Springer, Berlin, Germany. (Cited on page 10.)

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

[BGK$^+$14] Boaz Barak, Sanjam Garg, Yael Tauman Kalai, Omer Paneth, and Amit Sahai. Protecting obfuscation against algebraic attacks. In Phong Q. Nguyen and Elisabeth Oswald, editors, *Advances in Cryptology – EUROCRYPT 2014*, volume 8441 of *Lecture Notes in Computer Science*, pages 221–238, Copenhagen, Denmark, May 11–15, 2014. Springer, Berlin, Germany. (Cited on pages 6 and 10.)

[BH14] Mihir Bellare and Viet Tung Hoang. UCE+LTDFs: Efficient, subversion-resistant PKE in the standard model. Cryptology ePrint Archive, Report 2014/876, 2014. `http://eprint.iacr.org/2014/876`. (Cited on pages 9 and 27.)

[BHK13a] Mihir Bellare, Viet Tung Hoang, and Sriram Keelveedhi. Instantiating random oracles via UCEs. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology – CRYPTO 2013, Part II*, volume 8043 of *Lecture Notes in Computer Science*, pages 398–415, Santa Barbara, CA, USA, August 18–22, 2013. Springer, Berlin, Germany. (Cited on pages 3, 4, and 18.)

[BHK13b] Mihir Bellare, Viet Tung Hoang, and Sriram Keelveedhi. Instantiating random oracles via UCEs. Cryptology ePrint Archive, Report 2013/424., Sep 22, 2013. (Version after initial BFM attack.) `http://eprint.iacr.org/2013/424/20130924:163256`). (Cited on pages 18 and 19.)

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

[BHK14] Mihir Bellare, Viet Tung Hoang, and Sriram Keelveedhi. Instantiating random oracles via UCEs. Cryptology ePrint Archive, Report 2013/424., May 20, 2014. (Latest version at the time of writing.) `http://eprint.iacr.org/2013/424`. (Cited on pages 7, 18, and 27.)

[BK11] Mihir Bellare and Sriram Keelveedhi. Authenticated and misuse-resistant encryption of key-dependent data. In Phillip Rogaway, editor, *Advances in Cryptology – CRYPTO 2011*, volume 6841 of *Lecture Notes in Computer Science*, pages 610–629, Santa Barbara, CA, USA, August 14–18, 2011. Springer, Berlin, Germany. (Cited on pages 7, 25, 33, 34, 35, and 37.)

[BKR13] Mihir Bellare, Sriram Keelveedhi, and Thomas Ristenpart. Message-locked encryption and secure deduplication. In Thomas Johansson and Phong Q. Nguyen, editors, *Advances in Cryptology – EUROCRYPT 2013*, volume 7881 of *Lecture Notes in Computer Science*, pages 296–312, Athens, Greece, May 26–30, 2013. Springer, Berlin, Germany. (Cited on pages 7, 25, 37, and 38.)

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

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

[BM14b] Christina Brzuska and Arno Mittelbach. Indistinguishability obfuscation versus multi-bit point obfuscation with auxiliary input. In Palash Sarkar and Tetsu Iwata, editors, *Advances in Cryptology – ASIACRYPT 2014, Part II*, volume 8874 of *Lecture Notes in Computer Science*, pages 142–161, Kaoshiung, Taiwan, R.O.C., December 7–11, 2014. Springer, Berlin, Germany. (Cited on pages 3 and 5.)

[BM14c] Christina Brzuska and Arno Mittelbach. Using indistinguishability obfuscation via UCEs. In Palash Sarkar and Tetsu Iwata, editors, *Advances in Cryptology – ASIACRYPT 2014, Part II*, volume 8874 of *Lecture Notes in Computer Science*, pages 122–141, Kaoshiung, Taiwan, R.O.C., December 7–11, 2014. Springer, Berlin, Germany. (Cited on pages 3 and 9.)

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

[BR14] Zvika Brakerski and Guy N. Rothblum. Virtual black-box obfuscation for all circuits via generic graded encoding. In Yehuda Lindell, editor, *TCC 2014: 11th Theory of Cryptography Conference*, volume 8349 of *Lecture Notes in Computer Science*, pages 1–25, San Diego, CA, USA, February 24–26, 2014. Springer, Berlin, Germany. (Cited on pages 6 and 10.)

[BST14] Mihir Bellare, Igors Stepanovs, and Stefano Tessaro. Poly-many hardcore bits for any one-way function and a framework for differing-inputs obfuscation. In Palash Sarkar and Tetsu Iwata, editors, *Advances in Cryptology – ASIACRYPT 2014, Part II*, volume 8874 of *Lecture Notes in Computer Science*, pages 102–121, Kaoshiung, Taiwan, R.O.C., December 7–11, 2014. Springer, Berlin, Germany. (Cited on pages 3, 10, and 11.)

[Can97] Ran Canetti. Towards realizing random oracles: Hash functions that hide all partial information. In Burton S. Kaliski Jr., editor, *Advances in Cryptology – CRYPTO'97*, volume 1294 of *Lecture Notes in Computer Science*, pages 455–469, Santa Barbara, CA, USA, August 17–21, 1997. Springer, Berlin, Germany. (Cited on page 24.)

[CD08] Ran Canetti and Ronny Ramzi Dakdouk. Extractable perfectly one-way functions. In Luca Aceto, Ivan Damgård, Leslie Ann Goldberg, Magnús M. Halldórsson, Anna Ingólfsdóttir, and Igor Walukiewicz, editors, *ICALP 2008: 35th International Colloquium on Automata, Languages and Programming, Part II*, volume 5126 of *Lecture Notes in Computer Science*, pages 449–460, Reykjavik, Iceland, July 7–11, 2008. Springer, Berlin, Germany. (Cited on page 4.)

[CGH98] Ran Canetti, Oded Goldreich, and Shai Halevi. The random oracle methodology, revisited (preliminary version). In *30th Annual ACM Symposium on Theory of Computing*, pages 209–218, Dallas, Texas, USA, May 23–26, 1998. ACM Press. (Cited on pages 3, 8, and 14.)

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

[CHK03] Ran Canetti, Shai Halevi, and Jonathan Katz. A forward-secure public-key encryption scheme. In Eli Biham, editor, *Advances in Cryptology – EUROCRYPT 2003*, volume 2656 of *Lecture Notes in Computer Science*, pages 255–271, Warsaw, Poland, May 4–8, 2003. Springer, Berlin, Germany. (Cited on page 7.)

[DAB$^+$02] John R. Douceur, Atul Adya, William J. Bolosky, Dan Simon, and Marvin Theimer. Reclaiming space from duplicate files in a serverless distributed file system. In *International Conference on Distributed Computing Systems*, pages 617–624, 2002. (Cited on pages 7, 25, and 38.)

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

[FO99] Eiichiro Fujisaki and Tatsuaki Okamoto. Secure integration of asymmetric and symmetric encryption schemes. In Michael J. Wiener, editor, *Advances in Cryptology – CRYPTO'99*, volume 1666 of *Lecture Notes in Computer Science*, pages 537–554, Santa Barbara, CA, USA, August 15–19, 1999. Springer, Berlin, Germany. (Cited on pages 20 and 23.)

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

[Gen03] Craig Gentry. Certificate-based encryption and the certificate revocation problem. In Eli Biham, editor, *Advances in Cryptology – EUROCRYPT 2003*, volume 2656 of *Lecture Notes in Computer Science*, pages 272–293, Warsaw, Poland, May 4–8, 2003. Springer, Berlin, Germany. (Cited on page 7.)

[GGH$^+$13] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *54th*

Annual Symposium on Foundations of Computer Science, pages 40–49, Berkeley, CA, USA, October 26–29, 2013. IEEE Computer Society Press. (Cited on pages 3, 5, and 10.)

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

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

[GKMZ14] Matthew D. Green, Jonathan Katz, Alex J. Malozemoff, and Hong-Sheng Zhou. A unified approach to idealized model separations via indistinguishability obfuscation. Cryptology ePrint Archive, Report 2014/863, 2014. `http://eprint.iacr.org/2014/863`. (Cited on page 8.)

[GLSW14] Craig Gentry, Allison Lewko, Amit Sahai, and Brent Waters. Indistinguishability obfuscation from the multilinear subgroup elimination assumption. Cryptology ePrint Archive, Report 2014/309, 2014. `http://eprint.iacr.org/2014/309`. (Cited on pages 6 and 10.)

[GS02] Craig Gentry and Alice Silverberg. Hierarchical ID-based cryptography. In Yuliang Zheng, editor, *Advances in Cryptology – ASIACRYPT 2002*, volume 2501 of *Lecture Notes in Computer Science*, pages 548–566, Queenstown, New Zealand, December 1–5, 2002. Springer, Berlin, Germany. (Cited on page 7.)

[HSW14] Susan Hohenberger, Amit Sahai, and Brent Waters. Replacing a random oracle: Full domain hash from indistinguishability obfuscation. In Phong Q. Nguyen and Elisabeth Oswald, editors, *Advances in Cryptology – EUROCRYPT 2014*, volume 8441 of *Lecture Notes in Computer Science*, pages 201–220, Copenhagen, Denmark, May 11–15, 2014. Springer, Berlin, Germany. (Cited on page 3.)

[KLW14] Venkata Koppula, Allison Bishop Lewko, and Brent Waters. Indistinguishability obfuscation for turing machines with unbounded memory. Cryptology ePrint Archive, Report 2014/925, 2014. `http://eprint.iacr.org/2014/925`. (Cited on pages 6 and 11.)

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

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

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

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

[PST14] Rafael Pass, Karn Seth, and Sidharth Telang. Indistinguishability obfuscation from semantically-secure multilinear encodings. In Juan A. Garay and Rosario Gennaro, editors, *Advances in Cryptology – CRYPTO 2014, Part I*, volume 8616 of *Lecture Notes in Computer Science*, pages 500–517, Santa Barbara, CA, USA, August 17–21, 2014. Springer, Berlin, Germany. (Cited on pages 6 and 10.)

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

[RSV13] Ananth Raghunathan, Gil Segev, and Salil P. Vadhan. Deterministic public-key encryption for adaptively chosen plaintext distributions. In Thomas Johansson and Phong Q. Nguyen, editors, *Advances in Cryptology – EUROCRYPT 2013*, volume 7881 of *Lecture Notes in Computer Science*, pages 93–110, Athens, Greece, May 26–30, 2013. Springer, Berlin, Germany. (Cited on pages 8, 14, and 23.)

[SW05] Amit Sahai and Brent R. Waters. Fuzzy identity-based encryption. In Ronald Cramer, editor, *Advances in Cryptology – EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 457–473, Aarhus, Denmark, May 22–26, 2005. Springer, Berlin, Germany. (Cited on page 7.)

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

[Wic13] Daniel Wichs. Barriers in cryptography with weak, correlated and leaky sources. In Robert D. Kleinberg, editor, *ITCS 2013: 4th Innovations in Theoretical Computer Science*, pages 111–126, Berkeley, CA, USA, January 9–12, 2013. Association for Computing Machinery. (Cited on page 7.)

A Key-Dependent Security

In this section, we consider uninstantiability results for generic ROM constructions with the purpose of achieving security in the presence of key-dependent data. The transformations that we consider in this and the next appendix apply to symmetric encryption schemes, and show that our techniques are not confined to the asymmetric setting.

A.1 Definitions

Symmetric encryption. In line with [BK11] we consider an extended notion of (deterministic) symmetric encryption schemes that encompasses nonces. For simplicity, we do not introduce the additional header field introduced in [BK11]. A symmetric encryption scheme $\mathsf{SE} = (\mathsf{SE.Kg}, \mathsf{SE.Enc}, \mathsf{SE.Dec})$ is defined as follows. The probabilistic key-generation algorithm $\mathsf{SE.Kg}(1^\lambda)$ generates keys $k \in \{0,1\}^{\mathsf{SE.kl}(\lambda)}$.

The encryption algorithm $\mathsf{SE.Enc}$ takes as input a key k, a nonce $N \in \{0,1\}^{\mathsf{SE.nl}(\lambda)}$, and a message $m \in \{0,1\}^{\mathsf{SE.il}(\lambda)}$ and outputs a ciphertext c. In general it is up to the application to ensure transport of the nonce, but for simplicity we assume that the nonce is part of the ciphertext. The decryption algorithm $\mathsf{SE.Dec}$ on input a key k, a nonce N, and a ciphertext c outputs a message m or a special symbol $\perp$. As usual we require the scheme to be correct, that is, for all choices of the key k, message m and nonce N it holds that $\mathsf{SE.Dec}(k, N, \mathsf{SE.Enc}(k, N, m)) = m$.

```
$\mathrm{KIAE}^{\mathcal{A}}_{\mathsf{SE}}(\lambda)$
$S \leftarrow \emptyset$
$b \leftarrow\$ \{0,1\}$
$k \leftarrow\$ \mathsf{SE.Kg}(1^\lambda)$
$b' \leftarrow\$ \mathcal{A}^{\textsc{Enc},\textsc{Dec}}(1^\lambda)$
return $(b = b')$
```

```
$\textsc{Enc}(m)$
$N \leftarrow\$ \{0,1\}^{\mathsf{SE.nl}(\lambda)}$
$c_1 \leftarrow \mathsf{SE.Enc}(k, N, m)$
$m' \leftarrow\$ \{0,1\}^{|m|}$
$c_0 \leftarrow \mathsf{SE.Enc}(k, N, m')$
$S \leftarrow S \cup \{(N, c_b)\}$
return $(N, c_b)$
```

```
$\textsc{Dec}(N, c)$
if $(N, c) \in S$ then return $\perp$
$m \leftarrow \perp$
if $b = 1$ then
    $m \leftarrow \mathsf{SE.Dec}(k, N, c)$
return $(m \neq \perp)$
```

Figure 6: The KIAE security game for authenticated encryption. Note that we consider an authenticated setting and thus give the adversary access to a decryption oracle with checks ciphertext for well-formedness.

Key-independent authenticated encryption (KIAE). We consider symmetric encryption in the authenticated encryption setting as defined via the key-independent authenticated encryption (KIAE) security game in Figure 6. The game chooses a key k and gives the adversary access to two oracles Enc and Dec which allow an adversary $\mathcal{A}$ to encrypt messages of its choice and to test whether (mauled) ciphertexts are well-formed. Depending on a hidden bit b the decrypt oracle always returns $\perp$ (if $b = 0$) or if $b = 1$, it checks whether the supplied ciphertext is "fresh," decrypts it, and responds with a Boolean value indicating if decryption succeeded.[12] For the encryption oracle, according to the hidden bit b, either an encryption of the supplied message with a fresh chosen random nonce—the nonce is outside the control of the adversary—or an encryption of a random plaintext (of appropriate length) is returned. We define the advantage of an adversary $\mathcal{A}$ in the KIAE game against a symmetric encryption scheme SE by

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

We call a symmetric encryption scheme SE KAIE secure if the above advantage of any PPT adversary $\mathcal{A}$ is negligible. Note that [BK11] considers a stronger variant where the encryption oracle responds with random strings rather than random encryptions. We denote this stronger notion by \$-KIAE.

Key-dependent authenticated encryption (KDAE). We consider a strengthening of KIAE to the *key-dependent* setting, and allow the adversary to obtain encryptions of messages which are derived from the key in an adversarially specified manner. Observe that the KDAE game is parameterized by w which specifies the number of keys in the system. Note also that in its encryption queries the adversary specifies a key index as well as a function ϕ that is applied to the keys in the system to obtain a plaintext m. Following [BK11] we formally define the KDAE security of a symmetric encryption scheme in Figure 7, and let

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

We call a scheme KDAE secure if the above advantage for any PPT adversary $\mathcal{A}$ in game KDAE is negligible. The \$-KDAE notion of [BK11] strengthens this game further to one where the encryption oracle returns random strings when $b = 0$.

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

```
$\mathrm{KDAE}^{\mathcal{A}}_{\mathsf{SE},w}(\lambda)$
for $j = 1, \ldots, w$ do
    $k_j \leftarrow\$ \mathsf{SE.Kg}(1^\lambda)$
    $S_j \leftarrow \emptyset$
$b \leftarrow\$ \{0,1\}$
$b' \leftarrow\$ \mathcal{A}^{\textsc{Enc},\textsc{Dec}}(1^\lambda)$
return $(b = b')$
```

```
$\textsc{Enc}(j, \phi)$
$m \leftarrow \phi(k_1, \ldots, k_w)$
$N \leftarrow\$ \{0,1\}^{\mathsf{SE.nl}(\lambda)}$
$c_1 \leftarrow \mathsf{SE.Enc}(k_j, N, m)$
$m' \leftarrow\$ \{0,1\}^{|m|}$
$c_0 \leftarrow \mathsf{SE.Enc}(k_j, N, m')$
$S_j \leftarrow S_j \cup \{(N, c_b)\}$
return $(N, c_b)$
```

```
$\textsc{Dec}(j, N, c)$
if $(N, c) \in S_j$ then return $\perp$
$m \leftarrow \perp$
if $b = 1$ then
    $m \leftarrow \mathsf{SE.Dec}(k_j, N, c)$
return $(m \neq \perp)$
```

Figure 7: The KDAE security game with procedures Enc and Dec.

A.2 Uninstantiability of RHtE

The Randomized-Hash-then-Encrypt transform. Bellare and Keelveedhi [BK11] introduce the *Randomized-Hash-then-Encrypt* transform (RHtE) as a means to convert a KIAE symmetric encryption scheme to one which is KDAE secure in the random-oracle model. This transformation $\mathsf{RHtE}^{\mathcal{RO}}[\mathsf{SE}]$ takes a deterministic (and nonceless) symmetric encryption scheme SE as input. In the keyed random-oracle model a key is chosen during setup and assumed to be publicly available. (When this key is kept private as part of the key-generation process our attack also applies; see the comment after the proof.) The transformed encryption operation first hashes the nonce N together with key k to obtain a "one-time" key k' which is then used to encrypt message m. The transformed encryption and decryption operations are shown below.

```
$\mathsf{RHtE}^{\mathcal{RO}}[\mathsf{SE}].\mathsf{Enc}(\mathsf{hk}, k, N, m)$
$k' \leftarrow \mathcal{RO}(\mathsf{hk}, N \| k)$
$c \leftarrow \mathsf{SE.Enc}(k', m)$
return $(N, c)$
```

```
$\mathsf{RHtE}^{\mathcal{RO}}[\mathsf{SE}].\mathsf{Dec}(\mathsf{hk}, k, N, c)$
$k' \leftarrow \mathcal{RO}(\mathsf{hk}, N \| k)$
$m \leftarrow \mathsf{SE.Dec}(k', c)$
return $m$
```

Bellare and Keelveedhi (BK) [BK11, Theorem 4.1] show that the RHtE transform yields a \$-KDAE-secure scheme, when starting from a one-time \$-KIAE-secure symmetric encryption scheme. Loosely speaking, one-time security is sufficient as in the RHtE transform a fresh key for the symmetric scheme SE is chosen for every new encryption.

Uninstantiability of RHtE. BK require the base scheme to meet the slightly stronger "\$ variants" of KIAE and KDAE where ciphertexts are required to be indistinguishable from a random strings (rather than from encryption of random strings). It is not clear whether BK's result (in the RO model) can be modified to also hold for the case the base scheme is only assumed to be KIAE and KDAE (rather than \$-KIAE and \$-KDAE). We will elaborate on this distinction further in Section A.3. Here we show that such a modified result would also suffer from uninstantiability results. We show how to tweak any one-time KIAE-secure scheme SE to one which is still one-time KIAE secure, but which yields an insecure scheme when used within RHtE for standard-model hash functions. As before our result comes in two flavors assuming indistinguishability obfuscation for Turing machines and circuits respectively.

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

respect to this weaker decryption oracle can be shown to be equivalent to one with respect to a full decryption oracle that returns the message. Also observe that a full decryption oracle in the KDAE setting can lead to trivial attacks.

Proof. We start by noting that we cannot use our generalized result as we are now considering a transformation of (deterministic) symmetric encryption schemes rather than a transformation of public-key encryption schemes. The overall proof strategy, however, will be similar.

Let SE be a one-time KIAE-secure symmetric encryption scheme with a key-generation algorithm that samples a key uniformly at random from $\{0,1\}^{\mathsf{SE.kl}(\lambda)}$. Let PRG be a pseudorandom generator with outputs in $\{0,1\}^{\mathsf{SE.kl}(\lambda)}$. The encryption routine generates a ciphertext that consists of three components $(c, \overline{\mathsf{P}}, \tau)$ where c is the ciphertext generated via the underlying scheme SE, program $\overline{\mathsf{P}}$ is an indistinguishability obfuscation of program $\mathsf{P}[s, N, m]$ defined on the right below. Since we need to ensure that the modified scheme retains its (one-time) security even in the presence of a decryption oracle, an unforgeable MAC tag τ of $(c, \overline{\mathsf{P}})$ is appended to the outputs. For technical reasons that we will explain at the end of Section A.3 we use a pseudorandom function PRF instead of a MAC.[13] Decryption first checks the consistency of the tag and then calls $\mathsf{SE.Dec}$ on the (adapted) secret key and c to recover m.

```
ALGO. $\mathsf{SE}^*.\mathsf{Enc}(k', N, m)$
parse $k_0 \| k_1 \| r \leftarrow k'$
$s \leftarrow \mathsf{PRG}(k_0)$
$c \leftarrow \mathsf{SE.Enc}(s, N, m)$
$\overline{\mathsf{P}} \leftarrow \mathsf{iO}(\mathsf{P}[s, N, m](\cdot); r)$
$\tau \leftarrow \mathsf{PRF}(k_1, c \| \overline{\mathsf{P}} \| N)$
return $(N, c, \overline{\mathsf{P}}, \tau)$
```

```
ALGO. $\mathsf{SE}^*.\mathsf{Dec}(k', N, c, \overline{\mathsf{P}}, \tau)$
parse $k_0 \| k_1 \| r \leftarrow k'$
$\tau' \leftarrow \mathsf{PRF}(k_1, c \| \overline{\mathsf{P}} \| N)$
if $(\tau \neq \tau')$ then return $\perp$
$s \leftarrow \mathsf{PRG}(k_0)$
$m \leftarrow \mathsf{SE.Dec}(s, N, c)$
return $m$
```

```
PROG. $\mathsf{P}[s, N, m](\mathsf{H}(\mathsf{hk}, \cdot))$
$k_0 \| k_1 \| r \leftarrow \mathsf{UEval}(\mathsf{H}(\mathsf{hk}, \cdot), N \| m)$
$s' \leftarrow \mathsf{PRG}(k_0)$
if $(s' = s)$ then return $m$
return 0
```

To reduce the KIAE security of the adapted scheme to that of the underlying scheme, we first note that the decryption oracle can be simulated by returning $\perp$ for all queries down to the hardness of predicting PRF values. Indeed, an adversary would need to come up with a new ciphertext $(c, \overline{\mathsf{P}}, N)$ and a valid tag τ to detect any inconsistent simulation as otherwise, by the rules of the game, the oracle returns $\perp$. Such a query can be directly used to break the pseudorandomness of the PRF. In order to show that the scheme is KIAE secure, we follow steps similar to those in the proof of Theorem 3.1, and show that the obfuscations of P are indistinguishable from those of the zero program and hence inclusion of $\overline{\mathsf{P}}$ does not have any adverse effects on KIAE security.

We now show how to attack the KDAE security of RHtE when it is instantiated with scheme SE^* and an arbitrary hash function H. We construct an adversary $\mathcal{A}$ that uses a single key $(w = 1)$ and needs a single encryption query (i.e., it breaks one-time KDAE security). Adversary $\mathcal{A}$ sets ϕ to the identity function id and calls $\textsc{Enc}(1, id)$ to receive $(N, c, \overline{\mathsf{P}}, \tau)$. It interprets $\overline{\mathsf{P}}$ as a program, and runs it on (an encoding of) the hash function H (with key hk hard-coded in). Note that by our setup assumption the adversary is aware of the hash key hk. When the program returns a message m, adversary $\mathcal{A}$ interprets this value as a key k and attempts to decrypt the challenge ciphertext. If it decrypts successfully, $\mathcal{A}$ returns 1. Otherwise, it returns 0.

Assume that $b = 1$ and let k be the key of the transformed scheme. Since the adversary sets $\phi := id$, message $m = k$ will be encrypted. The key k' that is used by SE^* is derived as $k' \leftarrow \mathsf{H.Ev}(\mathsf{hk}, N \| k)$ for a nonce N and parses to $k_0 \| k_1 \| r$. This means that the second component of the ciphertext contains an obfuscation of the program $\mathsf{P}[\mathsf{PRG}(k_0), N, m = k]$. Hence $k_0 \| k_1 \| r$ will be correctly recovered under P and the check performed by P will passes. In other words, the KDM query has allowed the adversary to embed the key k in P so that the check passes when it is run on the code of the hash function used in instantiation. This means that the adversary recovers a non-zero value with overwhelming probability.[14]

[13] Note that a PRF with sufficiently long output is an unforgeable MAC with the additional property that the tags look random.

[14] Alternatively we could define P to output 1 instead of m when the check passes. The current formulation of P, however, leads to a total break of the scheme and recovers the key.

On the other hand, when $b = 0$, the message m embedded in P will be random, and the probability that the check passes is negligible. Hence, the adversary outputs a correct guess for b with an overwhelming probability. □

Private hash keys. The above proof relies on the public availability of the hash key. An alternative transform is to include the hash key generation as part of the scheme's key. That is a secret key (k, hk) is generated as $k \leftarrow\!\!\$\ \mathsf{SE.Kg}(1^\lambda)$ and $\mathsf{hk} \leftarrow\!\!\$\ \mathsf{H.Kg}(1^\lambda)$. Our attack can be also mounted against this transform. When the adversary $\mathcal{A}$ sets ϕ to the identity map, the output message becomes (k, hk). All that remains is to adapt the program to first extract the hash key and then plug it into the description of the hash function without the hash key hard-coded in.

A.3 $-KDAE security

The KIAE and KDAE definitions considered by Bellare and Keelveedhi [BK11] require the ciphertexts in the base scheme to be indistinguishable from *random strings* rather than encryptions of random strings (i.e., the base scheme is $-KIAE secure). This raises the question if our results also apply when starting with a $-KDAE-secure scheme.

The uninstantiability result crucially depends on including the obfuscation of a circuit into the ciphertext. Such an obfuscation is, however, heavily structured and it is not clear if indistinguishability obfuscation schemes exist that have an obfuscation which is indistinguishable from a uniformly random bit string. One straightforward distinguishing attack against any obfuscation scheme would be to simply execute the code and check the outputs. In particular, it cannot be the case that an obfuscation of both the zero circuit and the one circuit look like random strings. However, there could still exist an indistinguishability obfuscation scheme with the extra property that the obfuscations of the zero circuit look random. In this case, the symmetric encryption scheme that we constructed would also meet the stronger $-KIAE notion. We leave the study of such real-or-random indistinguishability obfuscators for future work.

Despite this, if the base scheme is $-KIAE secure, the tweaked scheme can be shown to have *simulatable* ciphertexts in the sense that it is possible to extend the ciphertexts to those which look indistinguishable from real tweaked ciphertexts. This explains our choice of using a PRF instead of a MAC for ciphertext integrity. An inspection of BK's proof [BK11, page 25] reveals that this weaker property is sufficient for their proof to go through. Put differently, our uninstantiability result shows that one cannot weaken the $-KIAE assumption to simulatability and still hope for generic standard-model instantiability, although this can be done in the ROM.

B Message-Locked Encryption

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

BKR propose several security notions for MLEs. One is called PRV-CDA and is similar to the IND notion for D-PKE. In place of the public key, the public parameters of the scheme π are now outside the reach of the first phase of the attack. These parameters are used by encryption to derive the encryption key. In order to rule out trivial re-encryption attacks, each component of the message vectors, similarly IND, are required to have high min-entropy. See Figure 8 (left) for the details of the

```
$\mathrm{PRV\text{-}CDA}^{\mathcal{A}_1,\mathcal{A}_2}_{\mathsf{MLE}}(\lambda)$
$\pi \leftarrow\!\!\$\ \mathsf{MLE.Pg}(\lambda)$
$b \leftarrow\!\!\$\ \{0,1\}$
$(\mathbf{m}_0, \mathbf{m}_1) \leftarrow\!\!\$\ \mathcal{A}_1(1^\lambda)$
for $i = 1 \ldots |\mathbf{m}_0|$ do
    $k \leftarrow \mathsf{MLE.Kg}(\pi, \mathbf{m}_b[i])$
    $\mathbf{c}[i] \leftarrow \mathsf{MLE.Enc}(k, \mathbf{m}_b[i])$
$b' \leftarrow\!\!\$\ \mathcal{A}_2(\pi, \mathbf{c})$
return $(b = b')$
```

```
$\mathrm{PRV\$\text{-}CDA}^{\mathcal{A}_1,\mathcal{A}_2}_{\mathsf{MLE}}(\lambda)$
$\pi \leftarrow\!\!\$\ \mathsf{MLE.Pg}(\lambda)$
$b \leftarrow\!\!\$\ \{0,1\}$
$\mathbf{m} \leftarrow\!\!\$\ \mathcal{A}_1(1^\lambda)$
for $i = 1 \ldots |\mathbf{m}|$ do
    $k \leftarrow \mathsf{MLE.Kg}(\pi, \mathbf{m}[i])$
    $\mathbf{c}_1[i] \leftarrow \mathsf{MLE.Enc}(k, \mathbf{m}[i])$
    $\mathbf{c}_0[i] \leftarrow\!\!\$\ \{0,1\}^{|\mathbf{c}_1[i]|}$
$b' \leftarrow\!\!\$\ \mathcal{A}_2(\pi, \mathbf{c}_b)$
return $(b = b')$
```

Figure 8: The MLE security games PRV-CDA and PRV\$-CDA of BKR [BKR13]. Here we have given slightly simpler variants where the adversaries are not allowed to share state via a zeroth-stage adversary. Once again, this only strengthens our negative results.

game. The PRV\$-CDA strengthens PRV-CDA by requiring the ciphertexts to look indistinguishable from random strings.

Convergent encryption. One transformation which is formally studied by BKR and originates in the work of Douceur et al. [DAB$^+$02] is *convergent encryption* (CE). The CE transform constructs a message-locked encryption scheme from a one-time-secure deterministic symmetric encryption scheme SE in the (keyed) random-oracle model. Note that, in contrast to the KDM transform of Appendix A, the SE scheme only takes a key and a message and does not use an additional nonce. Parameters π are chosen during setup as a uniformly random string. The encryption key k for a message m and public parameters π is computed as $k \leftarrow \mathcal{RO}(\mathsf{hk}, \pi \| m)$. Note that the range of the hash function must be a subset of the scheme's key space. The encryption and decryption algorithms of SE are used directly without change in the new scheme. Under one-time key recovery[15] and a slightly stronger variant of one-time IND-CPA security (which requieres ciphertexts are indistinguishable from random strings), the CE transformation is proved PRV-CDA secure in the random-oracle model [BKR13].

Attacking PRV-CDA. We now show that this transform yields an insecure schemes when starting from an adversarial one-time key-recovery and one-time IND-CPA-secure scheme SE^*. Our generalized result presented in Section 4 does not apply here as, similarly to the KDM case (Appendix A), we are considering a transformation of (deterministic) symmetric encryption schemes.

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

We next present our one-time IND-CPA-secure symmetric encryption SE^* that breaks CE^H. The idea, as before, is to append an obfuscated circuited to the ciphertext. Since in MLE the scheme is not randomized, we obtain the necessary randomness for obfuscation directly from the secret key. Given a one-time IND-CPA-secure and one-time key-recovery-secure scheme SE with uniform keys in $\{0,1\}^{\mathsf{SE.kl}(\lambda)}$ and a pseudorandom generator PRG of appropriate stretch we construct SE^* as follows. Key generation is left unchanged, and encryption is shown below. Decryption simply ignores the second component of the ciphertext and decrypts the first.

[15] One-time key recovery requires that in presence of at most one ciphertext no adversary can guess the key with non-negligible probability. The reason that it suffices to consider one-time security is that for each encryption a fresh key is chosen.

```
$\text{PRV-CDA}^{\mathcal{A}_1,\mathcal{A}_2}_{\mathsf{CE^H[SE^*]}}(\lambda)$
$\pi \leftarrow\$ \mathsf{MLE.Pg}(\lambda)$
$b \leftarrow\$ \{0,1\}$
$\mathsf{hk} \leftarrow\$ \mathsf{HKg}(1^\lambda)$
$(m_0, m_1) \leftarrow\$$
    $\mathcal{A}_1(1^\lambda)$
        $x_0 \leftarrow\$ \{0,1\}^{\mathsf{PKE.il}(\lambda)-1}$
        $x_1 \leftarrow\$ \{0,1\}^{\mathsf{PKE.il}(\lambda)-1}$
        return $(x_0\|0, x_1\|1)$
$(c, \overline{\mathsf{P}}) \leftarrow$
    $\mathsf{CE^H[SE^*].Enc}(\mathsf{hk}, m_b)$
        $k\|k' \leftarrow \mathsf{H.Ev}(\mathsf{hk}, \pi\|m_b)$
        $(c, \overline{\mathsf{P}}) \leftarrow$
            $\mathsf{SE^*.Enc}(k\|k', m_b)$
                $s \leftarrow \mathsf{PRG}(k)$
                $c \leftarrow \mathsf{SE.Enc}(s, m_b)$
                $\overline{\mathsf{P}} \leftarrow \mathsf{iO}(\mathsf{P}[m_b, s](\cdot,\cdot); k')$
                return $(c, \overline{\mathsf{P}})$
        return $(c, \overline{\mathsf{P}})$
$b' \leftarrow$
    $\mathcal{A}_2(\pi, \mathsf{hk}, (c, \overline{\mathsf{P}}))$
        $m_b \leftarrow \overline{\mathsf{P}}(\mathsf{H}(\mathsf{hk},\cdot), \pi)$
        $b' \leftarrow m_b[|m_b|]$
        return $b'$
return $(b = b')$
```

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

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

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

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

Proving that the above modifications do not affect the one-time IND-CPA and one-time key-recovery security properties of SE^* is analogous to that presented for D-PKEs. Since we consider one-time security, each encryption is run on a freshly sampled random key which can take the role of randomness used in the proof for D-PKEs. The attack against the PRV-CDA security of the transformed scheme also works analogously to the EwH case. There is, however, a minor modification that needs to be taken care of as the symmetric component of the scheme does not have access to the public parameters π of the MLE scheme (and in particular cannot hard-code them into the obfuscated circuit). We address this issue by considering a circuit which takes the public parameters π as an additional input. Note that according to the rules of the PRV-CDA game, π will be available to the second-stage adversary. We present the pseudocode outline of the attack in Figure 9.

Strengthening the base scheme. In the adapted scheme SE^* we somewhat abused the fact that the base scheme only needs to be one-time secure, and derived arbitrary randomness from the key. It is an interesting question whether our attack can also be mounted if the base scheme is required to meet the standard IND-CPA notion. Removing the one-time restriction is intricate as one would have to introduce fresh randomness to avoid trivial attacks. Another avenue to circumvent uninstantiability is to demand the stronger requirement that ciphertexts look random as we discuss in Appendix A.3.

C ROM Security of HD-EwH

We start by giving a high-level idea of the proof of Proposition 5.1. Our goal is to convert a given IND adversary $(\mathcal{A}_1, \mathcal{A}_2)$ into an IND-CPA adversary $\mathcal{B}$. In the reduction we need to simulate the random oracles $\mathcal{RO}_1, \ldots, \mathcal{RO}_4$. Adversary $\mathcal{A}_2$ expects as input ciphertexts of the form

$$\mathsf{PKE.Enc}\left(pk, \mathcal{RO}_1(pk\|m); \mathcal{RO}_3(pk\|m)\right), \mathsf{PKE.Enc}\left(pk, \mathcal{RO}_2(pk\|m); \mathcal{RO}_4(pk\|m)\right),$$

$$m \oplus \mathcal{RO}_1^2(pk\|m) \oplus \mathcal{RO}_2^2(pk\|m) \ .$$

Without having made any random-oracle queries, the first two values are distributed as encryptions of two uniformly random values and the last component is distributed identically to a uniformly random string. In particular the message m is information-theoretically hidden.

Let us consider an IND adversary $(\mathcal{A}_1, \mathcal{A}_2)$ and a ciphertext c consisting of two asymmetric parts and one symmetric part. In order for an adversary to have any advantage, $\mathcal{A}_2$ must learn some information about bit b and hence information about which of two messages was encrypted. The two asymmetric ciphertext parts are encryptions of $\mathcal{RO}_1(pk\|m)$ and $\mathcal{RO}_2(pk\|m)$ which, intuitively, hide m unless m is already known. The symmetric part information-theoretically hides m unless the adversary queries $\mathcal{RO}_1$ on $\mathcal{RO}_1(pk\|m)$ (and $\mathcal{RO}_2$ on $\mathcal{RO}_2(pk\|m)$). Our IND-CPA adversary $\mathcal{B}$ will completely ignore $\mathcal{A}_1$ (except for learning the length of the message vectors) and create two random message (vectors) m_0 and m_1 which it outputs to receive $c \leftarrow\!\$\ \mathsf{PKE.Enc}(pk, m_b)$ for a random bit b. It will use c as the first part of ciphertexts for $\mathcal{A}_2$ and simulate the second asymmetric part by encrypting a random value, and the symmetric part by simply choosing a uniformly random value of appropriate length. It will then monitor random-oracle queries of $\mathcal{A}_2$ and check if any of them matches m_b, in which case it stops and outputs b. Note that the simulated environment for $\mathcal{A}_2$ is perfect down to the high min-entropy requirement of $\mathcal{A}_1$ and the unpredictability of the public key.

We next make this intuition formal.

Proof (of Proposition 5.1). Assume there exists a ROM adversary $(\mathcal{A}_1, \mathcal{A}_2)$ against the IND security of $\mathsf{HD\text{-}EwH}^{\mathcal{RO}_1,\ldots,\mathcal{RO}_4}[\mathsf{PKE}]$. For the sake of simplicity of presentation, we assume that $\mathcal{A}_1$ outputs two messages instead of two message vectors. Based on $(\mathcal{A}_1, \mathcal{A}_2)$ we construct an adversary $\mathcal{B}$ against the IND-CPA security of PKE as follows. (See Figure 10 for $\mathcal{B}$'s pseudocode.)

Adversary $\mathcal{B}$ gets as input a public key pk. (It discards $\mathcal{A}_1$ as by assumption it outputs two messages instead of message vectors.)[16] It samples messages

$$m_0 \leftarrow\!\$\ \{0,1\}^{\mathsf{PKE.il}(\lambda)} \quad \text{and} \quad m_1 \leftarrow\!\$\ \{0,1\}^{\mathsf{PKE.il}(\lambda)}$$

and outputs these. Note that $\mathsf{PKE.il}(\lambda)$ is assumed to match the hash digest lengths $\mathsf{ol}_1(\lambda)$ and $\mathsf{ol}_2(\lambda)$.

The second phase of $\mathcal{B}$ receives as input a ciphertext c which is either an encryption of message m_0 or of m_1. Adversary $\mathcal{B}$ constructs two additional value c' and c''. Value c' is constructed as an encryption of a uniformly random message under key pk and c'' is sampled uniformly at random. Values c and c' correspond to the asymmetric parts of the ciphertext expected by $\mathcal{A}_2$ and c'' corresponds to the the symmetric part of the ciphertext.

Adversary $\mathcal{B}$ then calls adversary $\mathcal{A}_2$ on input $(pk, (c, c', c''))$ and answers its random-oracle queries via lazy sampling. If there is an oracle query q to $\mathcal{RO}_1$ by $\mathcal{A}_2$ such that there $q = m_d$ for some $d \in \{0,1\}$, then $\mathcal{B}$ stops and outputs d. If such a query does not occur, $\mathcal{B}$ outputs a random bit.

[16] Otherwise it would need to run $\mathcal{A}_1$ in order to learn the length of the vector and its equality pattern and match that in its simulation.

```
$\text{IND-CPA}^{\mathcal{B}}_{\mathsf{PKE}}(\lambda)$
$b \leftarrow\$ \{0,1\}$
$(sk, pk) \leftarrow\$ \mathsf{PKE.Kg}(1^\lambda)$
$(m_0, m_1) \leftarrow\$$ $\mathcal{B}(1^\lambda, pk)$
    $b_{\mathcal{B}} \leftarrow\$ \{0,1\}$
    $m_0 \leftarrow\$ \{0,1\}^{\mathsf{PKE.il}(\lambda)}$
    $m_1 \leftarrow\$ \{0,1\}^{\mathsf{PKE.il}(\lambda)}$
    return $(m_0, m_1)$
$c \leftarrow\$ \mathsf{PKE.Enc}(pk, m_b)$
$b' \leftarrow\$$ $\mathcal{B}(\mathbf{c})$
    $x \leftarrow\$ \{0,1\}^{\mathsf{PKE.il}(\lambda)}$
    $c' \leftarrow\$ \mathsf{PKE.Enc}(pk, x)$
    $c'' \leftarrow\$ \{0,1\}^{\mathsf{PKE.il}(\lambda)}$
    $b'_{\mathcal{B}} \leftarrow\$ \mathcal{A}_2^{\text{Ro}_1,\text{Ro}_2,\text{Ro}_3,\text{Ro}_4}(pk, (c, c', c''))$
    return $b_{\mathcal{B}}$
return $(b = b')$
```

```
$\text{Ro}_j(q)$
if $T_j[q] = \bot$ then
    $T_j[q] \leftarrow\$ \{0,1\}^{\mathsf{PKE.il}(\lambda)}$
if $(j = 1 \ \wedge \ m_d = q)$ then
    $b_{\mathcal{B}} \leftarrow d$
return $T_j[q]$
```

Figure 10: Adversary $\mathcal{B}$ against the IND-CPA security of scheme PKE in proof of Proposition 5.1

Analysis. We need to argue that (1) $\mathcal{B}$ wins if a query $q = m_d$ occurs, and that (2) $\mathcal{A}_2$ makes such a query with non-negligible probability. For (1) consider that m_0 and m_1 are chosen uniformly at random and ciphertext c only contains information about m_d but no information about m_{1-d}. For (2) consider the simulation from the point of view of adversary $\mathcal{A}_2$. Adversary $\mathcal{A}_2$ expects to receive as input the public key pk and a ciphertext

$$\mathsf{PKE.Enc}(pk, \mathcal{RO}_1(pk\|m_b); \mathcal{RO}_3(pk\|m_b)), \mathsf{PKE.Enc}(pk, \mathcal{RO}_2(pk\|m_b); \mathcal{RO}_4(pk\|m_b)),$$
$$m_b \oplus \mathcal{RO}_1^2(pk\|m_b) \oplus \mathcal{RO}_2^2(pk\|m_b) \ .$$

In the simulation of $\mathcal{B}$ it gets two encryptions of uniformly random values and a uniformly random value. The simulation of the symmetric part is perfect unless $\mathcal{A}_2$ queries $\mathcal{RO}_1(pk\|m_b)$ to $\mathcal{RO}_1$ and $\mathcal{RO}_2(pk\|m_b)$ to $\mathcal{RO}_2$. Furthermore, c'' hides m_b unless these random-oracle queries occur. Similarly, the asymmetric parts of the ciphertext are simulated perfectly since the message distribution output by $\mathcal{A}_1$ is assumed to have high min-entropy and hence

$$\Pr\left[m_b = \mathcal{P}^{\mathcal{RO}_1,\cdots}(pk, \mathcal{RO}_1(pk\|m_b)) : (sk, pk) \leftarrow\$ \mathsf{PKE.Kg}(1^\lambda); b \leftarrow\$ \{0,1\}; (m_0, m_1) \leftarrow\$ \mathcal{A}_1(1^\lambda)\right]$$

is negligible for any algorithm $\mathcal{P}$ that makes at most polynomially many random-oracle queries.

Thus, in order for adversary $\mathcal{A}_2$ to win, it must unmask m_b from the symmetric part of the ciphertext, which in turn requires it to query $\mathcal{RO}_1$ on $\mathcal{RO}_1(pk\|m_b)$. Assuming that $(\mathcal{A}_1, \mathcal{A}_2)$ has non-negligible advantage implies that $\mathcal{A}_2$ makes the said query with non-negligible probability. Hence $\mathcal{B}$ also has non-negligible advantage in the IND-CPA game. □