Indistinguishability Obfuscation and UCEs: The Case of Computationally Unpredictable Sources

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

Tel Aviv University, Israel
Royal Holloway, University of London, UK
Darmstadt University of Technology, Germany
brzuska@post.tau.ac.il   pooya.farshim@rhul.ac.uk
arno.mittelbach@cased.de

Abstract. Random oracles are powerful cryptographic objects. They facilitate the security proofs of an impressive number of practical cryptosystems ranging from KDM-secure and deterministic encryption to point-function obfuscation and many more. However, due to an uninstantiability result of Canetti, Goldreich, and Halevi (STOC 1998) random oracles have become somewhat controversial. Recently, Bellare, Hoang, and Keelveedhi (BHK; CRYPTO 2013 and ePrint 2013/424, August 2013) introduced a new abstraction called Universal Computational Extractors (UCEs), and showed that they suffice to securely replace random oracles in a number of prominent applications, including all those mentioned above, without suffering from the aforementioned uninstantiability result. This, however, leaves open the question of constructing UCEs in the standard model.

We show that the existence of indistinguishability obfuscation (iO) implies (non-black-box) attacks on all the definitions that BHK proposed within their UCE framework in the original version of their paper, in the sense that no concrete hash function can satisfy them. We also show that this limitation can be overcome, to some extent, by restraining the class of admissible adversaries via a *statistical* notion of unpredictability. Following our attack, BHK (ePrint 2013/424, September 2013), independently adopted this approach in their work.

In the updated version of their paper, BHK (ePrint 2013/424, September 2013) also introduce two other novel source classes, called *bounded parallel sources* and *split sources*, which aim at recovering the computational applications of UCEs that fall outside the statistical fix. These notions keep to a computational notion of unpredictability, but impose structural restrictions on the adversary so that our original iO attack no longer applies. We extend our attack to show that indistinguishability obfuscation is sufficient to also break the UCE security of any hash function against bounded parallel sources. Towards this goal, we use the *randomized encodings* paradigm of Applebaum, Ishai, and Kushilevitz (STOC 2004) to parallelize the obfuscated circuit used in our attack, so that it can be computed by a bounded parallel source whose second stage consists of constant-depth circuits. BHK, in the latest version of their paper (ePrint 2013/424, May 2014), have subsequently replace bounded parallel sources with new source classes. We conclude by discussing the composability and feasibility of hash functions secure against split sources.

Keywords Randomized encodings, obfuscation, UCE, random oracle

1 Introduction

Since their formal introduction in the seminal paper of Bellare and Rogaway [13], random oracles have found extensive use across a wide spectrum of cryptographic protocols. Their versatility has lead researchers to seek for a unified formalization of their useful properties, hoping that such a definition could be eventually realized. Canetti, Goldreich, and Halevi [20] proposed such a definition, but somewhat disappointingly, also proved a negative result which ruled out instantiations of random oracles in *arbitrary* (perhaps artificial) cryptographic protocols by *any* keyed hash functions. This negative result was subsequently extended in a number of works [35,25,22,32,7,21].

UCE security. Bellare, Hoang, and Keelvedhi (BHK) [8,9,10,12] [1] revisited the above question and formulated an attractive new security notion called *Universal Computational Extractor* (UCE). They were able to apply their framework to an interesting and diverse set of security goals, which included among other things, security under key-dependent attacks, security under related-key attacks, simultaneous hardcore bits, point-function obfuscation, garbling schemes, proofs of storage, and deterministic encryption. Recently, Matsuda and Hanaoka [33] used UCEs to also build CCA-secure public-key encryption schemes.

The UCE framework comes in two versions: a single-key version (UCE) and a multi-key version (mUCE). For a keyed hash function H, single-key UCE security is defined via a two-stage security game consisting of algorithms S and D, called the *source* and the *distinguisher*, respectively. In the first stage, the source is given access to an oracle $\textsc{Hash}$ that, depending on a challenge bit b, implements either a random oracle or the concrete hash function with a randomly chosen key hk. The source terminates with some leakage L, which is then communicated together with hk to the distinguisher D. The distinguisher's goal is to guess the bit b, i.e., guess whether the source interacted with the random oracle or the hash function. The UCE advantage of the pair (S, D) is defined as the probability of returning the correct answer scaled away from one-half. (The stronger multi-key version is defined analogously by introducing $\textsc{Hash}$ oracles for multiple keys and providing the keys together with leakage to the distinguisher.) We summarize this interaction schematically in Figure 1, and give the pseudocode in Figure 2. We refer the reader to the original work for an excellent philosophical perspective on this framework.

Without any restrictions UCE security cannot be achieved: the source can simply leak one of its oracle queries together with the corresponding answer to the distinguisher, which then can locally compute the hash value on the queried

[1] Citation [8] refers to the CRYPTO 2013 proceedings version, [9] refers to its full version on Cryptology ePrint Archive from August 2013 prior to communicating our basic iO attack (presented in this paper), and [10] refers to the version from September/October 2013, and [12] refers to the latest version from May 2014.

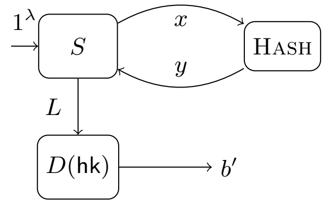

Fig. 1: Schematic of the UCE game.

```
MAIN $\mathrm{UCE}_{\mathsf{H}}^{S,D}(\lambda)$
$b \leftarrow\$ \{0,1\}$;  $\mathsf{hk} \leftarrow\$ \mathsf{H.Kg}(1^\lambda)$
$L \leftarrow\$ S^{\mathrm{HASH}}(1^\lambda)$
$b' \leftarrow\$ D(1^\lambda, \mathsf{hk}, L)$
return $(b = b')$
```

```
HASH$(x)$
if $T[x] = \bot$ then
  if $b = 1$ then
    $T[x] \leftarrow \mathsf{H.Ev}(1^\lambda, \mathsf{hk}, x)$
  else $T[x] \leftarrow\$ \{0,1\}^{\mathsf{H.ol}(\lambda)}$
return $T[x]$
```

Fig. 2: Pseudocode for the UCE game. Here $\mathsf{H.ol}(\lambda)$ is a function which specifies the length of hash values.

point (the distinguisher knows the hash key) and compare it to the leaked hash value. Thus, the source needs to be somehow restricted, and this restriction forms the actual UCE definition: for a source class $\mathcal{S}$ we denote the UCE assumption with sources restricted to $\mathcal{S}$ by UCE[$\mathcal{S}$]. Prior to our work, BHK proposed two source classes via *unpredictability* and *reset security* conditions, which in turn gave rise to two notions called UCE1 and UCE2, respectively.

The UCE1 notion [8,9] is defined using an unpredictability game which requires that when the source is run with a *random oracle*, its leakage does not *computationally* reveal any of its queries. This is formalized by requiring that the probability that an efficient predictor P can guess a query of S when given L is negligible. Such a source is then called unpredictable, and leads to the following definition of UCE1 security: a hash function is UCE1 secure if the advantage of all efficient, unpredictable sources S, and all efficient distinguishers D in the UCE game is negligible. The stronger notion of UCE2 security is defined analogously by requiring that the source satisfies the weaker requirement of reset security.

Following our obfuscation-based attack (that we describe next and that we communicated to BHK in August 2013 [11]), the UCE1 and UCE2 notions were revised in [10] and additional restrictions on sources were imposed. We will be discussing these shortly, after presenting our first attack.

An obfuscation-based attack on UCE1*.* Our first attack, described in Section 3, targets the original UCE notions UCE1 and UCE2, and is based on a recent breakthrough in the construction of obfuscation schemes. Garg et al. [24] give a candidate construction for the so-called notion of indistinguishability obfuscation [6] based on intractability assumptions related to multi-linear maps. Our attack shows that any UCE1 construction would need to falsify one of these assumptions. Put differently, if indistinguishability obfuscation exists, then UCE1 security (and hence also the stronger UCE2 security) cannot be achieved.

Roughly speaking, a secure indistinguishability obfuscation (iO) scheme assures that the obfuscations of any two circuits that implement the same function are computationally indistinguishable. Our attack uses this primitive as follows. The source picks a random point x, and queries it to HASH to get y. It then prepares an iO of the Boolean circuit $(\mathsf{H}(\cdot, x) = y)$, and leaks it to the distinguisher as L. The distinguisher now plugs the hash key hk into this obfuscated circuit and returns whatever the circuit outputs. It is easy to see that the distinguisher recovers the challenge bit correctly with an overwhelming probability. What is less clear, however, is whether or not the source is unpredictable. Recall that the unpredictability game operates with respect to a random oracle. Let us now assume, for simplicity, that $|\mathsf{hk}| < |y|/2$ (we will not need to rely on this assumption in our full attack). For any x, there are at most $2^{|\mathsf{hk}|}$ possible values for $\mathsf{H}(\mathsf{hk}, x)$, and a random y would be one of them with probability at most $2^{|\mathsf{hk}|}/2^{|y|} < 2^{-|y|/2}$, which is negligible. Consequently, the obfuscated circuit implements the constant *zero* function with overwhelming probability. This allows us to apply the security of the obfuscator to conclude the attack: the obfuscated circuit does not leak any more information about x than the zero function would, and since x was chosen randomly, it remains hidden from the view of any efficient predicator.

Salvaging UCE. Assuming the existence of indistinguishability obfuscation, we ask to what extent UCE can be salvaged. That is, do there exist other UCE assumptions that allow recovering (some of) the originally presented applications? We partially salvage UCEs by modifying the unpredictability condition and letting the predictor run in *unbounded* time. This statistical notion of unpredictability restricts the class of admissible sources such that the source implementing the iO attack falls outside it: an unbounded predictor can reverse-engineer the *computationally* secure obfuscator. This modification is validated by the work of Goldwasser and Rothblum [26] who show that a statistical analogue of iO is impossible unless the polynomial hierarchy collapses to its second level. As we discuss in Section 3.2, a large number of interesting applications (such as KDM and RKA security) survives under this definition.

After communicating our attack, BHK independently suggested the statistical patch [10]. In the revised version of their paper [10], they recast their proofs of security to rely only on statistical unpredictability for all applications where this is possible. We refer to [10] for details on the applications that can be salvaged by statistical UCE1. As mentioned earlier, not all applications can be salvaged by statistical unpredictability. Hence, BHK also present two additional UCE notions based on computational unpredictability, which together with the statistical patch allowed them to fully recover their original set of applications in light of the aforementioned iO attack. We discuss these next.

Computational UCE. Some applications discussed in [8,9], specifically hardcore functions, deterministic public-key encryption (D-PKE), message-locked encryption (MLE), and OAEP rely on computational unpredictability in an intrinsic way; that is, the reduction only works if the predictor is bound to run in polynomial time. For instance, the source presented in [8,9] for D-PKEs produces leakage which contains encryptions of messages that have been sent to the HASH oracle. An unbounded predictor can easily decipher the ciphertexts and predict HASH queries of the source.

Following the above attack, in the updated version of their paper, BHK [10] propose two novel UCE notions by imposing additional restrictions on the way the source operates, while keeping the original *computational* unpredictability game. The goal here is that these restrictions are sufficiently strong to circumvent our attack, but weak enough so that successful security reductions can be established.

To recover D-PKEs, MLEs, and OAEP, BHK propose a new UCE assumption based on computational unpredictability restricted to so-called *bounded parallel sources*. Such a source splits into two stages S_0 and S_1. In the first phase, algorithm S_0 prepares a vector of strings. In the second phase, independent instances of S_1 for each entry in the previously prepared vector are run in parallel. Each instance gets access to the HASH oracle and their combined outputs make up the final leakage. To circumvent our attack two restrictions on S_1 are imposed: its runtime and number of HASH queries (per instance). The idea here is that computing the obfuscation of a hash function is "too costly," and hence the attack cannot be mounted.

In Section 4 we show that this refined notion still falls prey to a similar, but somewhat more complex attack. The idea is to split the iO attack into two stages consisting of a high-complexity first stage and a parallelizable second stage. To this end, we use the powerful *randomized encodings* paradigm of Applebaum, Ishai, and Kushilevitz [2] to bring down the complexity of the second stage of the attack. The randomized encoding $\hat{f}(x;r)$ of $f(x)$ is simply an encoding of $f(x)$ such that a decoder dec can retrieve the original value $f(x)$ from it, i.e., $\mathsf{dec}(\hat{f}(x;r)) = f(x)$. In addition, a randomized encoding specifies an efficient simulator Sim such that for all x the distributions $\hat{f}(x;r)$ over uniformly chosen r and $\mathsf{Sim}(f(x))$ are computationally indistinguishable. These properties combined allow us to show that we can adapt our original attack such that the source does not leak the obfuscated circuit but rather a randomized encoding of it. This alone, however, is still not enough for an attack with the restrictions of bounded parallel sources. Finally, we utilize a special form of *decomposable* randomized encodings [31] to realize an attack. Such encodings have the property that each output bit of $\hat{f}(x;r)$ depends on at most a *single* bit of x (but possibly on the entire string r). The randomized encoding of Applebaum, Ishai, and Kushilevitz [3] is decomposable and supports all functions in $\mathcal{P}/poly$. We show how to use such an encoding scheme to split the computation of the encoding into two phases: a complex first preprocessing phase which does not depend on the actual input and a very simple second stage which can be parallelized and where each parallel instance essentially only has to drop one of two bits. We show that this second stage (which will correspond to S_1) can be implemented by constant-depth circuits consisting only of very few gates. This application of decomposable randomized encodings could be of interest also in other scenarios where efficiently computing an encoding is important and preprocessing is possible. In the latest version of their paper [12] BHK has removed bounded parallel sources and replaced them by new source classes to recover the original applications.

While bounded parallel sources suffice to also recover simultaneous hardcore functions, BHK propose a second, simpler UCE assumption based on *split*

```
MAIN UCE$_{\mathsf{H}}^{S,D}(\lambda)$
$b \leftarrow\$ \{0,1\}$;  $\mathsf{hk} \leftarrow\$ \mathsf{H.Kg}(\lambda)$
$L \leftarrow\$ S^{\text{HASH}}(1^\lambda)$
$b' \leftarrow\$ D(1^\lambda, \mathsf{hk}, L)$
return $(b = b')$

HASH$(x)$
if $T[x] = \bot$ then
  if $b = 1$ then
    $T[x] \leftarrow \mathsf{H.Ev}(1^\lambda, \mathsf{hk}, x)$
  else $T[x] \leftarrow\$ \{0,1\}^{\mathsf{H.ol}(\lambda)}$
return $T[x]$
```

```
MAIN Pred$_S^P(\lambda)$
$\mathsf{done} \leftarrow$ false;  $Q \leftarrow \emptyset$
$L \leftarrow\$ S^{\text{HASH}}(1^\lambda)$;  $\mathsf{done} \leftarrow$ true
$Q' \leftarrow\$ P^{\text{HASH}}(1^\lambda, L)$
return $(Q \cap Q' \neq \emptyset)$

HASH$(x)$
if $\mathsf{done}$ = false then
  $Q \leftarrow Q \cup \{x\}$
if $T[x] = \bot$ then
  $T[x] \leftarrow\$ \{0,1\}^{\mathsf{H.ol}(\lambda)}$
return $T[x]$
```

```
MAIN Reset$_S^R(\lambda)$
$\mathsf{Dom} \leftarrow \emptyset$;  $L \leftarrow\$ S^{\text{HASH}}(1^\lambda)$;
$b \leftarrow\$ \{0,1\}$
if $b = 0$ then
  for $x \in \mathsf{Dom}$ do
    $T[x] \leftarrow\$ \{0,1\}^{\mathsf{H.ol}(\lambda)}$
$b' \leftarrow\$ R^{\text{HASH}}(1^\lambda, L)$;
return $(b' = b)$

HASH$(x)$
$\mathsf{Dom} \leftarrow \mathsf{Dom} \cup \{x\}$
if $T[x] = \bot$ then
  $T[x] \leftarrow\$ \{0,1\}^{\mathsf{H.ol}(\lambda)}$
return $T[x]$
```

Fig. 3: The UCE security game together with the unpredictability and reset-security games.

sources. A split source consists of two parts S_0 and S_1, in which each part independently contributes to the leakage sent to the distinguisher. The idea is that none of these sub-sources gets direct access to the HASH oracle. Rather, algorithm S_0 defines the queries (without access to any hash values) and algorithm S_1 gets to see the hash values but not the queries. As for our attack, note that the associated source needs to know both the query x and its hash value $y \leftarrow \text{HASH}(x)$ in order to compute the circuit $(\mathsf{H}(\cdot, x) = y)$. We discuss split sources in a larger context and present necessary conditions for a hash function to achieve split-source UCE security in the full version of this work [19]. For example, we show that in order to prove the security of a hash function H, one needs to show that the function that maps x to the obfuscation of the circuit $\mathsf{H}(\cdot, x)$ must not be one way. We also discuss intricacies regarding composition of such functions with one-way permutations, and show that such a composition does not harm standard notions such as collision resistance, pseudorandomness (and indeed statistical UCE1 security) but provably fails for split-source security. We present this discussion in the full version.

To conclude, although UCEs strengthen our confidence in the security of many practical schemes in the random-oracle model, our attacks highlight the need for a thorough assessment of definitional choices that can be made within the UCE framework. This assessment, in addition to instantiability questions, should also include studying concrete instantiations of UCEs such as the SHA family [34] in HMAC mode, as suggested by BHK [8].

2 Preliminaries

Notation. We denote by $\lambda \in \mathbb{N}$ the security parameter, which is implicitly given to all algorithms (if not explicitly stated so) in the unary representation 1^λ. By $\{0,1\}^\ell$ we denote the set of all bit-strings of length ℓ, and by $\{0,1\}^*$ the set of all bit-strings of finite length. For two strings $x_1, x_2 \in \{0,1\}^*$ their concatenation is written as $x_1\|x_2$. The length of x is denoted by $|x|$ and $x[i]$ is the i-th bit of x. For a finite set X, we denote the action of sampling x uniformly at random from X by $x \leftarrow_\$ X$, and denote the cardinality of X by $|X|$. Algorithms are assumed to be randomized, unless otherwise stated. We call an algorithm efficient or PPT if it runs in time polynomial in the security parameter. By $y \leftarrow \mathcal{A}(x;r)$ we denote that y was output by algorithm $\mathcal{A}$ on input x and randomness r. If $\mathcal{A}$ is randomized and no randomness is specified, then we assume that $\mathcal{A}$ is run with freshly sampled uniform random coins, and write this is as $y \leftarrow_\$ \mathcal{A}(x)$. We often refer to algorithms, or tuples of algorithms, as adversaries. We say a function $\mathrm{negl}(\lambda)$ is negligible if $|\mathrm{negl}(\lambda)| \in \lambda^{-|\omega(1)|}$. In this paper we deploy the game-playing framework of Bellare and Rogaway [14] with the augmented game procedures described in [36].

Syntax of hash functions. In line with [8], we consider the following formalization of hash functions. A function family H is a five tuple of PPT algorithms $(\mathsf{H.Kg}, \mathsf{H.Ev}, \mathsf{H.kl}, \mathsf{H.il}, \mathsf{H.ol})$ as follows. The algorithms $\mathsf{H.kl}$, $\mathsf{H.il}$, and $\mathsf{H.ol}$ are deterministic and on input 1^λ define the key length, input length, and output lengths, respectively. (We have adopted the simplified notion from [8] here.) The key generation algorithm $\mathsf{H.Kg}$ gets the security parameter 1^λ as input and outputs a key $\mathsf{hk} \in \{0,1\}^{\mathsf{H.kl}(\lambda)}$. The deterministic evaluation algorithm $\mathsf{H.Ev}$ takes as input the security parameter 1^λ, a key hk, a message $x \in \{0,1\}^{\mathsf{H.il}(\lambda)}$ and generates a hash value $\mathsf{H.Ev}(1^\lambda, \mathsf{hk}, x) \in \{0,1\}^{\mathsf{H.ol}(\lambda)}$.

UCE *game.* Let $\mathsf{H} = (\mathsf{H.Kg}, \mathsf{H.Ev}, \mathsf{H.kl}, \mathsf{H.il}, \mathsf{H.ol})$ be a hash function and (S, D) be a pair of PPT algorithms. We define the UCE advantage of (S, D) against H through

$$\mathsf{Adv}^{\mathsf{uce}}_{\mathsf{H},S,D}(\lambda) := 2 \cdot \Pr\left[\mathrm{UCE}^{S,D}_{\mathsf{H}}(\lambda)\right] - 1 \ ,$$

where game $\mathrm{UCE}^{S,D}_{\mathsf{H}}(\lambda)$ is shown in Figure 3 on the left.

Unpredictability. A source S is called *computationally unpredictable* if the advantage of any PPT predictor P defined by

$$\mathsf{Adv}^{\mathsf{pred}}_{S,P}(\lambda) := \Pr\left[\mathsf{Pred}^{P}_{S}(\lambda)\right]$$

is negligible, where game $\mathsf{Pred}^{P}_{S}(\lambda)$ is shown in Figure 3 in the middle. We denote the class of all computationally unpredictable sources by $\mathcal{S}^{\mathrm{cup}}$.

UCE *security.* We say a hash function H is UCE1 secure if for all computationally unpredictable PPT sources S and all PPT distinguishers D the advantage $\mathsf{Adv}^{\mathsf{uce}}_{\mathsf{H},S,D}(\lambda)$ is negligible. In the later version of their paper [10], BHK refer to UCE1 as UCE$[\mathcal{S}^{\mathrm{cup}}]$. BHK introduce a stronger version called UCE2 which is based on the reset-security game $\mathsf{Reset}^{R}_{S}(1^\lambda)$ shown in Figure 3 on the right. We refer the reader to [9] for the details, but note here that UCE2 security implies UCE1 security and, thus, any attack on UCE1 also applies to UCE2.

We discuss the revised UCE assumptions introduced in [10], namely those for *bounded parallel sources* and *split sources*, in Section 4 and in the full version [19], respectively.

Indistinguishability obfuscation. Roughly speaking, an indistinguishability obfuscation (iO) scheme ensures that the obfuscations of any two functionally equivalent circuits are computationally indistinguishable. Indistinguishability obfuscation was originally proposed by Barak et al. [6] as a potential weakening of virtual-black-box obfuscation. We recall the definition from [24]. A PPT algorithm iO is called an *indistinguishability obfuscator* for a circuit class $\{\mathcal{C}_\lambda\}_{\lambda\in\mathbb{N}}$ if the following conditions are satisfied:

- Correctness. For all security parameters $\lambda \in \mathbb{N}$, for all $C \in \mathcal{C}_\lambda$, and for all inputs x we have that
$$\Pr\left[C'(x) = C(x) : C' \leftarrow_{\$} \mathsf{iO}(1^\lambda, C)\right] = 1 \ .$$
- Security. For any PPT distinguisher D, for all pairs of circuits $C_0, C_1 \in \mathcal{C}_\lambda$ such that $C_0(x) = C_1(x)$ on all inputs x the following distinguishing advantage is negligible:
$$\mathsf{Adv}^{\mathsf{io}}_{\mathsf{iO},D,C_0,C_1}(\lambda) := \Pr\left[D(\mathsf{iO}(1^\lambda, C_1)) = 1\right] - \Pr\left[D(\mathsf{iO}(1^\lambda, C_0)) = 1\right] \ .$$

With their recent candidate construction for indistinguishability obfuscation, Garg et al. [24] have revived interest in the study of obfuscation schemes (see, for example, [37,16,28,17,18,23,5,15] and the references therein). Garg et al. prove that under an intractability assumption related to multi-linear maps their construction yields an indistinguishability obfuscator for all circuits in $\mathcal{NC}^1$. Additionally, assuming a perfectly correct fully homomorphic encryption scheme and a perfectly sound non-interactive witness-indistinguishable proof system, they also show how their obfuscation scheme can be bootstrapped to support any polynomial-size circuit. In a recent work, Barak et al. [5] have further simplified the construction and showed that it is secure against all generic multi-linear attacks.

3 UCE1 and UCE2 Security

In this section we formalize our iO attack on the UCE1 (and hence the stronger UCE2) security of *any* concrete hash function. We also propose a fix to these notions which avoids the attack while still being applicable to a number of cryptosystems.

3.1 The iO attack

Our attack uses an indistinguishability obfuscation scheme in a black-box way, but is non-black-box as it relies on the code of the hash function for obfuscation. (Therefore, the attack does not contradict the positive feasibility of BHK in the random-oracle model.) We stress that the complexity of running our attack, although high, is polynomial, and will benefit from future advances in the construction of iO schemes.

Theorem 1 (UCE1 **infeasibility**). *If indistinguishability obfuscation exists, then* UCE1 *security cannot be achieved in the standard model.*

We now present a sketch of the proof and defer the full proof to the full version [19].

Proof (sketch). Let H be a UCE1-secure hash function family. Let us assume for now that $\mathsf{H.ol}(\lambda) \geq 2 \cdot \mathsf{H.kl}(\lambda)$, that is, the output length of the hash function is at least twice the size of a hash key. (We will be dropping this condition shortly.) Define a source S which generates a random value $x \leftarrow_{\$} \{0,1\}^{\mathsf{H.il}(\lambda)}$ and computes $y \leftarrow \textsc{Hash}(x)$. It then constructs the Boolean circuit

$$C_{\lambda,\mathsf{H},x,y}(\cdot) := (\mathsf{H.Ev}(1^{\lambda}, \cdot, x) = y)\,,$$

that returns 1 on input hk if, and only if, $\mathsf{H.Ev}(1^{\lambda}, \mathsf{hk}, x)$ equals y. The source S passes on an encoding of circuit $C_{\lambda,\mathsf{H},x,y}(\cdot)$ as leakage L to the distinguisher. We will later use obfuscation to ensure that x is not leaked by the encoding of $C_{\lambda,\mathsf{H},x,y}(\cdot)$ (this is needed for unpredictability). The distinguisher D recovers circuit $C_{\lambda,\mathsf{H},x,y}(\cdot)$ from the leakage L, and computes $b' \leftarrow C_{\lambda,\mathsf{H},x,y}(\mathsf{hk})$ using the given hash key hk, and returns b'. The UCE1 adversary (S, D) has advantage $1 - 2^{-\mathsf{H.ol}(\lambda)}$: when the source is run with oracle access to $\mathsf{H.Ev}(1^{\lambda}, \mathsf{hk}, \cdot)$, the circuit always returns 1. When S interacts with a random oracle, y coincides with $\mathsf{H.Ev}(1^{\lambda}, \mathsf{hk}, x)$ with probability $2^{-\mathsf{H.ol}(\lambda)}$.

Now let iO be an indistinguishability obfuscator. Instead of leaking circuit $C_{\lambda,\mathsf{H},x,y}(\cdot)$, we let S compute an obfuscation of the circuit and output $L \leftarrow_{\$} \mathsf{iO}(C_{\lambda,\mathsf{H},x,y}(\cdot))$. By the correctness property of the obfuscator, distinguisher D, as before, has an overwhelming advantage in guessing the challenge bit correctly. It remains to show that the adapted source S is unpredictable.

In the unpredictability game $\mathsf{Pred}_S^P(\lambda)$ oracle $\textsc{Hash}$ is always implemented by a random oracle. Thus, with high probability the circuit $C_{\lambda,\mathsf{H},x,y}(\cdot)$ is the constant zero circuit: for any $x \in \{0,1\}^n$, there are at most $2^{|\mathsf{H.kl}((\lambda))|}$ possible values for $\mathsf{H.Ev}(\mathsf{hk}, x)$, and a random y would be one of the image values with probability at most $2^{|\mathsf{H.kl}(\lambda)|}2^{-\ell}$ which by assumption is less than $2^{-\mathsf{H.ol}(\lambda)/2}$. Now, to see that the source S is unpredictable note that the zero function and $C_{\lambda,\mathsf{H},x,y}(\cdot)$ are functionally equivalent. This means that an indistinguishability obfuscation of $C_{\lambda,\mathsf{H},x,y}$ will not leak any more information about x than the zero function would. Since x was chosen randomly, it remains hidden from the view of any PPT predicator P.

It remains to argue how we can drop the requirement on the size of hash keys. For this note that we can simply choose a t such that $t \geq 2 \cdot \lceil \mathsf{H.kl}(\lambda)/\mathsf{H.ol}(\lambda) \rceil$ and let the source leak an obfuscation of the circuit $(\mathsf{H.Ev}(1^{\lambda}, \cdot, x_1) = y_1 \wedge \cdots \wedge \mathsf{H.Ev}(1^{\lambda}, \cdot, x_t) = y_t)$. □

In the above proof, we relied on the source being able to make multiple queries to its hash oracle. Bellare, Hoang, and Keelveedhi [11] point out that the theorem can be extended to a single-query source by applying a pseudorandom generator to the output of the hash function. This result is noteworthy as several applications only require the source to make a single query.

3.2 Statistical unpredictability

The iO attack immediately gives rise to the following question: can the UCE1 and/or UCE2 notions be somehow patched so that they avoid the attack while maintaining (part of) their wide applicability? Fortunately, we show that this is indeed the case. We start by observing that the security guarantee of the indistinguishability obfuscator is only computational. Consequently, the attack can be directly ruled out by demanding the source to be *statistically* unpredictable, i.e., by letting a potential predictor run in unbounded time (but still impose polynomial query complexity). More formally, we say a source S is *statistically unpredictable* if the advantage of any (possibly unbounded) predictor P with polynomial query complexity in the $\mathsf{Pred}_S^P(\lambda)$ game shown in Figure 3 (middle) is negligible. Statistical UCE2 security can be defined analogously, where we let the reset distinguisher run in unbounded time and only place a polynomial bound on the number of its queries.

The above definition, in turn, leads to the following two questions: (1) Is a statistically secure variant of indistinguishability obfuscation possible? (2) Are there any application scenarios which only rely on this weaker property? Goldwasser and Rothblum [26] provide a negative answer to the first question by showing that the existence of a statistically secure iO scheme implies the collapse of the polynomial hierarchy to its second level. This impossibility result reinforces our confidence in the soundness of the above definition. For the second question, recall that the unpredictability game is always defined with respect to a random oracle, and hence statistical unpredictability may be (non-trivially) achievable. Indeed, consider a source which samples a random point x, queries it to its oracle, and leaks the result to the distinguisher. It is easy to see that this source is statistically unpredictable as a random oracle is one-way against unbounded adversaries. Indeed, many of the cryptosystems considered by BHK admit security proofs with sources that essentially take this simple form [8,9]. We present a brief discussion of these in the full version of this work.

After we communicated our attack [11], BHK in the revised version of their paper [10] also independently suggested the statistical notion of unpredictability. They denote by $\mathcal{S}^{\mathrm{sup}}$ the class of all statistically unpredictable sources and recast their proofs of the above to use UCE$[\mathcal{S}^{\mathrm{sup}}]$. We refer to [10] for details on the applications that can be salvaged with statistical UCE1 aka UCE$[\mathcal{S}^{\mathrm{sup}}]$ (resp. statistical UCE2 aka UCE$[\mathcal{S}^{\mathrm{srs}}]$).

We end this section by noting that for the hardcore predicate, BR93 encryption, D-PKE, MLE and OAEP application scenarios discussed in [8,9], the leakage contains auxiliary information related to a query x that only computationally hides x (e.g., it might contain a one-way image $f(x)$, or an encryption of x). Consequently, an unbounded predictor might well be able to guess the point x, and in these cases our statistical patch is no longer useful. Despite this, we observe that UCE-secure hash functions with regard to statistical unpredictability are hardcore for highly non-injective one-way functions. (The proof is essentially equivalent to that in [9] and relies on the fact that any (even an unbounded) predictor cannot recover the *exact* query if the preimage space is super-polynomially large.)

4 Bounded Parallel Sources

In version [10] of their paper, BHK introduce novel UCE-type security notions to recover applications where statistical unpredictability is of no help. The main idea behind these new UCE assumptions is that, in order to keep the unpredictability condition computational, the source needs to operate in a restricted way so that the iO attack cannot be mounted any longer.

A new restricted source class that BHK introduce to recover the deterministic public-key encryption (D-PKE), message-locked encryption (MLE), and OAEP applications is that of *bounded parallel sources*. In parallel sources the source splits into two parts S_0 and S_1 as follows. The first part of the source S_0 does not get oracle access to HASH, and simply outputs some preliminary leakage L_0 and a vector $\mathbf{L}'$ of arbitrary bit strings. For each entry in $\mathbf{L}'$ an independent instance of the second part of the source S_1 is run. This can be done in parallel as the several invocations do not share any coins or state. Instance i of S_1 is given $\mathbf{L}'[i]$ as input which then produces leakage $\mathbf{L}[i]$. As opposed to S_0, the second part S_1 of parallel sources has oracle access to HASH. The final leakage of the source $S := \mathsf{Prl}[S_0, S_1]$ is set to be $L := (L_0, \mathbf{L})$. The details of a parallel source $S = \mathsf{Prl}[S_0, S_1]$ are given in Figure 4 on the left.

Without any further restrictions, parallel sources are as powerful as regular sources: simply ignore S_0 and let a single S_1 generate the entire leakage. Thus, in order to circumvent the iO attack, further restrictions are necessary. To this end, BHK restrict the resources of S_0 and S_1 via polynomials τ, σ, and q as follows: (1) the running time (circuit size) of each invocation of S_1 is at most $\tau(\cdot)$; (2) each invocation of S_1 makes at most $q(\cdot)$ oracle queries; and (3) the length of initial leakage L_0 output by S_0 is at most $\sigma(\cdot)$. BHK then consider the class $\mathcal{S}^{\mathrm{prl}}_{\tau,\sigma,q}$ consisting of all parallel sources satisfying these bounds, and define UCE for computationally unpredictable, bounded parallel sources by considering $\mathrm{UCE}[\mathcal{S}^{\mathrm{cup}} \cap \mathcal{S}^{\mathrm{prl}}_{\tau,\sigma,q}]$.

For their results on D-PKE and MLE schemes, the parameters τ, σ, and q need to be fine-tuned according to the underlying encryption scheme. More precisely, BHK set q to 1 (each instance of S_1 makes a single hash query), σ to the size of a key-pair (0 in the case of MLEs), and τ to the runtime of the encryption operation plus the input and key sizes of the encryption scheme. It is easily seen that our basic attack does not fall into this class as long as the computation of the obfuscated circuit takes longer than what is granted by τ.

```
Prl SOURCE $S^{\text{HASH}}(1^\lambda)$
    $(L_0, \mathbf{L}') \leftarrow\!\!\$\; S_0(1^\lambda)$
    for $i = 1, \ldots, |\mathbf{L}'|$ do
        $\mathbf{L}[i] \leftarrow\!\!\$\; S_1^{\text{HASH}}(1^\lambda, \mathbf{L}'[i])$
    $L \leftarrow (L_0, \mathbf{L})$
    return $L$
```

```
Splt SOURCE $S^{\text{HASH}}(1^\lambda)$
    $(L_0, \mathbf{x}) \leftarrow\!\!\$\; S_0(1^\lambda)$
    for $i = 1, \ldots, |\mathbf{x}|$ do
        $\mathbf{y}[i] \leftarrow\!\!\$\; \text{HASH}(\mathbf{x}[i])$
    $L_1 \leftarrow\!\!\$\; S_1(1^\lambda, \mathbf{y}); L \leftarrow (L_0, L_1)$
    return $L$
```

Fig. 4: The parallel source $S = \mathsf{Prl}[S_0, S_1]$ on the left and the split source $S = \mathsf{Splt}[S_0, S_1]$ on the right as defined in the updated version of [10]. In both cases the source consists of two parts S_0 and S_1 that jointly generate leakage L. For split sources neither part gets direct oracle access to HASH. For parallel sources additional restrictions on the runtime and the number of queries of S_1, and the length of leakage L_0 are imposed. Note that the invocations of S_1 are parallelizable and independent of one another.

In choosing the parameters for bounded parallel sources, one has to strike a delicate balance between the complexity of obfuscating a hash function and the cost of encryption (resp. the application in question). Indeed, suppose that a bounded parallel source assumption with parameters as above is used to prove an MLE scheme secure in the standard model. Now if the complexity of the encryption scheme is high (e.g., because it is implemented based on iO [37] or because it includes (artificial) redundant code), then the assumption can be broken by the iO attack, as described in the previous section. Similarly, if one could reduce the complexity of obfuscating the hash function, an attack would become feasible. However, considering the current state of research, obfuscation is a very costly operation and thus, intuitively, computing the obfuscation of a hash function should be harder than encrypting a message.

Interestingly, as we show in this section, it is the *parallel* complexity of obfuscating a hash function (after a possibly complex preprocessing phase) that matters for the attack, and we can show that the latter can lie in a complexity class which is dramatically below that of computing the obfuscation of the hash function. More precisely, we show how to combine our iO attack with the *randomized encodings* of Applebaum, Ishai, and Kushilevitz [2] to split the attack into two stages such that the second stage is highly parallelizable. Before describing our attack, let us briefly recall the notion of randomized encodings.

4.1 Randomized encodings

Randomized encodings allow one to substantially reduce the complexity of computing a function f by instead computing an *encoding* of it. This technique was first introduced by Ishai and Kushilevitz [29,30] in the context of multi-party computation and has since found many applications [2,3,31,27,4,1]. The formalization of randomized encodings that we use here is due to Applebaum, Ishai, and Kushilevitz (AIK) [2] and is adapted to the setting of perfect correctness and computational privacy. Informally, we say that $\hat{f}(x; r)$ is a randomized encoding of some function $f(x)$ if (1) given $\hat{f}(x;r)$ one can efficiently recover function value $f(x)$, and (2) given $f(x)$, one can efficiently sample from the distribution $\hat{f}(x;r)$ induced by uniformly choosing r.

More precisely, a randomized encoding scheme RE consists of three efficient algorithms $(\mathsf{enc}, \mathsf{dec}, \mathsf{Sim})$ as follows: (1) a probabilistic encoding algorithm enc which on input a security parameter 1^λ, a circuit computing $f_\lambda : \{0,1\}^{n(\lambda)} \to \{0,1\}^{\ell(\lambda)}$ (of size polynomial in λ) and an $x \in \{0,1\}^{n(\lambda)}$ outputs an encoding $z \in \{0,1\}^{s(\lambda)}$; (2) a deterministic decoder algorithm dec which on input the security parameter 1^λ and an encoding $z \in \{0,1\}^{s(\lambda)}$ outputs an image point $y \in \{0,1\}^{\ell(\lambda)}$; and (3) a probabilistic simulation algorithm Sim which on input 1^λ and an image point $y \in \{0,1\}^{\ell(\lambda)}$ outputs an encoding $z \in \{0,1\}^{s(\lambda)}$. To keep our notation consistent with the previous literature on randomized encoding, for a given circuit f_λ, we will refer to the the mapping $\mathsf{enc}(1^\lambda, f_\lambda, \cdot; \cdot)$ by $\hat{f}_\lambda : \{0,1\}^{n(\lambda)} \times \{0,1\}^{m(\lambda)} \to \{0,1\}^{s(\lambda)}$, where $\{0,1\}^{m(\lambda)}$ is the randomness space of enc. We say scheme RE is a perfectly correct, computationally private randomized encoding for a circuit class $\{\mathcal{F}_\lambda\}_{\lambda\in\mathbb{N}}$ if it satisfies the following two conditions.

– Correctness. For any $f_\lambda \in \mathcal{F}_\lambda$ and any input $x \in \{0,1\}^{n(\lambda)}$ we have that

$$\Pr\left[\mathsf{dec}(1^\lambda, \hat{f}_\lambda(x; r_{\mathsf{enc}})) = f_\lambda(x) : r_{\mathsf{enc}} \leftarrow_{\$} \{0,1\}^{m(\lambda)}\right] = 1 \ .$$

– Privacy. For any PPT distinguisher D, any $f_\lambda \in \mathcal{F}_\lambda$, and any input $x \in \{0,1\}^{n(\lambda)}$ the distinguishing advantage $\mathsf{Adv}^{\mathsf{re}}_{\mathsf{RE},D,x}$ is negligible, where advantage $\mathsf{Adv}^{\mathsf{re}}_{\mathsf{RE},D,x}(\lambda)$ is defined as:

$$\Pr[D(1^\lambda, \hat{f}_\lambda(x; r_{\mathsf{enc}})) = 1 : r_{\mathsf{enc}} \leftarrow_{\$} \{0,1\}^{m(\lambda)}] - \Pr[D(1^\lambda, \mathsf{Sim}(1^\lambda, f_\lambda(x)) = 1] \ .$$

Functions n, ℓ, s, and m are polynomials, however, we will be dropping the explicit dependency on λ in order to simplify notation, and set $n := n(\lambda)$, $\ell := \ell(\lambda)$, $s := s(\lambda)$, and $m := m(\lambda)$.

AIK used randomized encodings to construct cryptography in $\mathcal{NC}^0$. For us, the complexity of the encoding is not important. Rather, we will make use of encodings with small locality, where each bit in the randomized encoding $\hat{f}(x;r)$ only depends on at most a single bit of x (but possibly many bits of r). We will return to the topic of locality in Section 4.3.

4.2 Composing iO with randomized encodings

To ease readability, we present our attack in two stages. First, we show that our iO attack can be composed with any randomized encoding scheme in a way which neither affects the adversary's advantage nor the unpredictability of its implicit source. Then, in the next subsection, we use a special type of RE scheme known as *decomposable randomized encodings* [31] to split and parallelize the adversary's source in order to meet the (minimal) bounds of $q(\lambda) = 1$, $\sigma(\lambda) = 0$, and $\tau(\lambda) \in \mathcal{O}(\lambda)$. Consequently, our attack will rule out bounded parallel sources for these parameters. Since the bounds that our attacks achieves are very stringent, and an encryption scheme has to at least run in time $\mathcal{O}(\lambda)$ (and make a single HASH query), assuming indistinguishability obfuscation, it is unlikely that bounded parallel sources can be used to instantiate ROs in any meaningful application scenario.

Let H be a UCE$[\mathcal{S}^{\text{cup}} \cap \mathcal{S}^{\text{prl}}_{\tau,\sigma,q}]$-secure hash function, iO be an indistinguishability obfuscator, and let us assume once again that $\mathsf{H.ol}(\lambda) \geq 2 \cdot \mathsf{H.kl}(\lambda)$. (As in the proof of Theorem 1, this assumption will be without loss of generality.)

The attacker. Define $C_{\lambda,\mathsf{H},x,y}(\cdot) := (\mathsf{H.Ev}(1^\lambda, \cdot, x) = y)$, and compute a randomized encoding of the circuit

$$f : (x, y, r_{\mathsf{io}}) \mapsto \mathsf{iO}(C_{\lambda,\mathsf{H},x,y}(\cdot); r_{\mathsf{io}}),$$

where r_{io} is the randomness used by the obfuscator. As in the proof of Theorem 1, we consider the source S which chooses random values x, r_{io}, and r_{enc}, queries x to its oracle to obtain $y \leftarrow \text{HASH}(x)$, and leaks the randomized encoding

$$L := \hat{f}(x, y, r_{\mathsf{io}}; r_{\mathsf{enc}}) \ .$$

The distinguisher D gets as input a hash key hk and an encoding $\hat{f}(x, y, r_{\mathsf{io}}; r_{\mathsf{enc}})$. It uses the decoder dec of the randomized encoding scheme to recover

$$f(x, y, r_{\mathsf{io}}) \leftarrow \mathsf{dec}(\hat{f}(x, y, r_{\mathsf{io}}; r_{\mathsf{enc}})) \ .$$

It then interprets the result as a circuit, runs it on on hk, and returns whatever the circuit outputs.

By correctness of the randomized encoding, the advantage of the adversary is identical to the one in our original iO-attack. Moreover, the source is computationally unpredictable which follows when combining the analysis of the previous section with the privacy of the randomized encoding. We give the formal analysis of advantage and success probability in the full version [19].

4.3 Splitting and parallelizing S using decomposable REs

The attack described in the previous subsection works for any randomized encoding scheme. In particular, now, we will use a *decomposable* randomized encoding scheme to instantiate the attack; this allows us to recast the above source as a bounded parallel source. Let us begin with the definition decomposable randomized encodings.

Decomposable encodings. In a decomposable randomized encoding (DRE) scheme, every output bit of the encoding $\hat{f}(x; r)$ depends on at most a single bit of x (but possibly on arbitrarily many bits of r). More precisely, a decomposable randomized encoding scheme DRE consists of a four tuple of algorithms $(\mathsf{idx}, \mathsf{enc}, \mathsf{dec}, \mathsf{Sim})$ as follows. Algorithm idx on input a circuit f and an index $i \in [s]$ outputs an index $j \in [n] \cup \{0\}$. The decomposable encoding algorithm

enc operates based on a local encoding algorithm $\overline{\mathsf{enc}}$ as follows. On input a circuit f, a point x, and random coins r_{enc}, for each $i \in [s]$ compute $z_i \leftarrow \overline{\mathsf{enc}}(f, i, x[\mathsf{idx}(f, i)]; r_{\mathsf{enc}})$, where we define $x[0] := \perp$, and return $z \leftarrow (z_1, \ldots, z_s)$. Algorithms dec and Sim play the same roles as those in a conventional RE scheme. As before, we denote $\overline{\mathsf{enc}}(f, i, b; r_{\mathsf{enc}})$ by $\hat{f}_i(b; r_{\mathsf{enc}})$. Thus we may write

$$\hat{f}(x; r_{\mathsf{enc}}) = \hat{f}_1(x[\mathsf{idx}(1)]; r_{\mathsf{enc}}) \| \hat{f}_2(x[\mathsf{idx}(2)]; r_{\mathsf{enc}}) \| \cdots \| \hat{f}_s(x[\mathsf{idx}(s)]; r_{\mathsf{enc}}) \ .$$

As Ishai et al. [31] point out, several constructions of randomized encodings are decomposable. For example, AIK's construction based on garbled circuits [3] is a decomposable, perfectly correct, and computationally private randomized encoding for any function in $\mathcal{P}/poly$. Their construction relies only on the existence of secure pseudorandom generators.

Using decomposable encodings, we show that our attack can be parallelized. The idea is that each instance of S_1 is responsible for computing a single bit of $\hat{f}(x; r_{\mathsf{enc}})$. However, potentially, computing even a single bit of $\hat{f}(x; r_{\mathsf{enc}})$ can be a computationally heavy task. We thus outsource pre-computation to S_0 such that for S_1, computing a single bit of $\hat{f}(x; r_{\mathsf{enc}})$ becomes easy. For concreteness, let us think about the instance of S_1 that computes the first bit of $\hat{f}(x; r_{\mathsf{enc}})$. As the encoding is decomposable, the first bit of $\hat{f}(x; r_{\mathsf{enc}})$ only depends on a single bit x_i of x. S_0 now picks r_{enc} and computes the first bit of $\hat{f}(x; r_{\mathsf{enc}})$ simply for both cases, $x_i = 0$ and if $x_i = 1$. It obtains two values and passes these two values to S_1. Now, as source S_1 has access to HASH it can compute the actual x_i and its task is thus merely picking the right precomputed bit as output. We give the full description of the attack in the full version [19].

Theorem 2 (Bounded parallel UCE **infeasibility).** *If indistinguishability obfuscation (and PRGs) exist, then* $\mathrm{UCE}[\mathcal{S}^{\mathrm{cup}} \cap \mathcal{S}^{\mathrm{prl}}_{\tau,\sigma,q}]$ *security cannot be achieved in the standard model for* $q \neq 0$*, any* $\sigma \geq 0$*, and* $\tau \in \Omega(\lambda)$*.*

Following the above attack, BHK [12] retracted bounded parallel sources and replaced them by new source classes that are specifically designed according to each application scenario.

5 Split Sources

In principle, bounded parallel sources would also suffice to recover the application of UCEs to hardcore functions. However, for this purpose, BHK [10] introduce a second, simpler UCE notion which is based on computational unpredictability and so-called *split sources*. A split source S is composed of two algorithms S_0 and S_1, where neither gets direct access to the HASH oracle. Algorithm S_0 outputs L_0 together with a vector of points $\mathbf{x}$. For each entry of $\mathbf{x}$, the corresponding HASH value is computed, and the vector of hash values $\mathbf{y}$ is formed. Algorithm S_1 is then run on $\mathbf{y}$ produces leakage L_1. The leakage of the split source $S := \mathsf{Splt}[S_0, S_1]$ then equals $L := (L_0, L_1)$. We give the pseudocode in Figure 4 on the right.

Split sources avoid our original attack, as well as its generalized version, as neither component of the source gets direct access to the HASH oracle. In the full version of this work [19], we discuss the composition of split-source UCE-secure functions with one-way permutations and also study the implications of existence of certain forms of obfuscators on their feasibility.

For example, consider a hash function where its inputs are first run through a one-way permutation before being hashed. Intuitively, this application of a one-way permutation should not harm UCE security. Indeed, this can be easily seen to be the case for the standard notions of one-wayness, collision resistance, and pseudorandomness. We show that statistical UCE1 security also enjoys this property. However, when composing a UCE[$\mathcal{S}^{\rm cup} \cap \mathcal{S}^{\rm splt}$] hash functions with a one-way permutation, the resulting function fails to be UCE[$\mathcal{S}^{\rm cup} \cap \mathcal{S}^{\rm splt}$] secure.

We also show that certain levels of unobfuscatability are necessary for a hash function to achieve UCE security with respect to split sources. For instance, the function that maps x to an obfuscation of the circuit $\mathsf{H}(\cdot, x)$ must *not* be one way. Further, this must also be the case for obfuscators that are specially designed to support H. For example, as a practical instantiation of UCEs, BHK suggest to use the SHA family [34] in HMAC mode [8]. Our results imply that in order to obtain confidence in the security of this construction, its extractability properties in conjunction with, say, the candidate obfuscator of Garg et al. [24] should be studied. We note that due to their simplicity, our results potentially also apply to other UCE notions which rely on a computational unpredictability notion. We refer to [19] for a more detailed discussion of split sources.

Acknowledgments

We thank Mihir Bellare, Viet Tung Hoang, and Sriram Keelveedhi for their personal communication [11]. Christina Brzuska was supported by the Israel Science Foundation (grant 1076/11 and 1155/11), the Israel Ministry of Science and Technology grant 3-9094), and the German-Israeli Foundation for Scientific Research and Development (grant 1152/2011). Pooya Farshim was supported by grant Fi 940/4-1 of the German Research Foundation (DFG). Arno Mittelbach was supported by CASED (www.cased.de) and DFG SPP 1736.

References

1. Applebaum, B.: Bootstrapping obfuscators via fast pseudorandom functions. Cryptology ePrint Archive, Report 2013/699 (2013), http://eprint.iacr.org/2013/699
2. Applebaum, B., Ishai, Y., Kushilevitz, E.: Cryptography in NC^0. In: 45th FOCS. pp. 166–175. IEEE Computer Society Press (Oct 2004)
3. Applebaum, B., Ishai, Y., Kushilevitz, E.: Computationally private randomizing polynomials and their applications. Computational Complexity 15(2), 115–162 (2006)
4. Applebaum, B., Ishai, Y., Kushilevitz, E., Waters, B.: Encoding functions with constant online rate or how to compress garbled circuits keys. In: Canetti, R.,

Garay, J.A. (eds.) CRYPTO 2013, Part II. LNCS, vol. 8043, pp. 166–184. Springer (Aug 2013)

5. Barak, B., Garg, S., Kalai, Y.T., Paneth, O., Sahai, A.: Protecting obfuscation against algebraic attacks. In: Nguyen, P.Q., Oswald, E. (eds.) EUROCRYPT 2014. LNCS, vol. 8441, pp. 221–238. Springer (May 2014)
6. Barak, B., Goldreich, O., Impagliazzo, R., Rudich, S., Sahai, A., Vadhan, S.P., Yang, K.: On the (im)possibility of obfuscating programs. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 1–18. Springer (Aug 2001)
7. Bellare, M., Boldyreva, A., Palacio, A.: An uninstantiable random-oracle-model scheme for a hybrid-encryption problem. In: Cachin, C., Camenisch, J. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 171–188. Springer (May 2004)
8. Bellare, M., Hoang, V.T., Keelveedhi, S.: Instantiating random oracles via UCEs. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part II. LNCS, vol. 8043, pp. 398–415. Springer (Aug 2013)
9. Bellare, M., Hoang, V.T., Keelveedhi, S.: Instantiating random oracles via UCEs. Cryptology ePrint Archive, Report 2013/424. (Aug 1, 2013), (Latest version prior to our attack [11].) http://eprint.iacr.org/2013/424/20130801:043135.)
10. Bellare, M., Hoang, V.T., Keelveedhi, S.: Instantiating random oracles via UCEs. Cryptology ePrint Archive, Report 2013/424. (Oct 17, 2013), (Latest version at the time of writing.) http://eprint.iacr.org/2013/424/20131017:000316
11. Bellare, M., Hoang, V.T., Keelveedhi, S.: Personal communication (Sep, 2013)
12. Bellare, M., Hoang, V.T., Keelveedhi, S.: Instantiating random oracles via UCEs. Cryptology ePrint Archive, Report 2013/424. (May 20, 2014), (Latest version at the time of writing.) http://eprint.iacr.org/2013/424/20140520:182716
13. Bellare, M., Rogaway, P.: Random oracles are practical: A paradigm for designing efficient protocols. In: Ashby, V. (ed.) ACM CCS 93. pp. 62–73. ACM Press (Nov 1993)
14. Bellare, M., Rogaway, P.: The security of triple encryption and a framework for code-based game-playing proofs. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 409–426. Springer (May / Jun 2006)
15. Boyle, E., Chung, K.M., Pass, R.: On extractability obfuscation. In: Lindell, Y. (ed.) TCC 2014. LNCS, vol. 8349, pp. 52–73. Springer (Feb 2014)
16. Brakerski, Z., Rothblum, G.N.: Obfuscating conjunctions. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part II. LNCS, vol. 8043, pp. 416–434. Springer (Aug 2013)
17. Brakerski, Z., Rothblum, G.N.: Black-box obfuscation for d-CNFs. In: Naor, M. (ed.) ITCS 2014. pp. 235–250. ACM (Jan 2014)
18. Brakerski, Z., Rothblum, G.N.: Virtual black-box obfuscation for all circuits via generic graded encoding. In: Lindell, Y. (ed.) TCC 2014. LNCS, vol. 8349, pp. 1–25. Springer (Feb 2014)
19. Brzuska, C., Farshim, P., Mittelbach, A.: Indistinguishability obfuscation and UCEs: The case of computationally unpredictable sources. Cryptology ePrint Archive, Report 2014/099 (2014), http://eprint.iacr.org/2014/099
20. Canetti, R., Goldreich, O., Halevi, S.: The random oracle methodology, revisited (preliminary version). In: 30th ACM STOC. pp. 209–218. ACM Press (May 1998)
21. Canetti, R., Goldreich, O., Halevi, S.: On the random-oracle methodology as applied to length-restricted signature schemes. In: Naor, M. (ed.) TCC 2004. LNCS, vol. 2951, pp. 40–57. Springer (Feb 2004)
22. Dodis, Y., Oliveira, R., Pietrzak, K.: On the generic insecurity of the full domain hash. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 449–466. Springer (Aug 2005)

23. Garg, S., Gentry, C., Halevi, S., Raykova, M.: Two-round secure MPC from indistinguishability obfuscation. In: Lindell, Y. (ed.) TCC 2014. LNCS, vol. 8349, pp. 74–94. Springer (Feb 2014)

24. Garg, S., Gentry, C., Halevi, S., Raykova, M., Sahai, A., Waters, B.: Candidate indistinguishability obfuscation and functional encryption for all circuits. In: 54th FOCS. pp. 40–49. IEEE Computer Society Press (Oct 2013)

25. Goldwasser, S., Kalai, Y.T.: On the (in)security of the Fiat-Shamir paradigm. In: 44th FOCS. pp. 102–115. IEEE Computer Society Press (Oct 2003)

26. Goldwasser, S., Rothblum, G.N.: On best-possible obfuscation. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, pp. 194–213. Springer (Feb 2007)

27. Goyal, V., Ishai, Y., Sahai, A., Venkatesan, R., Wadia, A.: Founding cryptography on tamper-proof hardware tokens. In: Micciancio, D. (ed.) TCC 2010. LNCS, vol. 5978, pp. 308–326. Springer (Feb 2010)

28. Hohenberger, S., Sahai, A., Waters, B.: Replacing a random oracle: Full domain hash from indistinguishability obfuscation. In: Nguyen, P.Q., Oswald, E. (eds.) EUROCRYPT 2014. LNCS, vol. 8441, pp. 201–220. Springer (May 2014)

29. Ishai, Y., Kushilevitz, E.: Randomizing polynomials: A new representation with applications to round-efficient secure computation. In: 41st FOCS. pp. 294–304. IEEE Computer Society Press (Nov 2000)

30. Ishai, Y., Kushilevitz, E.: Perfect constant-round secure computation via perfect randomizing polynomials. In: Automata, Languages and Programming, 29th International Colloquium, ICALP 2002, Malaga, Spain, July 8-13, 2002, Proceedings. pp. 244–256 (2002)

31. Ishai, Y., Kushilevitz, E., Ostrovsky, R., Sahai, A.: Cryptography with constant computational overhead. In: Ladner, R.E., Dwork, C. (eds.) 40th ACM STOC. pp. 433–442. ACM Press (May 2008)

32. Kiltz, E., Pietrzak, K.: On the security of padding-based encryption schemes - or - why we cannot prove OAEP secure in the standard model. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 389–406. Springer (Apr 2009)

33. Matsuda, T., Hanaoka, G.: Chosen ciphertext security via UCE. In: Krawczyk, H. (ed.) PKC 2014. LNCS, vol. 8383, pp. 56–76. Springer (Mar 2014)

34. National Institute of Standards and Technology: FIPS 180-4, Secure Hash Standard (SHS). Tech. rep. (March 2012)

35. Nielsen, J.B.: Separating random oracle proofs from complexity theoretic proofs: The non-committing encryption case. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 111–126. Springer (Aug 2002)

36. Ristenpart, T., Shacham, H., Shrimpton, T.: Careful with composition: Limitations of the indifferentiability framework. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 487–506. Springer (May 2011)

37. Sahai, A., Waters, B.: How to use indistinguishability obfuscation: Deniable encryption, and more. Cryptology ePrint Archive, Report 2013/454 (2013), http://eprint.iacr.org/2013/454