Using Indistinguishability Obfuscation via UCEs

Christina Brzuska[1] Arno Mittelbach[2]

[1]Tel Aviv University, Israel
[2]Darmstadt University of Technology, Germany
brzuska@post.tau.ac.il arno.mittelbach@cased.de

Abstract. We provide the first standard model construction for a powerful class of Universal Computational Extractors (UCEs; Bellare et al. Crypto 2013) based on indistinguishability obfuscation. Our construction suffices to instantiate correlation-secure hash functions and universal one-way functions.

For many cryptographic primitives and in particular for correlation-secure hash functions all known constructions are in the random-oracle model. Indeed, recent negative results by Wichs (ITCS 2013) rule out a large class of techniques to prove the security of correlation-secure hash functions in the standard model. Our construction is based on puncturable PRFs (Sahai und Waters; STOC 2014) and indistinguishability obfuscation. However, our proof also relies on point obfuscation under auxiliary inputs (AIPO). This is crucial in light of Wichs' impossibility result. Namely, Wichs proves that it is often hard to reduce two-stage games (such as UCEs) to a "one-stage assumption" such as DDH. In contrast, AIPOs and their underlying assumptions are inherently two-stage and, thus, allow us to circumvent Wichs' impossibility result.

Our positive result is also noteworthy insofar as Brzuska, Farshim and Mittelbach (Crypto 2014) have shown recently, that iO and some variants of UCEs are mutually exclusive. Our results, hence, validate some of the new UCE notions that emerged as a response to the iO-attack.

Keywords. Correlation-secure hash functions, hardcore functions, indistinguishability obfuscation, differing-inputs obfuscation, point-function obfuscation, auxiliary-input obfuscation, universal computational extractors (UCEs)

Contents

1 Introduction

For many cryptographic primitives, it is easy to construct a secure scheme in the random oracle model, but it is hard to give a construction in the standard model. For example, correlated-input hash functions (CIH) which were introduced by Goyal, O'Neill, and Rao [GOR11], are easy to construct in the random oracle model, because the random oracle itself is secure under correlated inputs. However, up to now, no standard-model construction is known, and indeed, a recent black-box separation by Wichs [Wic13] explains why it is so hard to construct them. Namely, the security definition of a CIH involves a pair of adversaries $(\mathcal{A}_1, \mathcal{A}_2)$ and is thus a two-stage game (i.e., the adversary is not a single algorithm but consists of two separate algorithms). The first adversary samples correlated inputs $(x_1, ..., x_t)$. Then a hash key hk is generated and the second adversary with access to hk needs to distinguish between getting a tuple of random strings and getting the tuple $(H(\mathsf{hk}, x_1), ..., H(\mathsf{hk}, x_t))$. Now, Wichs employs a meta reduction to show that it is unlikely to have a black-box reduction $\mathcal{R}$ from CIH to a (one-stage) cryptographic assumption such as the decisional Diffie–Hellman assumption (DDH). Namely, he shows that if such a reduction to DDH exists, then the DDH assumption is wrong. In his proof, he substantially exploits that the CIH game is a two-stage game. For a black-box reduction $\mathcal{R}$ it must hold that if the reduction $\mathcal{R}$ gets access to a pair of oracles $(\mathcal{A}_1, \mathcal{A}_2)$ that break CIH, then $\mathcal{R}^{\mathcal{A}_1,\mathcal{A}_2}$ must also break DDH. Wichs constructs a pair of inefficient $(\mathcal{A}_1, \mathcal{A}_2)$ which, however, can be efficiently emulated using a simulator Sim that is stateful. That is, the simulator simulates both adversaries together while sharing state between them. As the reduction cannot distinguish between the two settings $\mathcal{R}^{\mathcal{A}_1,\mathcal{A}_2}$ and $\mathcal{R}^{\mathsf{Sim}}$ this breaks DDH, and hence, if we believe that DDH is a hard problem, then such an $\mathcal{R}$ cannot exist. His proof is not specific to DDH, but rather applies to any one-stage assumption and presents a substantial barrier to prove security.

Before coming back to possibilities of circumventing Wichs' impossibility result let us briefly talk about two obfuscation techniques that we will employ in this paper: point-function obfuscation and indistinguishability obfuscation.

Point and indistinguishability obfuscation. A point function p_x is a function that returns 1 on input x and $\perp$ on all other values. A point function obfuscator under auxiliary input AIPO returns a point function $p \leftarrow_\$ \mathsf{AIPO}(x)$ that hides the point x even in case the adversary receives some side-channel information about x. More formally, the security of AIPO is defined as security for all computationally unpredictable distributions $\mathcal{D}$, that is, $\mathcal{D}$ outputs a pair (x, z), where x is a point and z is some leakage that hides x computationally. AIPO is secure, if for all computationally unpredictable $\mathcal{D}$, $(\mathsf{AIPO}(x), z)$ is indistinguishable from $(\mathsf{AIPO}(u), z)$, where $(x, z) \leftarrow \mathcal{D}$ and u is a uniformly random point. Such AIPO schemes have been constructed in [Can97, BP12].

While point function obfuscators are obfuscation schemes for a very specific class of functionalities (namely point functions) Garg et al. [GGH$^+$13] have recently revived the study of general obfuscation schemes with their candidate construction of indistinguishability obfuscation. The notion of indistinguishability obfuscation is weaker than VBB-obfuscation—thereby circumventing the impossibility results of Barak et al. [BGI$^+$01, BGI$^+$12]—and says intuitively that, for any two circuits that compute the same function, their obfuscations are indistinguishable. With the publication of their candidate, Garg et al. revived the interest and inspired simultaneous breakthroughs for hard problems in various sub-areas of cryptography [SW13, BCP14, ABG$^+$13, GGHR14, HSW14, BZ13, BST13, GGG$^+$14] including functional and deniable encryption, two-round secure multi-party computation, full-domain hash, poly-many hardcore bits from any one-way function, multi-input functional encryption and more.

Correlated-input hash-functions. In this paper, we give the first standard-model construction for CIHs. On a high-level, our construction is a de facto instantiation of a random oracle. As the behavior of a PRF is similar to that of a random function, we instantiate the random oracle by securely delegating a PRF, that is, we obfuscate a PRF with a hard coded key. Indeed, our hash-function construction only consists of a (puncturable) PRF that is obfuscated via an indistinguishability obfuscator (iO):

$$\textbf{Hash Construction: } \mathsf{iO}(\mathsf{PRF}(k, .)) \ .$$

This natural construction has already proved useful in the direct construction of universal hardcore functions from differing-inputs obfuscation by Bellare, Stepanovs, and Tessaro (BST; [BST13]). We will discuss BST and the relation to our our work shortly.

Circumventing Wichs' impossibility result. Although the construction is natural, proving its security is non-trivial, as the security guarantees of iO do not even allow us to show easily that it is hard to extract the PRF key. Towards proving the security of our construction, we build on the puncturable PRF technique by Waters and Sahai [SW13] and combine it with point function obfuscators secure under auxiliary input (AIPO).

Using AIPOs is crucial to circumvent the impossibility result by Wichs [Wic13], because the security of AIPOs is defined via a two-stage security game. The first AIPO adversary samples a point, and the second adversary tries to break the obfuscation of the point function. In a sense, the impossibility result of Wichs tells us that using a two-stage assumption such as AIPO in the proof is, indeed, necessary. In particular, iO and PRFs are both one-stage assumptions. Note that, as AIPOs are only used in the proof and not in the construction, it might be possible that the same construction can be proven secure without making use of AIPOs possibly through some other two-stage assumption.

Universal hardcore functions for any one-way function. The same construction also yields a universal hardcore function for any one-way function. This result has recently been established by Bellare, Stepanovs, and Tessaro [BST13], however, under different assumptions. For some classes of one-way functions, they originally assumed a variant of so-called differing-inputs obfuscation (diO). Differing-inputs obfuscation is a stronger assumption than iO and indeed, it has been shown conditionally impossible by Garg et al. [GGHW13] assuming special-purpose obfuscators. Therefore, in the current version of their paper, Bellare et al. [BST13] use variants of diO called diO$^-$ that are not affected by the results of Garg et al. [GGHW13].

Our proof for the same construction, instead, assumes AIPO in addition to iO. Thereby, we are able to avoid diO variants altogether. As the construction is the same, our proof can be shown as additional positive evidence for the construction. If iO and one-way functions exist, then the construction is secure, as long as at least one out of diO$^-$ or AIPO exist. Our proof builds on ideas by Bellare et al. [BST13], and we will discuss their result in the context of presenting our proof techniques.

Modularizing proofs via UCEs. We could prove the security of our construction directly, but instead, we split our proof into two parts. First, we show that our construction enjoys some useful, abstract properties. Then we use results by Bellare et al. [BHK13a] that show that these abstract properties suffice for the application at hand. This way, we provide a means of using iO in a black-box way. Our abstraction is a version of UCE security [BHK13a] that we discuss next.

The UCE Framework by Bellare, Hoang, and Keelveedhi (BHK; [BHK13a]) introduces assumptions that allow us to instantiate random oracles in a wide range of applications. Loosely speaking, UCEs are PRF-like assumptions that split the distinguisher into two parts: a first adversary S that gets access to a keyed hash function or a random oracle (and which is called the *source*), and a second adversary $\mathcal{D}$ that gets the hash key hk (and which is called the *distinguisher*). The two algorithms together try to guess whether the source was given access to a keyed hash function or to a random oracle.

Concretely, the UCE notions are defined via a two-stage UCE game (we depict the communication flow in Figure 1 and the pseudocode in Figure 2). First, the source S is run with oracle access to $\textsc{Hash}$ to output some leakage L. Subsequently, distinguisher D is run on the leakage L and hash key hk but without access to oracle $\textsc{Hash}$. Distinguisher D outputs a single bit b indicating whether oracle $\textsc{Hash}$ implements a random oracle or hash function H with key hk.

Without any restrictions, (S, D) can easily win the UCE game. For example, say, source S makes a random query x to receive $y \leftarrow \textsc{Hash}(x)$ and outputs (x, y) as leakage. As distinguisher D knows the hash key hk as well as the leakage (x, y), it can recompute the hash value and check whether $y = \mathsf{H}(\mathsf{hk}, x)$. BHK present several possible restrictions on the source which give rise to various UCE notions.

It turns out to be particularly useful to restrict sources to be computationally unpredictable, that is, the leakage created by the source S—when interacting with a random oracle— should not reveal (computationally) any of the source's queries to $\textsc{Hash}$. This notion is denoted by UCE[$\mathcal{S}^{\mathrm{cup}}$], where $\mathcal{S}^{\mathrm{cup}}$ denotes the class of computationally unpredictable sources [BHK13b]. BHK show that UCE[$\mathcal{S}^{\mathrm{cup}}$]-secure hash functions can safely replace a random oracle in a large number of interesting applications such as hardcore functions or deterministic public-key encryption [BHK13a]. In a recent work Brzuska, Farshim and Mittelbach (BFM;[BFM14]) show that UCE security with respect to computational unpredictability cannot be achieved in the standard model assuming indistinguishability obfuscation exists. Several refinements have been proposed since including a statistical notion of unpredictability denoted by $\mathcal{S}^{\mathrm{sup}}$ as well as source classes containing sources that are structurally required to produce output in a special way (called split sources and bounded parallel sources) as well as sources which are restricted to only a fixed number of queries [BHK13b, BFM14].

Our notion of UCE security strengthens the notion of unpredictability to what we call strong unpredictability and we denote the corresponding class of sources by $\mathcal{S}^{\text{s-cup}}$ for the computational variant and by $\mathcal{S}^{\text{s-sup}}$ for its statistical version. Namely, we demand that the leakage be (computationally/statistically) unpredictable even if the predictor additionally gets the answers to the queries that the source received from the oracle. We give the pseudo-code for strong unpredictability in Figure 3.

It turns out that UCEs for strongly computationally unpredictable sources that can only make a single query (denoted by UCE[$\mathcal{S}^{\text{s-cup}} \cap \mathcal{S}^{\text{1-query}}$]) already imply hardcore functions for any one-way function. Furthermore, UCEs for strongly statistically unpredictable sources (denoted UCE[$\mathcal{S}^{\text{s-sup}}$]) imply correlation-secure hash functions. We note that strongly unpredictable sources can be regarded as a generalization of so-called split sources [BHK13b] which were introduced by BHK after the BFM impossibility results. We will discuss the exact relationship later.

So far UCEs have only been constructed in idealized models. BHK showed that a random oracle is UCE-secure in the strongest proposed settings and conjectured that HMAC is UCE-secure if the underlying compression function is modeled as an ideal function. This conjecture has recently been confirmed by Mittelbach [Mit14] who shows that HMAC and various Merkle-Damgård variants are UCE-secure in the ideal compression function model. We note that so far, no standard model instantiation of any (non-trivial) UCE variant has been proposed and, hence, we present the first standard model construction of UCEs.[1]

Techniques. Our construction is based on indistinguishability obfuscation and similar to many other recent constructions from iO [SW13, BST13, HSW14, BZ13] our construction also makes use of puncturable PRFs [SW13] which admit the generation of keys that allow to evaluate the PRF on all points except for points in a small target set (often containing just a single point). Our security reduction, however, differs from existing techniques. That is, we make use of point function obfuscations which allows us to hide the punctured points within our constructed circuits. Hiding the punctured points was also the key problem in a recent work by Bellare, Stepanovs and Tessaro [BST13] who show how to build hardcore functions for any one-way function from indistinguishability obfuscation and differing-inputs obfuscation. They solve the problem elegantly by using the one-way function from the security game to blind the punctured point by embedding the image under the one-way function. However, when testing whether a given point is equivalent to the punctured point this test is not unambiguous which is why they need to assume differing-inputs obfuscators for certain types of one-way functions. This is where point function obfuscation comes into the picture which allows us to bypass any assumptions related to differing-input obfuscation variants. Yet, of course, point obfuscators are as far as is currently known an assumption incomparable to differing-inputs obfuscation.

We next discuss the specific UCE assumptions that our construction will meet and the relation to point obfuscation. In Section 3 we will show that our construction is $\mathrm{UCE}[\mathcal{S}^{\text{s-cup}} \cap \mathcal{S}^{\text{1-query}}]$-secure assuming iO, puncturable PRFs and the existence of AIPO. That is, we consider UCE-secure for computationally strongly unpredictable sources that make a single query. In Section 3.3, we prove that our construction is also $\mathrm{UCE}[\mathcal{S}^{\text{s-sup}} \cap \mathcal{S}^{q\text{-query}}]$-secure, that is, secure against statistically unpredictable sources that make at most q queries.

As explained, we base the security of our construction on the existence of a different (incomparable) notion of point obfuscation. We consider a notion of AIPO which only needs to be secure against statistically unpredictable distributions but, in turn, we require it to be q-composable [CD08, BC10]. q-composability intuitively says that an obfuscation remains secure even if an adversary sees q many (possibly related) obfuscations. The reason that we need q-composable AIPO is that now, the source is a allowed to make q queries and hence, we need to hide q points in the proof. However, as we here only consider sources in $\mathcal{S}^{\text{s-sup}}$, that is, sources which are only statistically strongly unpredictable, it suffices that our AIPO-notion is secure against statistically unpredictable samplers which weakens the notion of AIPO. Note that, for the proof to work, we need to let the circuit size of our construction grow, artificially, with the number of queries q. Towards this goal, we use some padding that does not have any functionality.

In summary we get the following results:

Theorem [informal].

- *Our construction is* $\mathrm{UCE}[\mathcal{S}^{\text{s-cup}} \cap \mathcal{S}^{\text{1-query}}]$*-secure assuming* AIPO *secure against computationally unpredictable sources exist.*

- *Our construction is* $\mathrm{UCE}[\mathcal{S}^{\text{s-sup}} \cap \mathcal{S}^{q\text{-query}}]$*-secure assuming q-composable* AIPO *for statistically unpredictable sources, exist.*

[1]The UCE Framework is very flexible, and hence, it is possible to come up with a UCE restriction that corresponds to PRF security and is thus easy to achieve in the standard model.

On the feasibility of our AIPO assumptions. Standard AIPO secure against computationally unpredictable samplers has been constructed by Canetti in [Can97] under (non-standard) variants of the DDH assumption and by Bitansky and Paneth in [BP12] under (non-standard) assumptions on pseudorandom permutations. We present the constructions and the underlying assumptions in Appendix A. One might hope that AIPO is naturally composable. However, as Canetti et al. [CD08, BC10] show, this is generally not the case. On the other hand, Bitansky and Canetti [BC10] show that under the *t-Strong Vector Decision Diffie Hellman assumption* the original point obfuscation scheme of Canetti [Can97] composes in the so-called virtual grey-box (VGB) setting. The VGB setting was introduced by Bitansky and Canetti [BC10] and is a relaxation of the strongest obfuscation setting the virtual black-box (VBB) setting [BGI+01, BGI+12]. Similarly to VBB obfuscation, VGB obfuscation is in general not achievable, yet for the class of point functions it seems in reach [BC10]. The VGB setting is particularly interesting because it can be shown that allowing for auxiliary input does not yield a more powerful notion of obfuscation when considering the VGB setting [BC10]. This result stands in contrast to the VBB setting where allowing auxiliary information results in a stronger notion. Furthermore, we currently have no candidate constructions for composable point obfuscation schemes in this stronger setting.

On the feasibility of our UCE notions. In a recent work, Brzuska, Farshim, and Mittelbach (BFM; [BFM14]) show that, assuming indistinguishability obfuscation exists, no standard model hash construction can be UCE-secure with respect to computationally unpredictable sources. Our construction achieves a weaker yet related notion of security, namely UCE-security with respect to strongly computationally unpredictable sources which raises the question whether the BFM result can be extended to this setting.

The BFM result crucially hinges on the possibility of extending the output-length of the studied hash construction such that it is significantly larger than the key size. For example, this can be achieved by using multiple queries to the hash construction or via extending the output size by applying a pseudo-random generator. Both approaches fail with our construction: the size of our hash key grows with the number of allowed queries and since we consider strong unpredictability it seems implausible to prove the construction $\mathsf{PRG}(\mathsf{H}(\cdot,\cdot))$-secure under the assumption that H is UCE-secure with respect to strongly computationally unpredictable sources. Thus, we think that extending the BFM attack is implausible. Furthermore, if it can be extended this would immediately imply that indistinguishability obfuscation implies the non-existence of AIPO, which would be a surprising result. We discuss the BFM result in greater detail in Section 5 and note that, even if an extension of the BFM result were to break AIPOs with computational unpredictability, then the second construction would not be affected, as it only considers AIPOs secure with respect to statistically hard-to-invert auxiliary information.

2 Preliminaries

Notation. By $\lambda \in \mathbb{N}$, we denote the security parameter that we give to all algorithms implicitly in 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. If $x, y \in \{0,1\}^*$ are two bit strings of the same length, then we denote their inner product over $\mathbb{GF}(2)$ by $\langle x, y\rangle$. The length of x is denoted by $|x|$. 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 $|X|$. We denote by $[i]$ the set $\{1, \ldots, i\}$. Algorithms are assumed to be randomized, unless otherwise stated. We call a algorithm efficient or PPT if it runs in time polynomial in the security parameter. If $\mathcal{A}$ is randomized then by $y \leftarrow \mathcal{A}(x; r)$ we denote that $\mathcal{A}$ is run on input x and with random coins r and produced output y. If no randomness is specified, then we assume that $\mathcal{A}$ is run with freshly sampled uniform random coins, and write this as $y \leftarrow_{\$} \mathcal{A}(x)$. We often refer to algorithms, or tuples of algorithms, as adversaries. If E is an event then we denote by $\Pr[E]$ its probability and if X is a random variable, we denote its expectation by $\mathbb{E}[X]$. We say a function $\text{negl}(\lambda)$ is negligible if $\text{negl}(\lambda) \in \lambda^{-\omega(1)}$. We say a function poly is polynomial if $\text{poly} \in \lambda^{\mathcal{O}(1)}$.

2.1 Obfuscation

Obfuscation has a long tradition within cryptographic research and comes in many flavors. In the following section we present the various definitions that we use in this paper. We discuss constructions and candidates in Appendix A.

We start by recalling the strongest definition of virtual black-box (VBB) obfuscation with auxiliary inputs due to [BGI+01, GK05, BGI+12].

Definition 2.1 (Worst-case obfuscator with auxiliary input). *A* PPT $\mathcal{O}$ *is a worst-case obfuscator with auxiliary input for an ensemble* $\mathcal{C} = \{\mathcal{C}_\lambda\}_{\lambda \in \mathbb{N}}$ *of families of poly-size circuits if it satisfies:*

- **Functionality.** *For any* $\lambda \in \mathbb{N}$ *and* $C \in \mathcal{C}_\lambda$, $\mathcal{O}(C)$ *is a circuit which computes the same function as* C*.*
- **Polynomial slowdown.** *For any* $\lambda \in \mathbb{N}$ *and* $C \in \mathcal{C}_\lambda$, $|\mathcal{O}(C)| \leq \text{poly}(|C|)$*.*
- **Virtual black-box.** *For any* PPT *adversary* $\mathcal{A}$ *there is a* PPT *simulator* Sim *such that for all sufficiently large* $\lambda \in \mathbb{N}, C \in \mathcal{C}_\lambda$ *and* $z \in \{0,1\}^{\mathsf{poly}(\lambda)}$*:*

$$\left|\Pr[\mathcal{A}(z, \mathcal{O}(C)) = 1] - \Pr\left[\mathsf{Sim}^C(z, 1^{|C|}) = 1\right]\right| \leq \text{negl}(\lambda)$$

  where the probability is taken over the coins of $\mathcal{A}$, Sim *and* $\mathcal{O}$*.*

VBB obfuscation with auxiliary input requires that for any PPT adversary given the code of some functionality (and some auxiliary input) there exists a PPT simulator that given only black-box access to the functionality (and as input the same auxiliary input) produces a computationally indistinguishable distribution.

A provably weaker notion of obfuscation called virtual grey-box (VGB) was introduced by Bitansky and Canetti [BC10]. VGB is defined analogously to VBB with the exception that the simulator is given unbounded computation time but still restricted to only make polynomially many oracle queries. We will return to VGB obfuscation when discussing composition of so-called point function obfuscators.

Indistinguishability obfuscation. While VBB and VGB obfuscation as defined above provably do not exist in general [BGI+01, BC10] for all circuits, weaker notions such as *indistinguishability obfuscation* may well exist. While VBB requires the existence of a simulator, an indistinguishability obfuscation (iO) scheme, only ensures that the obfuscations of any two functionally equivalent circuits are computationally indistinguishable. Indistinguishability obfuscation was originally proposed by Barak et al. [BGI+01] as a potential weakening of virtual-black-box obfuscation. We recall the definition from [GGH+13].

Definition 2.2. *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*
$$\Pr\Big[C'(x) = C(x) : C' \leftarrow_\$ \mathsf{iO}(1^\lambda, C)\Big] = 1 \ .$$
- **Security.** *For any* PPT *distinguisher $\mathcal{D}$, 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:*
$$\Big|\Pr\Big[\mathcal{D}(1^\lambda, \mathsf{iO}(1^\lambda, C_1)) = 1\Big] - \Pr\Big[\mathcal{D}(1^\lambda, \mathsf{iO}(1^\lambda, C_0)) = 1\Big]\Big| \leq \text{negl}(\lambda) \ .$$

Closely related to indistinguishability obfuscation is the notion of *differing-inputs obfuscation* (diO) which also goes back to the seminal paper of Barak et al. [BGI+01]. Building on a Theorem by Boyle, Chung and Pass [BCP14], we are able to avoid diO as an assumption and only use it as an intermediary concept in our proof. Hence, we defer it to Appendix 2.1.

Differing-Inputs Obfuscation. The notion of *differing-inputs obfuscation* is closely related to indistinguishability obfuscation and also goes back to the seminal paper of Barak et al. [BGI+01]. While indistinguishability obfuscation requires circuits to be identical on all inputs, differing-inputs obfuscation intuitively says that if a distinguisher can tell apart two obfuscated circuits then one can efficiently extract a value on which the circuits differ. We follow the definition of Ananth et al. [ABG+13] and Boyle et al. [BCP14] and first define the notion of *differing-inputs circuits.*

Definition 2.3 (Differing-Inputs Circuits). *A circuit family $\{\mathcal{C}_\lambda\}_{\lambda\in\mathbb{N}}$ with a sample algorithm $(C_0, C_1, z) \leftarrow_\$ \mathsf{Sam}(1^\lambda)$ which samples $C_0, C_1 \in \mathcal{C}_\lambda$ is said to be a* differing-inputs *family if for all* PPT *algorithms $\mathcal{A}$ there is a negligible function* negl *such that:*
$$\Pr\Big[C_0(x) \neq C_1(x) : (C_0, C_1, z) \leftarrow_\$ \mathsf{Sam}(1^\lambda), x \leftarrow_\$ \mathcal{A}(1^\lambda, C_0, C_1, z)\Big] \leq \text{negl}(\lambda)$$

Definition 2.4 (Differing-Inputs Obfuscation). *A* PPT *algorithm* diO *is a* differing-inputs obfuscator *for a differing-inputs family $(\{\mathcal{C}_\lambda\}, \mathsf{Sam})$ if the following holds:*

- **Correctness.** *For all security parameters $\lambda \in \mathbb{N}$, for all $C \in \mathcal{C}_\lambda$, and for all inputs x we have that*
$$\Pr\Big[C'(x) = C(x) : C' \leftarrow_\$ \mathsf{diO}(1^\lambda, C)\Big] = 1 \ .$$
- **Security.** *For any* PPT *distinguisher $\mathcal{D}$, for any $(C_0, C_1, z) \leftarrow_\$ \mathsf{Sam}(1^\lambda)$ the following distinguishing advantage is negligible:*
$$\Big|\Pr\Big[\mathcal{D}(1^\lambda, \mathsf{diO}(1^\lambda, C_1), z) = 1\Big] - \Pr\Big[\mathcal{D}(1^\lambda, \mathsf{diO}(1^\lambda, C_0), z) = 1\Big]\Big| \leq \text{negl}(\lambda) \ .$$

The notion of differing-inputs obfuscation recently also gained much attention [ABG+13, BCP14, BP13]. In particular, we will build on the work by Boyle, Chung and Pass [BCP14] who show that any general indistinguishability obfuscator also yields a mild version of a differing-inputs obfuscator. That is, any indistinguishability obfuscator for all circuits in $\mathcal{P}$/poly is a also a differing-inputs obfuscator for circuits that differ on at most polynomially many inputs. We will use their result in a crucial way on circuits that differ on a single input.

Theorem 2.5 ([BCP14]). *Let* iO *be an indistinguishability obfuscator for $\mathcal{P}$/poly. Let $(\{\mathcal{C}_\lambda\}, \mathsf{Sam})$ be a differing-inputs family for which there exists a polynomial $d : \mathbb{N} \to \mathbb{N}$, such that for all $\lambda \in \mathbb{N}$ and all pairs $C_0, C_1 \in \mathcal{C}_\lambda$ it holds that $|\{x : C_0(x) \neq C_1(x)\}| \leq d(\lambda)$. Then* iO *is a differing-inputs obfuscator for $(\{\mathcal{C}_\lambda\}, \mathsf{Sam})$.*

Point obfuscation. While indistinguishability, as well as differing-inputs, obfuscation are obfuscation schemes for general circuits one can also study obfuscation schemes for particular function classes such as point functions. A point function I_x for some value $x \in \{0,1\}^*$ is defined as

$$I_x(s) := \begin{cases} 1 & \text{if } s = x \\ \perp & \text{o/w} \end{cases}$$

We consider a variant of point function obfuscators under auxiliary input which was first formalized by Canetti [Can97], although in a slightly different context. We here give the definition from [BP12]. The first definition formalizes unpredictable distributions which are in turn used to define obfuscators for point functions.

Definition 2.6 (Unpredictable Distribution). *A distribution ensemble $\mathcal{D} = \{D_\lambda = (Z_\lambda, Y_\lambda)\}_{\lambda\in\mathbb{N}}$, on pairs of strings is unpredictable if no poly-size circuit family can predict Y_λ from Z_λ. That is, for every poly-size circuit family $\{C_\lambda\}_{\lambda\in\mathbb{N}}$ and for all large enough λ:*

$$\Pr_{(z,y) \leftarrow\$ D_n}\left[C_\lambda(z) = y\right] \leq \text{negl}(\lambda)$$

Definition 2.7 (Auxiliary input point obfuscation for unpredictable distributions (AIPO)). *A* PPT *algorithm* AIPO *is a* point obfuscator for unpredictable distributions *if it satisfies the functionality and polynomial slowdown requirements as in Definition 2.1, and the following secrecy property: for any unpredictable distribution $\mathcal{D} = \{D_\lambda = (Z_\lambda, Y_\lambda)\}_{\lambda\in\mathbb{N}}$ over $\{0,1\}^{\text{poly}(\lambda)} \times \{0,1\}^\lambda$ it holds for any* PPT *algorithms $\mathcal{A}$ that there exists a negligible function* negl *such that:*

$$\left|\Pr_{(z,y) \leftarrow\$ D_\lambda}\left[\mathcal{A}(1^\lambda, \mathsf{AIPO}(y), z) = 1\right] - \Pr_{z \leftarrow\$ Z_\lambda, u \leftarrow\$ \{0,1\}^\lambda}\left[\mathcal{A}(1^\lambda, \mathsf{AIPO}(u), z) = 1\right]\right| \leq \text{negl}(\lambda)$$

Composable VGB point obfuscation. The definition of AIPO requires that a single point obfuscation is secure. A natural question to ask is whether the scheme remains secure even if the adversary is allowed to see multiple obfuscations, possibly of related points. This leads to the study of composition of obfuscators and the version we consider in this work is composition by concatenation formalized by Lynn, Prabhakaran, and Sahai [LPS04]:

Definition 2.8 (t-Composable Obfuscation [LPS04]). *A* PPT *machine $\mathcal{O}$ is a t-composable obfuscator for a circuit ensemble $\mathcal{C} = \{\mathcal{C}_\lambda\}_{\lambda\in\mathbb{N}}$ if it satisfies the functionality and polynomial slow-down requirements, as in Definition 2.1, and for any* PPT *distinguisher A and polynomial p, there is a simulator* Sim*, such that for any sequence of circuits $C^1, \ldots, C^t \in C_\lambda$ (where $t = \text{poly}(\lambda)$), and any sufficiently large λ:*

$$\left|\Pr\left[\mathcal{A}(\mathcal{O}(C^1), \ldots, \mathcal{O}(C^t)) = 1\right] - \Pr\left[\mathsf{Sim}^{C^1,\ldots,C^t}(1^{|C^1|}, \ldots, 1^{|C^t|}) = 1\right]\right| \leq \frac{1}{p(\lambda)}$$

where oracle $C^1, \ldots, C^t$ gets as input (x, i) and returns $C^i(x)$.

Note that while [LPS04] consider t-composability in the VBB setting, we only require the relaxed VGB setting, that is, we allow the simulator to run in unbounded time. Interestingly, while VBB obfuscation in the presence of auxiliary input (AI) is a seemingly stronger requirement than plain VBB obfuscation, Bitansky and Canetti show that AI does not add any power to VGB. Note that in this setting we can only allow auxiliary input that statistically hides the target points, as the simulator could otherwise trivially recover the obfuscated points from the auxilliary input.

Proposition 2.9 ([BC10]). *Let $\mathcal{O}$ be a VGB obfuscator for a circuit ensemble $\mathcal{C} = \{\mathcal{C}_\lambda\}_{\lambda\in\mathbb{N}}$. Then $\mathcal{O}$ is also a VGB obfuscator with (statistically unpredictable) auxiliary input for the ensemble.*

Bitansky and Canetti, furthermore, show that the point obfuscation scheme of Canetti [Can97] is a t-composable VGB point obfuscator under the t-Strong Vector Decision Diffie Hellman assumption [BC10]. Note that, as we can first compose and then introduce auxiliary input, this implies that under the t-Strong Vector Decision Diffie Hellman assumption Canetti's obfuscation scheme is also a VGB-AI point obfuscator. We recall the scheme by Canetti [Can97] in Appendix A.

From VGB back to AIPO. In this work we develop techniques to work with AIPOs. In a recent work, Matsuda and Hanoka [MH14] relate the notions of VGB point obfuscators (resp. VGB-AI point obfuscators) and AIPO and show that composable VGB-AI point obfuscators imply the existence of composable AIPO with respect to statistically unpredictable distributions (Matsuda et al. call this notion AIND-δ-sPUAI [MH14]). Statistically unpredictable distributions are defined analogously to unpredictable distributions (Definition 2.6) with the exception that we allow the predictor to run in unbounded time.

2.2 Universal Computational Extractors (UCE)

The UCE Framework by Bellare, Hoang, and Keelveedhi (BHK; [BHK13a]) introduces assumptions that allow us to instantiate random oracles in a wide range of applications and which are not succeptible to the impossibility result by Canetti, Goldreich and Halevi [CGH98]. Loosely speaking, UCEs are PRF-like assumptions that split the distinguisher into two parts: a first adversary S that gets access to a keyed hash function or a random oracle (and which is called the *source*), and a second adversary $\mathcal{D}$ that gets the hash key hk (and which is called the *distinguisher*). The two algorithms together try to guess whether the source was given access to a keyed hash function or to a random oracle.

Concretely, the UCE notions are defined via a two-stage UCE game (we depict the communication flow in Figure 1 and the pseudocode in Figure 2). First, the source S is run with oracle access to Hash to output some leakage L. Subsequently, distinguisher D is run on the leakage L and hash key hk but without access to oracle Hash. Distinguisher D outputs a single bit b indicating whether oracle Hash implements a random oracle or hash function H with key hk.

Without any restrictions, (S,D) can easily win the UCE game. For example, say, source S makes a random query x to receive $y \leftarrow \textsc{Hash}(x)$ and outputs (x, y) as leakage. As distinguisher D knows the hash key hk as well as the leakage (x, y), it can recompute the hash value and check whether $y = \mathsf{H}(\mathsf{hk}, x)$. BHK present several possible restrictions on the source which give rise to various UCE notions.

It turns out to be particularly useful to restrict sources to be computationally unpredictable, that is, the leakage created by the source S —when interacting with a random oracle— should not reveal (computationally) any of the source's queries to Hash. This notion is denoted by UCE[$\mathcal{S}^{\mathrm{cup}}$], where $\mathcal{S}^{\mathrm{cup}}$ denotes the class of computationally unpredictable sources [BHK13b].[2] BHK show that UCE[$\mathcal{S}^{\mathrm{cup}}$]-secure hash functions can safely replace a random oracle in a large number of interesting applications such as hardcore functions or deterministic public-key encryption [BHK13a]. We next give a formal definition of UCEs.

Formal UCE Definition. In line with [BST13] we consider families of functions F consisting of algorithms $F.\mathsf{KGen}$, $F.\mathsf{kl}$, $F.\mathsf{Eval}$, $F.\mathsf{il}$ and $F.\mathsf{ol}$. Algorithm $F.\mathsf{KGen}$ is a PPT algorithm taking the

[2]The notion was originally named UCE1 and later changed to UCE[$\mathcal{S}^{\mathrm{cup}}$] [BHK13a, BHK13b].

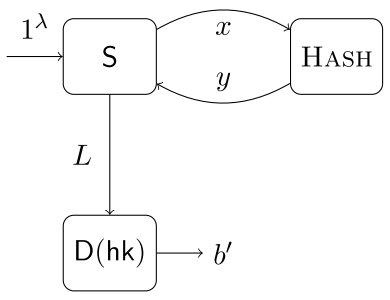

Figure 1: Schematic of the UCE game.

```
Main UCE$_{\mathsf{H}}^{\mathsf{S,D}}(\lambda)$
$b \leftarrow\!\!\$ \{0,1\}$; $\mathsf{hk} \leftarrow\!\!\$ \mathsf{H.KGen}(1^\lambda)$
$L \leftarrow\!\!\$ \mathsf{S}^{\textsc{Hash}}(1^\lambda)$
$b' \leftarrow\!\!\$ \mathsf{D}(1^\lambda, \mathsf{hk}, L)$
return $(b = b')$

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

```
Main $\mathsf{Pred}_{\mathsf{S}}^{\mathsf{P}}(\lambda)$
$\mathsf{done} \leftarrow$ false; $Q \leftarrow \emptyset$
$L \leftarrow\!\!\$ \mathsf{S}^{\textsc{Hash}}(1^\lambda)$; $\mathsf{done} \leftarrow$ true
$Q' \leftarrow\!\!\$ \mathsf{P}^{\textsc{Hash}}(1^\lambda, L)$
return $(Q \cap Q' \neq \emptyset)$

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

Figure 2: The UCE security game together with the unpredictability game (on the right). In the UCE game source S has access to Hash, which returns real or ideal hash values, and leaks L to a distinguisher D. The latter additionally gets the hash key and outputs a bit b'. On the right we give the unpredictability game.

security parameter 1^λ and outputting a key $k \in \{0,1\}^{F.\mathsf{kl}(\lambda)}$ where $F.\mathsf{kl} : \mathbb{N} \to \mathbb{N}$ denotes the key length. Functions $F.\mathsf{il} : \mathbb{N} \to \mathbb{N}$ and $F.\mathsf{ol} : \mathbb{N} \to \mathbb{N}$ denote the input and output length functions associated to F and for any $x \in \{0,1\}^{F.\mathsf{il}(\lambda)}$ and $k \leftarrow\!\!\$ F.\mathsf{KGen}(1^\lambda)$ we have that $F.\mathsf{Eval}(k,x) \in \{0,1\}^{F.\mathsf{ol}(\lambda)}$, where the PPT algorithm $F.\mathsf{Eval}$ denotes the "evaluation" function associated to F.

We denote hash functions by H. Let $\mathsf{H} = (\mathsf{H.KGen}, \mathsf{H.Eval}, \mathsf{H.kl}, \mathsf{H.il}, \mathsf{H.ol})$ be a hash-function family and let (S, D) be a pair of PPT algorithms. We define the UCE advantage of a pair (S, D) against H through

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

where game $\mathrm{UCE}_{\mathsf{H}}^{\mathsf{S,D}}(\lambda)$ is shown in Figure 2 on the left (in Figure 1 we give a schematic overview of the communication within the game).

Unpredictability. Without any further restrictions there are PPT pairs (S, D) that achieve an advantage in the $\mathrm{UCE}_{\mathsf{H}}^{\mathsf{S,D}}(\lambda)$ game close to 1. BHK define several possible restrictions for sources yielding various flavors of UCE assumptions [BHK13a]. Here, we are interested in a strengthened version of the original *computational* unpredictability [BHK13a] restriction.

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

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

is negligible, where game $\mathsf{Pred}_{\mathsf{S}}^{\mathsf{P}}(\lambda)$ is shown in Figure 2 on the right.[3] In line with [BHK13b], we call the class of all computationally unpredictable sources $\mathcal{S}^{\mathrm{cup}}$, where $\mathcal{S}^{\mathrm{cup}}$ denotes the class (set) of all computationally unpredictable sources. Similarly, we define the class of statistically unpredictable sources where the predictor in game $\mathsf{Pred}_{\mathsf{S}}^{\mathsf{P}}(\lambda)$ can run in unbounded time but is still restricted to only polynomially many oracle queries. The class of statistically unpredictable sources is denoted by $\mathcal{S}^{\mathrm{sup}}$.

[3]We here only present the simple unpredictability definition which BHK show to be equivalent to the full unpredictability notion, but which is easier to work with.

UCE Security. We say a hash function H is UCE secure for sources $\mathsf{S} \in \mathcal{S}$ denoted by UCE[$\mathcal{S}$], if for all PPT sources $\mathsf{S} \in \mathcal{S}$ and all PPT distinguishers D the advantage $\mathsf{Adv}^{\mathsf{uce}}_{\mathsf{H,S,D}}(\lambda)$ is negligible. In that way we get the UCE assumptions UCE[$\mathcal{S}^{\mathrm{cup}}$] and UCE[$\mathcal{S}^{\mathrm{sup}}$].

2.3 Puncturable PRFs

Besides point function obfuscation schemes, our main ingredient in the upcoming proofs are so-called puncturable pseudorandom functions (PRF) [SW13]. A family of puncturable PRFs $G :=(G.\mathsf{KGen}, G.\mathsf{Puncture}, G.\mathsf{kl}, G.\mathsf{Eval}, G.\mathsf{il}, G.\mathsf{ol})$ consists of functions that specify input length, output length and key length as well as a key generation algorithm $k \leftarrow G.\mathsf{KGen}$, a deterministic evaluation algorithm $G.\mathsf{Eval}(k, x)$ that takes a key k, an input x of length $G.\mathsf{il}(1^\lambda)$ and outputs a value y of length $G.\mathsf{ol}(1^\lambda)$. Additionally, there is a PPT puncturing algorithm $G.\mathsf{Puncture}$ which on input a polynomial-size set $S \subseteq \{0,1\}^{G.\mathsf{il}(\lambda)}$, outputs a special key k_S. A family of functions is called puncturable PRF if the following two properties are observed

- **Functionality preserved under puncturing.** For every PPT adversary $\mathcal{A}$ such that $\mathcal{A}(1^\lambda)$ outputs a polynomial-size set $S \subseteq \{0,1\}^{G.\mathsf{il}(\lambda)}$, it holds for all $x \in \{0,1\}^{G.\mathsf{il}(\lambda)}$ where $x \notin S$ that:
$$\Pr\Big[G.\mathsf{Eval}(k,x) = G.\mathsf{Eval}(k_S,x) : k \leftarrow\!\!\$\; G.\mathsf{KGen}(1^\lambda), k_S \leftarrow\!\!\$\; G.\mathsf{Puncture}(k,S)\Big] = 1$$
- **Pseudorandom at punctured points.** For every PPT adversary $(\mathcal{A}_1, \mathcal{A}_2)$ such that $\mathcal{A}_1(1^\lambda)$ outputs a set $S \subseteq \{0,1\}^{G.\mathsf{il}(\lambda)}$ and state σ, consider an experiment where $k \leftarrow G.\mathsf{KGen}(1^\lambda)$ and $k_S = G.\mathsf{Puncture}(k,S)$. Then we have
$$\Big|\Pr\Big[\mathcal{A}_2(\sigma, k_S, S, G.\mathsf{Eval}(k,S)) = 1\Big] - \Pr\Big[\mathcal{A}_2(\sigma, k_S, S, U_{G.\mathsf{ol}(\lambda)\cdot|S|}) = 1\Big]\Big| \le \mathrm{negl}(\lambda)$$
where $\mathsf{Eval}(k,S)$ denotes the concatenation of $\mathsf{Eval}(k,x_1), \ldots, \mathsf{Eval}(k,x_k)$ where $S = \{x_1,\ldots,x_k\}$ is the enumeration of the elements of S in lexicographic order, negl is a negligible function, and U_ℓ denotes the uniform distribution over $\{0,1\}^\ell$.

As observed by [BW13, BGI14, KPTZ13] puncturable PRFs can, for example, be constructed from pseudorandom generators via the GGM tree-based construction [GGM84]. Note that, as AIPO implies one-way functions (see Lemma B.1) AIPO, thus, also implies the existence of puncturable PRFs.

3 UCEs from iO and point obfuscation

In this section we present our constructions of UCEs from iO and AIPO. We first define the precise UCE notions that our constructions achieve and introduce the UCE restriction of *strong unpredictability.* We will then in Section 3.2 present a construction of a UCE-secure function with respect to sources which are strongly computationally unpredictable and which make exactly one oracle query. In Section 3.3 we will show how to extend the construction to allow for a constant number of queries by switching to a statistical version of strong unpredictability.

Interestingly, our construction for both cases is basically the same modulo circuit padding. That is, our constructions depend on an obfuscation of a circuit, which in both cases is the same but padded to a different length. A larger, but functionally equivalent, circuit seems to be necessary to allow for multiple source queries.

We will discuss applications of our constructions in Section 4. In Section 5 we will discuss why our construction does not (seem to) fall pray to the BFM attacks on computationally unpredictable sources [BFM14].

```
MAIN $\mathsf{stPred}_{\mathsf{S}}^{\mathsf{P}}(\lambda)$
$X^*, Y^* \leftarrow \{\ \}$
$b \leftarrow\$ \{0,1\}$
$L \leftarrow\$ \mathsf{S}^{\mathrm{HASH}}(1^\lambda)$
$x' \leftarrow\$ \mathsf{P}^{\mathrm{HASH}}(1^\lambda, L, Y^*)$
return $(x' \in X^*)$

HASH$(x)$
$X^* \leftarrow X^* \cup \{x\}$
$y \leftarrow\$ \{0,1\}^{\mathsf{H.ol}(\lambda)}$
$Y^* \leftarrow Y^* \cup \{y\}$
return $y$
```

```
Splt SOURCE $\mathsf{S}^{\mathrm{HASH}}(1^\lambda)$
$(L_0, \mathbf{x}) \leftarrow\$ \mathsf{S}_0(1^\lambda)$
for $i = 1, \ldots, |\mathbf{x}|$ do $\mathbf{y}[i] \leftarrow\$ \mathrm{HASH}(\mathbf{x}[i])$
$L_1 \leftarrow\$ \mathsf{S}_1(1^\lambda, \mathbf{y}); L \leftarrow (L_0, L_1)$
return $L$
```

Figure 3: On the left: the strong unpredictability game where the predictor, in addition to the leakage is also given the result of the HASH queries. Note that we only consider sources S that make a single query to HASH. On the right: the definition of split sources [BHK13b]. A split source $\mathsf{S} = \mathsf{Splt}[S_0, S_1]$ consists of two parts S_0 and S_1 that jointly generate leakage L and neither part gets direct oracle access to HASH.

3.1 Strongly unpredictable and q-query sources

We now introduce the precise source restrictions for our upcoming UCE constructions. We define a new restriction that we call *strong unpredictability* and which can be seen as either a stronger form of unpredictability or a relaxed version of split sources. Secondly, we consider sources that make only a bounded number of oracle queries.

STRONG UNPREDICTABILITY. We consider sources which are strongly unpredictable both in the computational and in the statistical sense. We denote by $\mathcal{S}^{\text{s-cup}}$ the class of sources which are strongly, computationally unpredictable and by $\mathcal{S}^{\text{s-sup}}$ the class of strongly, statistically unpredictable sources.

Strong unpredictability is a stronger requirement than unpredictability and we require that the leakage hides queries to HASH even if the predictor is given the query results. We say that a source S is called *strongly computationally unpredictable* if the advantage of any PPT predictor P, defined by

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

is negligible, where game $\mathsf{stPred}_{\mathsf{S}}^{\mathsf{P}}(\lambda)$ is shown in Figure 3. For the case of strongly statistically unpredictable sources ($\mathcal{S}^{\text{s-sup}}$) we allow the predictor to be unbounded in its running time, but restrict the number of oracle queries to be bounded polynomially.

In order to circumvent the BFM attacks on computationally unpredictable sources BHK introduce the notion of split sources [BHK13b]. A source S is called split source, denoted by $\mathsf{S} \in \mathcal{S}^{\text{splt}}$ if it can be decomposed into two algorithms S_0 and S_1 such that neither part gets direct access to oracle HASH. We give the pseudocode of split sources in Figure 3 on the right. In a first step algorithm S_0 outputs a leakage string L_0 together with a vector $\mathbf{x}$. Then, each of the entries in x is queried to HASH and the results stored in vector $\mathbf{y}$. The second algorithm S_1 is then run on vector $\mathbf{y}$ to produce the second part of the leakage L_1.

It is easily established that split sources are a (strict) subclass of strongly unpredictable sources, that is, $\mathcal{S}^{\text{splt}} \cap \mathcal{S}^{\text{cup}} \subsetneq \mathcal{S}^{\text{s-cup}}$ (and similarly in the statistical case $\mathcal{S}^{\text{splt}} \cap \mathcal{S}^{\text{sup}} \subsetneq \mathcal{S}^{\text{s-sup}}$). For this note that the leakage L_0 of the first algorithm of a split source is independent of any oracle answers. Similarly, if the oracle is implemented by a random oracle (which is the case in the unpredictability experiment) then the leakage L_1 of the second algorithm is independent of any actual oracle query. The inclusion is strict. Consider, for example, a source that queries HASH on x to receive y to then output $PRF_x(y)$ that is the image of a pseudorandom function at point y under key x. This distribution could not be simulated by a split source. We thus get

Lemma 3.1. *The class of split sources is a strict subclass of strongly unpredictable sources:*

$$\mathcal{S}^{\text{splt}} \cap \mathcal{S}^{\text{cup}} \subsetneq \mathcal{S}^{\text{s-cup}} \qquad \textit{and} \qquad \mathcal{S}^{\text{splt}} \cap \mathcal{S}^{\text{sup}} \subsetneq \mathcal{S}^{\text{s-sup}}$$

Q-QUERY UCE. Our first construction will only admit sources which make exactly one query. We call such sources single-query sources and denote the corresponding source class by $\mathcal{S}^{\text{1-query}}$. We will later relax this notion to allow for any constant number of queries. We call the corresponding sources q-query sources and denote their source class by $\mathcal{S}^{q\text{-query}}$. We note that sources restricted to a constant number of queries are also discussed in [BHK13b].

3.2 A UCE construction secure against sources in $\mathcal{S}^{\text{s-cup}} \cap \mathcal{S}^{\text{1-query}}$

We will now present our construction which depending on different assumptions on the existence of point obfuscators will achieve UCE[$\mathcal{S}^{\text{s-cup}} \cap \mathcal{S}^{\text{1-query}}$]-security or UCE[$\mathcal{S}^{\text{s-sup}} \cap \mathcal{S}^{q\text{-query}}$]-security. Note that depending on the number of supported queries the construction needs to pad the circuit before obfuscating it.

Construction 3.2. *Let $s : \mathbb{N} \to \mathbb{N}$, let G be a puncturable PRF and let* iO *be an indistinguishability obfuscator for all circuits in $\mathcal{P}/\text{poly}$. We define our hash function family* H *as*

```
H.KGen(1^λ)
    $k \leftarrow\$ G.\mathsf{KGen}(1^\lambda)$
    $\mathsf{hk} \leftarrow\$ \mathsf{iO}(\mathsf{PAD}(s(\lambda), G.\mathsf{Eval}(k,\cdot)))$
    return hk

H.Eval(hk, x)
    $\overline{C} \leftarrow \mathsf{hk}$
    return $\overline{C}(x)$
```

where $\mathsf{PAD} : \mathbb{N} \times \{0,1\}^* \longrightarrow \{0,1\}^*$ *denotes a deterministic padding algorithm that takes as input an integer and a circuit and outputs a functionally equivalent circuit padded to length $s(\lambda)$.*[4]

That is, the key generation algorithm $\mathsf{H.KGen}(1^\lambda)$ runs $k \leftarrow G.\mathsf{KGen}(1^\lambda)$ and returns $\mathsf{iO}(G.\mathsf{Eval}(k,\cdot))$, i.e., an obfuscation of the evaluation circuit of PRF G with key k hardwired into it. Function H.Eval is basically a universal Turing machine which runs input x on the obfuscated circuit hk.

Theorem 3.3. *If G is a secure puncturable PRF, if* iO *is a secure indistinguishability obfuscator and if AIPO exists, then the hash function family* H *defined in Construction 3.2 is* UCE[$\mathcal{S}^{\text{s-cup}} \cap \mathcal{S}^{\text{1-query}}$]-*secure.*

We prove the theorem via a sequence of 5 games (depicted in Figure 4) where game Game_1 denotes the original UCE[$\mathcal{S}^{\text{s-cup}} \cap \mathcal{S}^{\text{1-query}}$] game with hidden bit b fixed to 1. We first present the games and subsequently the analysis of the individual game hops. Let $\mathsf{S} \in \mathcal{S}^{\text{s-cup}} \cap \mathcal{S}^{\text{1-query}}$.

Game_1**:** The first game is the original UCE[$\mathcal{S}^{\text{s-cup}} \cap \mathcal{S}^{\text{1-query}}$]-game. Here, the hash key hk is an obfuscation of the circuit $C_1[k](x) := G.\mathsf{Eval}(k,x)$ (see Figure 4) where k is a key for the puncturable PRF.

[4]Function s needs to be chosen in accordance with the puncturable PRF to allow for the required number of puncturings.

$$\begin{tikzcd}
\mathsf{Game}_1(\lambda) \arrow[r, bend left, "\mathsf{iO}"] & \mathsf{Game}_2(\lambda) \arrow[r, bend left, "\mathsf{PRF}"] & \mathsf{Game}_3(\lambda) \arrow[r, bend left, "\mathsf{iO}"] & \mathsf{Game}_4(\lambda) \arrow[r, bend left, "\mathsf{AIPO} + \mathsf{iO} + \text{[BCP14]}"] & \mathsf{Game}_5(\lambda)
\end{tikzcd}$$

```
$\mathsf{Game}_1(\lambda)$
    $x^*, y^* \leftarrow \perp$
    $k \leftarrow\$ G.\mathsf{KGen}(1^\lambda)$
    $L \leftarrow\$ \mathsf{S}^{\text{HASH}}(1^\lambda)$
    $\mathsf{hk} \leftarrow\$ \mathsf{iO}(C_1[k])$
    $b' \leftarrow\$ \mathsf{D}(1^\lambda, \mathsf{hk}, L)$
    return $(1 = b')$

$\mathsf{Game}_2(\lambda)$
    $x^*, y^* \leftarrow \perp$
    $k \leftarrow\$ G.\mathsf{KGen}(1^\lambda)$
    $L \leftarrow\$ \mathsf{S}^{\text{HASH}}(1^\lambda)$
    $p \leftarrow\$ \mathsf{AIPO}(x^*)$
    $k^* \leftarrow G.\mathsf{Puncture}(k, x^*)$
    $\mathsf{hk} \leftarrow\$ \mathsf{iO}(C_2[k^*, p, y^*])$
    $b' \leftarrow\$ \mathsf{D}(1^\lambda, \mathsf{hk}, L)$
    return $(1 = b')$

$\mathsf{Game}_3(\lambda)$
    $x^*, y^* \leftarrow \perp$
    $k \leftarrow\$ G.\mathsf{KGen}(1^\lambda)$
    $L \leftarrow\$ \mathsf{S}^{\text{HASH}}(1^\lambda)$
    $p \leftarrow\$ \mathsf{AIPO}(x^*)$
    $k^* \leftarrow G.\mathsf{Puncture}(k, x^*)$
    $\mathsf{hk} \leftarrow\$ \mathsf{iO}(C_2[k^*, p, y^*])$
    $b' \leftarrow\$ \mathsf{D}(1^\lambda, \mathsf{hk}, L)$
    return $(1 = b')$

$\mathsf{Game}_4(\lambda)$
    $x^*, y^* \leftarrow \perp$
    $k \leftarrow\$ G.\mathsf{KGen}(1^\lambda)$
    $L \leftarrow\$ \mathsf{S}^{\text{HASH}}(1^\lambda)$
    $p \leftarrow\$ \mathsf{AIPO}(x^*)$
    $\mathsf{hk} \leftarrow\$ \mathsf{iO}(C_3[k, p, y^*])$
    $b' \leftarrow\$ \mathsf{D}(1^\lambda, \mathsf{hk}, L)$
    return $(1 = b')$

$\mathsf{Game}_5(\lambda)$
    $x^*, y^* \leftarrow \perp$
    $k \leftarrow\$ G.\mathsf{KGen}(1^\lambda)$
    $L \leftarrow\$ \mathsf{S}^{\text{HASH}}(1^\lambda)$
    $\mathsf{hk} \leftarrow\$ \mathsf{iO}(C_4[k])$
    $b' \leftarrow\$ \mathsf{D}(1^\lambda, \mathsf{hk}, L)$
    return $(1 = b')$
```

```
HASH$(x)$   (Game$_1$)
    $x^* \leftarrow x$
    $y^* \leftarrow G.\mathsf{Eval}(k, x)$
    return $y^*$

HASH$(x)$   (Game$_2$)
    $x^* \leftarrow x$
    $y^* \leftarrow G.\mathsf{Eval}(k, x)$
    return $y^*$

HASH$(x)$   (Game$_3$)
    $x^* \leftarrow x$
    $y^* \leftarrow\$ \{0,1\}^{\mathsf{H.ol}(\lambda)}$
    return $y^*$

HASH$(x)$   (Game$_4$)
    $x^* \leftarrow x$
    $y^* \leftarrow\$ \{0,1\}^{\mathsf{H.ol}(\lambda)}$
    return $y^*$

HASH$(x)$   (Game$_5$)
    $x^* \leftarrow x$
    $y^* \leftarrow\$ \{0,1\}^{\mathsf{H.ol}(\lambda)}$
    return $y^*$
```

```
CIRCUIT $C_1[k](x)$
    return $G.\mathsf{Eval}(k, x)$

CIRCUIT $C_2[k^*, p, y^*](x)$
    if $p(x) = \perp$ then    // if $x \neq x^*$
        return $G.\mathsf{Eval}(k^*, x)$
    return $y^*$

CIRCUIT $C_3[k, p, y^*](x)$
    if $p(x) = \perp$ then
        return $G.\mathsf{Eval}(k, x)$
    return $y^*$

CIRCUIT $C_4[k](x)$
    return $G.\mathsf{Eval}(k, x)$
```

Figure 4: The games used in the proof of Theorem 3.3 on the top and the used circuits on the bottom. To highlight the changes from game to game we have marked the changed lines with a light gray background color. By $C[k](x)$ we denote that circuit C depends on k (during construction time) and takes x as input. The arrows above the games indicate the security reduction to get from Game_i to Game_{i+1}.

Game_2**:** Let x^* be the single query that the source S makes to its HASH oracle and let $y^* := G.\mathsf{Eval}(k, x^*)$. Game_2 is similar to Game_1 except that we puncture the PRF at x^*. Namely, the hash key hk does not consist of an obfuscation of $C_1[k]$ anymore, but rather of an obfuscation of the circuit $C_2[k^*, p, y^*]$. The circuits $C_1[k]$ and $C_2[k^*, p, y^*]$ are functionally equivalent. However, instead of the normal PRF key, C_2 uses a punctured PRF key k^* which is punctured at value x^* (or equivalently, at all values x where $p(x) = 1$). Here, p is computed as the AIPO obfuscation of the point function I_{x^*} and hence, $p(x) = 1$ if and only if x is equal to the single hash query x^* of the source. On input a value x, circuit $C_2[k^*, p, y^*]$ checks whether $p(x) = \perp$ (i.e., if $x \neq x^*$): if so, it returns $G.\mathsf{Eval}(k^*, x)$, otherwise it outputs y^*.

Game_3**:** The game is equivalent to Game_2 except that oracle HASH now samples y^* uniformly at random instead of invoking $G.\mathsf{Eval}(k, .)$. Note that $C_2[k^*, p, y^*]$ is parametrized by y^*.

Game_4**:** The game is equivalent to the previous game except that now an obfuscation of circuit $C_3[k, p, y^*]$ is used as hash key hk. Circuit $C_3[k, p, y^*]$ is identical to circuit $C_2[k^*, p, y^*]$, except that it uses the original PRF key k instead of the punctured key k^*. Note that circuits $C_3[k, p, y^*]$ and $C_2[k^*, p, y^*]$ have identical input-output behaviour.

Game_5**:** The game is equivalent to the previous game except that now an obfuscation of circuit $C_4[k]$ is used as hash key hk. Circuit $C_4[k]$ is our original circuit again, that is, $C_4[k](\cdot) := G.\mathsf{Eval}(k, \cdot)$. Game_5 is our intended target. It is the UCE-security game for our construction in the random oracle world (that is, oracle HASH implements a random oracle).

In Game_5 we are in an identical setting to the UCE-game with the hidden bit set to 0. That is, the HASH oracle answers with randomly chosen values independent of the hash key. Further note, that C_4 and C_1 are identical, that is they are as in the construction. Thus, we can write the advantage of an adversary (S,D) in the UCE-security game as

$$\begin{aligned}\mathsf{Adv}^{\mathsf{uce}}_{\mathsf{S},\mathsf{D},\mathsf{H}}(\lambda) &= \Pr\left[\mathrm{UCE}^{\mathsf{S},\mathsf{D}}_{\mathsf{H}}(\lambda)\;\Big|\, b=1\right] + \Pr\left[\mathrm{UCE}^{\mathsf{S},\mathsf{D}}_{\mathsf{H}}(\lambda)\;\Big|\, b=0\right] - 1 \\ &= \Pr\left[\mathsf{Game}^{\mathsf{S},\mathsf{D}}_{1}(\lambda)\right] - \Pr\left[\mathsf{Game}^{\mathsf{S},\mathsf{D}}_{5}(\lambda)\right] \\ &\le \sum_{i=1}^{4}\left|\Pr\left[\mathsf{Game}^{\mathsf{S},\mathsf{D}}_{i}(\lambda)\right] - \Pr\left[\mathsf{Game}^{\mathsf{S},\mathsf{D}}_{i+1}(\lambda)\right]\right|\end{aligned}$$

It remains to show that the individual games are negligibly close.

Analysis $\mathsf{Game}_1(\lambda)$ to $\mathsf{Game}_2(\lambda)$. In order to reduce to the security to iO, we show that, by construction, the circuits $C_1[k]$ and $C_2[k^*,p,y^*]$ compute the same function. If $p(x)=\perp$, then $C_2[k^*,p,y^*]$ returns $G.\mathsf{Eval}(k^*,x)$. If $p(x)=1$, then then $x=x^*$ and $C_2[k^*,p,y^*]$ returns $y^*=G.\mathsf{Eval}(k,x^*)=G.\mathsf{Eval}(k,x)$. Hence, on all inputs x, $C_2[k^*,p,y^*]$ returns $G.\mathsf{Eval}(k,x)$ and so does $C_1[k]$.

As the circuits $C_1[k]$ and $C_2[k^*,p,y^*]$ compute the same functionality, $\mathsf{iO}(C_1[k])$ and $\mathsf{iO}(C_2[k^*,p,y^*])$ are indistinguishable and we can bound the difference between games Game_1 and Game_2 by the distinguishing advantage against the indistinguishability obfuscator iO. We now formalize this intuition.

Firstly, let us externalize some of the variables that the games use and introduce a unified notation for Game_1 and Game_2. For $i\in\{1,2\}$, let $\mathsf{Game}_i[k,r,y^*,k^*](\lambda)$ be equal to the game $\mathsf{Game}_i(\lambda)$ where key k is chosen as PRF key, source S uses randomness r, which defines its single query to Hash, the result to that query is y^* and the punctured key is chosen as k^* (if such a punctured key exists, namely only in Game_2). Note that the query x^* of the source is well-defined by its randomness. We define $\mathcal{A}[k,r,y^*,k^*](C)$ to be an adversary against the indistinguishability obfuscator iO that gets a circuit C as input, where C is either an obfuscation of circuit $C_1[k]$ or an obfuscation of circuit $C_2[k^*,p,y^*]$, where p is a point obfuscation of the point generated by S with randomness r. Adversary $\mathcal{A}[k,r,y^*,k^*](C)$ runs source $\mathsf{S}(r)$ to get the query x^*, returns y^* on the single Hash query and receives leakage L. It then runs distinguisher D on input $(1^\lambda,C,L)$ and outputs whatever D outputs. We present the pseudo-code of the adversary in Figure 5:

If $C=C_1[k]$ then adversary $\mathcal{A}[k,r,y^*,k^*](C)$ perfectly simulates game $\mathsf{Game}_1[k,r,y^*](\lambda)$ and if $C=C_2[k^*,p,y^*]$ then the adversary simulates $\mathsf{Game}_2[k,r,y^*,k^*](\lambda)$. Thus, we can rewrite the difference between the two games' distributions

$$\Pr[\mathsf{Game}_1(\lambda)] - \Pr[\mathsf{Game}_2(\lambda)]$$

as

$$\begin{aligned}&\mathbb{E}_{k,r,y^*}\Big[\Pr[\mathsf{Game}_1[k,r,y^*](\lambda)]\Big] - \mathbb{E}_{k,r,y^*,k^*}\Big[\Pr[\mathsf{Game}_2[k,r,y^*,k^*](\lambda)]\Big] \\ &= \mathbb{E}_{k,r,y^*,k^*}\left[\Pr[\mathsf{Game}_1[k,r,y^*](\lambda)] - \Pr[\mathsf{Game}_2[k,r,y^*,k^*](\lambda)]\right] \\ &= \mathbb{E}_{k,r,y^*,k^*}\left[\Pr\Big[\mathcal{A}[k,r,y^*,k^*](1^\lambda,\mathsf{iO}(C_1[k]))=1\Big] - \Pr\Big[\mathcal{A}[k,r,y^*,k^*](1^\lambda,\mathsf{iO}(C_2[k^*,p,y^*]))=1\Big]\right] \\ &= \mathbb{E}_{k,r,y^*,k^*}\Big[\mathsf{Adv}^{\mathsf{io}}_{\mathsf{iO},\mathcal{A}[k,r,y^*,k^*],C_1[k],C_2[k^*,p,y^*]}(\lambda)\Big] \\ &\le \max_{k,r,y^*,k^*}\mathsf{Adv}^{\mathsf{io}}_{\mathsf{iO},\mathcal{A}[k,r,y^*,k^*],C_1[k],C_2[k^*,p,y^*]}(\lambda)\end{aligned}$$

```
iO ADVERSARY $\mathcal{A}[k, r, y^*, k^*](1^\lambda, C)$
    $L \leftarrow\$ \mathsf{S}(r)^{\textsc{Hash}[y^*]}(1^\lambda)$
    $b' \leftarrow\$ \mathsf{D}(1^\lambda, C, L)$
    return $b'$

HASH$[y^*](x)$
    return $y^*$

PRF ADVERSARY $\mathcal{A}_1(1^\lambda)$
    run $\mathsf{S}(1^\lambda)$ until it makes query $x^*$ and
      denote by $\sigma$ the state at this point
    $S \leftarrow \{x^*\}$
    return $(S, \sigma)$

$\mathcal{A}_2(1^\lambda, \sigma, k_S, S, \tau)$
    $x^* \leftarrow S$
    $L \leftarrow\$$ continue computation of $\mathsf{S}$ with state $\sigma$ and
          answer its query with $\tau$
    $p \leftarrow\$ \mathsf{AIPO}(I_{x^*})$
    $\mathsf{hk} \leftarrow\$ \mathsf{iO}(C_2[k_S, p, \tau])$
    $b' \leftarrow\$ \mathsf{D}(1^\lambda, \mathsf{hk}, L)$
    return $b'$
```

Figure 5: Pseudo-code of the iO adversary used in game transition from Game_1 to Game_2 on the left, and the puncturable PRF adversary used in the transition of Game_2 to Game_3 on the right.

By the security of the indistinguishability obfuscator, the advantage of any efficient adversary is negligible and, hence, also $\max_{k,r,y^*,k^*} \mathsf{Adv}^{\mathsf{io}}_{\mathsf{iO},\mathcal{A}[k,r,y^*,k^*],C_1[k],C_2[k^*,p,y^*]}(\lambda)$ is negligible.

ANALYSIS $\mathsf{Game}_2(\lambda)$ TO $\mathsf{Game}_3(\lambda)$. We reduce the difference between Game_2 and Game_3 to the security of the puncturable PRF G. We define an adversary $(\mathcal{A}_1, \mathcal{A}_2)$ against the puncturable PRF as follows. On input the security parameter, adversary $\mathcal{A}_1$ runs source $\mathsf{S}(1^\lambda)$ on the security parameter. When source S makes its single HASH query x^*, adversary $\mathcal{A}_1$ stops, outputs $\{x^*\}$ as puncture set S together with the current state σ of the source S. Adversary $\mathcal{A}_2$ gets as input state σ, the punctured key $k^* = k_S$, the puncture point $\{x^*\} = S$ and a target value τ which is either $G.\mathsf{Eval}(k, x^*)$ or a uniformly random value. Adversary $\mathcal{A}_2$ uses the source's state σ to continue the simulation of the source S, which expects an answer from its HASH oracle. The adversary $\mathcal{A}_2$ passes value τ to S and receives leakage L. It then constructs an obfuscation $p \leftarrow\$ \mathsf{AIPO}(I_{x^*})$ as well as an obfuscation $\mathsf{hk} \leftarrow\$ \mathsf{iO}(C_2[k^*, p, \tau])$. Subsequently, it runs distinguisher D on input $(1^\lambda, \mathsf{hk}, L)$ and outputs whatever D outputs. We give the pseudo-code of the adversary in Figure 5.

If $\tau = G.\mathsf{Eval}(k, x^*)$, then adversary $(\mathcal{A}_1, \mathcal{A}_2)$ perfectly simulates Game_2 and otherwise it perfectly simulates Game_3. Thus, we have that

$$\Pr[\mathsf{Game}_2(\lambda)] - \Pr[\mathsf{Game}_3(\lambda)] \leq \mathsf{Adv}^{\mathsf{p\text{-}prf}}_{G,\mathcal{A}_1,\mathcal{A}_2}(\lambda)$$

which by the security of the puncturable PRF G is negligible.

ANALYSIS $\mathsf{Game}_3(\lambda)$ TO $\mathsf{Game}_4(\lambda)$. We show that the circuits $C_2[k^*, p, y^*]$ and $C_3[k, p, y^*]$ compute the same function. As the functionality of punctured PRFs is preserved under puncturing, for all x where $p(x) = \bot$, it holds that $C_2[k^*, p, y^*](x) = G.\mathsf{Eval}(k^*, x) = G.\mathsf{Eval}(k, x) = C_3[k, p, y^*]$. For x with $p(x) = 1$, by definition, both circuits $C_2[k^*, p, y^*](x)$ and $C_3[k, p, y^*]$ return y^*.

As both circuits are equal, $\mathsf{iO}(C_2[k^*, p, y^*])$ and $\mathsf{iO}(C_3[k, p, y^*])$ are indistinguishable by the security of indistinguishability obfuscation. Analogously to the first game hop, we get that

$$\Pr[\mathsf{Game}_3(\lambda)] - \Pr[\mathsf{Game}_4(\lambda)] \leq \max_{k,x^*,y^*,k^*} \mathsf{Adv}^{\mathsf{io}}_{\mathsf{iO},\mathcal{A}[k,x^*,y^*,k^*],C_2[k^*,p,y^*],C_3[k,p,y^*]}(\lambda) .$$

diO Sampler $\mathsf{Sam}(1^\lambda)$

```
$x^*, y^* \leftarrow \bot$
$k \leftarrow\$ G.\mathsf{KGen}(1^\lambda)$
$L \leftarrow\$ \mathsf{S}^{\text{HASH}}(1^\lambda)$
$p \leftarrow\$ \mathsf{AIPO}(I_{x^*})$
return $(C_3[k, p, y^*], C_4[k], L)$
```

Hash(x)

```
$x^* \leftarrow x$
$y^* \leftarrow\$ \{0,1\}^{\mathsf{H.ol}(\lambda)}$
return $y^*$
```

diO Adversary $\mathcal{A}(1^\lambda, C, L)$

```
$b' \leftarrow\$ \mathsf{D}(1^\lambda, C, L)$
return $b'$
```

AIPO Adversary $\mathcal{B}_1(1^\lambda)$

```
$x^*, y^* \leftarrow \bot$
$L \leftarrow\$ \mathsf{S}^{\text{HASH}}(1^\lambda)$
$k \leftarrow\$ G.\mathsf{KGen}(1^\lambda)$
$r \leftarrow\$ \{0,1\}^{\mathsf{H.il}(\lambda)}$
$b \leftarrow \langle r, x^* \rangle$
return $(x^*, (b, r, y^*, L))$
```

$\mathcal{B}_2(1^\lambda, b, r, k, L, p)$

```
$c \leftarrow \bot$
$\tau \leftarrow\$ \mathsf{Ext}(C_3[k, p, y^*], C_4[k], L)$
if $\tau = \bot$ then
    $c \leftarrow\$ \{0,1\}$
else if $\langle r, \tau \rangle = b$ then
    $c \leftarrow 1$
else $c \leftarrow 0$
return $c$
```

Figure 6: On the left, pseudo-code of the differing-inputs sampler Sam that we use in the transition from Game_3 to Game_4. We present the differing-inputs adversary $\mathcal{A}$ in the middle and the AIPO adversary $(\mathcal{B}_1, \mathcal{B}_2)$ on the right. Both of them are used in the game transition from Game_4 to Game_5. Note that the Hash oracle given to S in the description of $\mathcal{B}_1$ is equivalent to the Hash oracle used by sampler Sam on the left.

Analysis $\mathsf{Game}_4(\lambda)$ to $\mathsf{Game}_5(\lambda)$. By construction, the circuits $C_3[k, p, y^*]$ and $C_4[k]$ only differ on points where $p(x)$ is not equal to $\bot$, that is, they differ on a single point, which is the query point x^*. We will bound the difference between games Game_4 and Game_5 by the differing-inputs security of the indistinguishability obfuscator iO. For this, we build on a result by Boyle, Chung and Pass (here given as Theorem 2.5) who show that any indistinguishability obfuscator is also a differing-inputs obfuscator for differing-inputs circuits which differ on at most polynomially many points [BCP14]. As explained above, our circuits differ only on a single point and we can, thus, apply their theorem. In order to argue with the security property of differing-inputs obfuscation, we need to show that the family of circuit pairs $(C_3[k, p, y^*], C_4[k], \mathsf{Sam})$ is differing-inputs, where Sam is the circuit sampler that runs the same steps as game Game_4 up-to and including the obfuscation of the point function I_{x^*}, constructs circuits $C_3[k, p, y^*]$ and $C_4[k]$, and outputs $(C_3[k, p, y^*], C_4[k], L)$. We give the pseudo-code of sampler Sam in Figure 6.

Claim 3.4. *If p is a secure AIPO obfuscator, then the family of circuit pairs $(C_3[k, p, y^*], C_4[k], \mathsf{Sam})$ is differing-inputs.*

Before proving Claim 3.4, we show how to use it to prove that the difference between $\mathsf{Game}_4(\lambda)$ and $\mathsf{Game}_5(\lambda)$ is small. Theorem 2.5 by Boyle et al. [BCP14] says that, if a family is differing-inputs and only differs on at most polynomially many points, then their indistinguishability obfuscations are indistinguishable. Claim 3.4 establishes that the family $(C_3[k, p, y^*], C_4[k], \mathsf{Sam})$ is differing-inputs, and we already observed that circuits $C_3[k, p, y^*]$ and $C_4[k]$ only differ on a single input value. Hence, Theorem 2.5 allows us to do an analysis similar to the one from the first game hop. That is, we define adversary $\mathcal{A}$ which gets as input a circuit C and leakage L where C is either an indistinguishability obfuscation of circuit $C_3[k, p, y^*]$ or of circuit $C_4[k]$. It runs distinguisher D on input $(1^\lambda, C, L)$ and outputs whatever D outputs. We give the pseudo-code of adversary $\mathcal{A}$ in Figure 6.

If $C = \mathsf{iO}(C_3[k, p, y^*])$, then adversary $\mathcal{A}$ perfectly simulates $\mathsf{Game}_4(\lambda)$, and if $C = \mathsf{iO}(C_4[k])$, then it perfectly simulates $\mathsf{Game}_5(\lambda)$. Thus, we have

$$\Pr[\mathsf{Game}_4(\lambda)] - \Pr[\mathsf{Game}_5(\lambda)] \leq \mathsf{Adv}^{\mathsf{dio}}_{\mathsf{iO},\mathcal{A},C_3,C_4,\mathsf{Sam}}(\lambda) \leq \mathrm{negl}(\lambda)$$

We now proceed to proving Claim 3.4. Assume there exists an adversary (i.e., an extractor) Ext against the differing-inputs of the above circuit family which receives as input $(C_3[k,p,y^*], C_4[k], L)$ and outputs a value τ such that $C_3[k,p,y^*](\tau) \neq C_4[k](\tau)$. Then, $p(\tau) = 1$ and thus, intuitively, Ext breaks the AIPO property of the point obfuscation scheme. Let us now make this intuition formal.

We construct adversary $(\mathcal{B}_1, \mathcal{B}_2)$ where $\mathcal{B}_1$ describes an unpredictable distribution. On input the security parameter, $\mathcal{B}_1$ runs source $\mathsf{S}(1^\lambda)$ and answers its single $\textsc{Hash}$ query x^* with a uniformly random value y^* and then receives leakage L from S. $\mathcal{B}_1$ chooses draws a random string r. It then computes $b := \langle r, x^* \rangle$ and finally outputs $(x^*, (b, r, y^*, L))$.

Adversary $\mathcal{B}_2$ gets as input the security parameter, the auxiliary input (b, r, y^*, L) and an obfuscation p which is either an obfuscation of point function I_{x^*} or of a point function I_u for a uniformly random u. It samples a random key $k \leftarrow_\$ G.\mathsf{KGen}(\lambda)$ and constructs circuits $C_3[k,p,y^*]$ and $C_4[k]$. It then calls Ext on input $(C_3[k,p,y^*], C_4[k], L)$ to receive a value τ. If Ext outputs $\tau = \perp$, then $\mathcal{B}_2$ flips a bit and returns the outcome of the bitflip. Else, if τ is such that $C_3[k,p,y^*](\tau) \neq C_4[k](\tau)$ and hence $p(\tau) = 1$. $\mathcal{B}_2$ outputs 1 if $\langle r, \tau \rangle$ equals b and 0 otherwise.

If p is an obfuscation of I_{x^*} then circuits $C_3[k,p,y^*]$ and $C_4[k]$ differ on input τ if and only if $\tau = x^*$, unless $y^* = G.\mathsf{Eval}(k,x)$, which happens only with negligible probability. Hence, if the differing-inputs adversary Ext outputs τ, then $\tau = x^*$ and, thus, with probability 1, $\mathcal{B}_2$ will output 1. If, on the other hand, p is an obfuscation of I_u, then the circuits $C_3[k,p,y^*]$ and $C_4[k]$ differ on input τ if and only if $\tau = u$. Hence, if the differing-inputs adversary Ext outputs τ, then $\tau = u$ and thus, $\mathcal{B}$ will only output 1 with probability $\frac{1}{2}$ (since $\Pr[\langle u, r \rangle = b] = \frac{1}{2}$). Let us make the probability analysis formal. Let $d = 0$ describe the event that in the AIPO-game, I_{x^*} gets obfuscated, and $d = 1$ describe the event that in the AIPO-game, I_u gets obfuscated for a random u. Let ϵ be the probability that Ext returns a value $\tau \neq \perp$ in the differing-inputs game, that is, $\epsilon := \Pr[\perp \neq \mathsf{Ext} \mid d = 0]$. Note, that for readability we do not specify the input of adversaries Ext and $\mathcal{B}_2$ in the following treatment. We now consider the distinguishing probability of adversary $\mathcal{B}_2$

$$\Pr[\mathcal{B}_2 = 1 \mid d = 0] - \Pr[\mathcal{B}_2 = 1 \mid d = 1]$$

which can be rewritten as

$$\begin{aligned}
=&\Pr[\mathcal{B}_2 = 1 \mid d = 0, \mathsf{Ext} \neq \perp] \cdot \Pr[\mathsf{Ext} \neq \perp \mid d = 0] + \\
&\Pr[\mathcal{B}_2 = 1 \mid d = 0, \mathsf{Ext} = \perp] \cdot \Pr[\mathsf{Ext} = \perp \mid d = 0] - \\
&\Pr[\mathcal{B}_2 = 1 \mid d = 1] \\
=&\Pr[\mathsf{Ext} \neq \perp \mid d = 0] + \frac{1}{2} \cdot \Pr[\mathsf{Ext} = \perp \mid d = 0] - \Pr[\mathcal{B}_2 = 1 \mid d = 1] \\
=&\Pr[\mathsf{Ext} \neq \perp \mid d = 0] + \frac{1}{2} \cdot \Big(1 - \Pr[\mathsf{Ext} \neq \perp \mid d = 0]\Big) - \Pr[\mathcal{B}_2 = 1 \mid d = 1] \\
=&\frac{1}{2} \cdot \Pr[\mathsf{Ext} \neq \perp \mid d = 0] + \frac{1}{2} - \Pr[\mathcal{B}_2 = 1 \mid d = 1]
\end{aligned}$$

In the following, we consider the random variable U to describe the underlying choice of point function I_u (in case $d = 1$).

$$
\begin{aligned}
=&\frac{1}{2}\epsilon + \frac{1}{2} - \Pr[\,\mathcal{B}_2 = 1 \mid d = 1, \mathsf{Ext} \neq \bot\,] \cdot \Pr[\,\mathsf{Ext} \neq \bot \mid d = 1\,] + \\
&\qquad\qquad\qquad \Pr[\,\mathcal{B}_2 = 1 \mid d = 1, \mathsf{Ext} = \bot\,] \cdot \Pr[\,\mathsf{Ext} = \bot \mid d = 1\,] \\
=&\frac{1}{2}\epsilon + \frac{1}{2} - \\
&\frac{1}{2^{\mathsf{H.il}(\lambda)}} \sum_{u \in \{0,1\}^{\mathsf{H.il}(\lambda)}} \Big(\Pr[\,\mathcal{B}_2 = 1 \mid d = 1, U = u, \mathsf{Ext} \neq \bot\,] \cdot \Pr[\,\mathsf{Ext} \neq \bot \mid U = u, d = 1\,] + \\
&\qquad\qquad\qquad \Pr[\,\mathcal{B}_2 = 1 \mid d = 1, U = u, \mathsf{Ext} = \bot\,] \cdot \Pr[\,\mathsf{Ext} = \bot \mid U = u, d = 1\,] \Big)
\end{aligned}
$$

If extractor Ext outputs a value u (given that $d = 1$), then the probability of $\mathcal{B}_2$ of outputting 1, that is, $\Pr[\,\mathcal{B}_2 = 1 \mid d = 1, U = u, \mathsf{Ext} \neq \bot\,]$ is equivalent to $\Pr_{R,b}[\,\langle R, u\rangle = b]$ where random variable R denotes the choice of value r by $\mathcal{B}_1$ to compute $b = \langle r, x^*\rangle$. Note that extractor Ext is independent of R and b and, thus, we have that $\Pr_{R,b}[\,\langle R, u\rangle = b] = \frac{1}{2}$.

$$
\begin{aligned}
=&\frac{1}{2}\epsilon + \frac{1}{2} - \\
&\frac{1}{2^{\mathsf{H.il}(\lambda)}} \sum_{u \in \{0,1\}^{\mathsf{H.il}(\lambda)}} \Big(\Pr_{R,b}[\,\langle R, u\rangle = b] \cdot \Pr[\,\mathsf{Ext} \neq \bot \mid U = u, d = 1\,] + \frac{1}{2} \cdot \Pr[\,\mathsf{Ext} = \bot \mid U = u, d = 1\,] \Big) \\
=&\frac{1}{2}\epsilon + \frac{1}{2} - \frac{1}{2^{\mathsf{H.il}(\lambda)}} \sum_{u \in \{0,1\}^{\mathsf{H.il}(\lambda)}} \Big(\frac{1}{2} \cdot \Big(\Pr[\,\mathsf{Ext} \neq \bot \mid U = u, d = 1\,] + \Pr[\,\mathsf{Ext} = \bot \mid U = u, d = 1\,] \Big) \Big) \\
=&\frac{1}{2}\epsilon + \frac{1}{2} - \frac{1}{2^{\mathsf{H.il}(\lambda)}} \sum_{u \in \{0,1\}^{\mathsf{H.il}(\lambda)}} \frac{1}{2} \cdot 1 = \frac{1}{2}\epsilon
\end{aligned}
$$

To finish the proof of Claim 3.4, we need to argue that $\mathcal{B}_1$ implements an unpredictable distribution. By assumption, the source S is strongly computationally unpredictable (i.e., $\mathsf{S} \in \mathcal{S}^{\text{s-cup}}$) and hence leakage L hides x^* even in the presence of y^*. Thus, to see that $\mathcal{B}_1$ defines an unpredictable distribution, we need to argue that x^* remains unpredictable if additionally given a single bit of x^*. But a single bit can be guessed with probability $\frac{1}{2}$. Hence, $(\mathcal{B}_1, \mathcal{B}_2)$ breaks the security of the AIPO obfuscation, which concludes the proof of Claim 3.4 and the proof of Theorem 3.3.

3.3 A UCE construction secure against sources in $\mathcal{S}^{\text{s-sup}} \cap \mathcal{S}^{q\text{-query}}$

In this section we prove that our construction is also UCE-secure with respect to sources which are strongly unpredictable in a statistical sense and which allow the source to make q-many queries for any constant q. That is, we consider sources in class $\mathcal{S}^{\text{s-sup}} \cap \mathcal{S}^{q\text{-query}}$.

In case we allow the source to make q many queries, the first observation is that we need to choose the size of our obfuscated circuit such we can puncture at q many points. For each point, we will encode a random string into the circuit and thus, the circuit size grows with the number of point we puncture at. Besides this, the construction is identical to the one before with the exception that we need a different (incomparable) security property of our point function obfuscation scheme. That is, we require the point obfuscator to be a q-composable VGB obfuscator secure in the presence of statistically unpredictable auxiliary information (see Section 2.1).

Theorem 3.5. *Let q be a polynomial. If G is a secure puncturable PRF, if* iO *is a secure indistinguishability obfuscator and if there exist a q-composable VGB point obfuscator for statistically unpredicatable auxiliary input, then the hash function family* H *defined in Construction 3.2 is* $\mathrm{UCE}[\mathcal{S}^{\text{s-sup}} \cap \mathcal{S}^{q\text{-query}}]$*-secure.*

The proof follows analogously to the proof of Theorem 3.3, except for puncturing at several points instead of a single point and therefore, we reduce to t-composable VGB point obfuscation. The proof is deferred to Appendix C. For an overview of the game-hops, see Figure 9.

4 Applications

In the following section we describe the applications that our UCE constructions fulfill. Our $\mathrm{UCE}[\mathcal{S}^{\text{s-cup}} \cap \mathcal{S}^{1\text{-query}}]$-secure function can be shown to be a universal hardcore function for any one-way function and our $\mathrm{UCE}[\mathcal{S}^{\text{s-sup}} \cap \mathcal{S}^{q\text{-query}}]$-secure function achieves correlated-input security. We note that our $\mathrm{UCE}[\mathcal{S}^{\text{s-sup}} \cap \mathcal{S}^{q\text{-query}}]$ construction is also sufficient to instantiate proof-of-storage schemes and we refer to [BHK13b] for further details.

4.1 Hash functions secure under correlated inputs

Correlated-input secure hash functions (CIH) demand that an adversary that is able to obtain a sequence of (potentially correlated) hash values cannot distinguish between these being real or uniformly random assuming that source values come from a distribution that has super-logarithmic min-entropy. The notion was introduced by Goyal, O'Neill, and Rao [GOR11] and GOR present several constructions for limited CIH in the standard model. However, constructions for full CIH are only known in the random oracle model. Bellare, Hoang, and Keelvedhi show that hash functions secure under $\mathrm{UCE}[\mathcal{S}^{\text{sup}} \cap \mathcal{S}^{\text{splt}}]$-assumptions are also CIH secure [BHK13b]. With our construction from Section 3.3 we, thus, get the first candidate construction of CIH-secure hash functions.

Definitions. We present game $\mathrm{CIH}_{\mathsf{H}}^{\mathcal{A}}$ for correlated-input secure hash functions in Figure 7. We say that a function H is CIH-secure if the advantage of any admissible PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ defined as

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

is negligible. An adversary is admissible if on input the security parameter adversary $\mathcal{A}_1$ outputs a vector $\mathbf{m}$ of distinct values and of length $|\mathbf{m}| = v(\lambda)$ where v is a polynomial depending on $\mathcal{A}_1$. Furthermore, we require that the guessing probability of each entry is negligible, that is, the min-entropy of $[i]$ for all $i = 1, \ldots, v(\lambda)$ must be at least super-logarithmic in the security parameter.

Results. BHK give the following theorem [BHK13b]:

Theorem 4.1 ([BHK13b])**.** *If* H *is* $\mathrm{UCE}[\mathcal{S}^{\text{sup}} \cap \mathcal{S}^{\text{splt}}]$*-secure then* H *is a correlated-input hash function (CIH).*

In our construction of a $\mathrm{UCE}[\mathcal{S}^{\text{s-sup}} \cap \mathcal{S}^{q\text{-query}}]$ the number of queries that we can allow a source to make can be regarded as a parameter of the key generation function. We give the following adaption of CIH-secure hash functions called q-CIH-secure hash functions where the number of queries is specified as part of the key generation process.

Main $\mathrm{CIH}_{\mathsf{H}}^{\mathcal{A}}(\lambda)$

```
$b \leftarrow\$ \{0,1\}$
$\mathsf{hk} \leftarrow\$ \mathsf{H.KGen}(1^\lambda)$
$\mathbf{m} \leftarrow\$ \mathcal{A}_1(1^\lambda)$
for $i = 1, \dots, |\mathbf{m}|$ do
    $\mathbf{h}_0 \leftarrow\$ \{0,1\}^{\mathsf{H.ol}(\lambda)}$
    $\mathbf{h}_1 \leftarrow\$ \mathsf{H.Eval}(\mathsf{hk}, \mathbf{m}[i])$
$b' \leftarrow\$ \mathcal{A}_2(1^\lambda, \mathsf{hk}, \mathbf{h}_b)$
return $(b = b')$
```

Figure 7: The security game for correlated-input hash functions.

Main $\mathrm{HC}_{F,\mathsf{H}}^{\mathcal{A}}(\lambda)$

```
$b \leftarrow\$ \{0,1\}$
$k \leftarrow\$ F.\mathsf{KGen}(1^\lambda)$
$\mathsf{hk} \leftarrow\$ \mathsf{H.KGen}(1^\lambda)$
$x \leftarrow\$ \{0,1\}^\lambda$
$y \leftarrow F.\mathsf{Eval}(k, x)$
if $b = 1$ then $r \leftarrow\$ \{0,1\}^{\mathsf{H.ol}(\lambda)}$ else $r \leftarrow\$ \{0,1\}^{\mathsf{H.ol}(\lambda)}$
$b' \leftarrow\$ \mathcal{A}_2(k, \mathsf{hk}, y, r)$
return $(b = b')$
```

Figure 8: The security game for hardcore functions.

Definition 4.2. *Let q be a polynomial. A hash construction H is called q-CIH if its key generation algorithm takes as input parameter q and if the advantage of any adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ where $\mathcal{A}_1$ outputs a message vector of length at most q in the $\mathrm{CIH}_{\mathsf{H}}^{\mathcal{A}}$ game is negligible.*

Combining Theorems 3.5 and 4.1 we get:

Proposition 4.3. *Let q be a polynomial. If G is a secure puncturable PRF, if iO is a secure indistinguishability obfuscator and if there exist a q-composable VGB point obfuscator for statistically unpredicatable auxiliary input, then the hash function family defined in Construction 3.2 is q-CIH secure.*

4.2 Universal hardcore functions

A hardcore function for a one-way function f is a (deterministic) algorithm whose output on a random point x is indistinguishable from random even given $f(x)$. Random oracles are natural hardcore functions. BHK show that also UCE-secure functions, secure with respect to computationally unpredictable split sources are hardcore for any one-way function

Definitions. Let F be a (possibly keyed) one-way function. We say that function H is hardcore for F if the advantage of any PPT adversary $\mathcal{A}$ in game $\mathrm{HC}_{F,\mathsf{H}}^{\mathcal{A}}$ (given in Figure 8) is negligible, where we define the advantage as

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

Results. Bellare, Hoang, and Keelvedhi show that any UCE-secure function with respect to split sources that are computationally unpredictable are universal hardcore functions. We recall their result:

Theorem 4.4 ([BHK13b])**.** *If H is $\mathrm{UCE}[\mathcal{S}^{\mathrm{cup}} \cap \mathcal{S}^{\mathrm{splt}} \cap \mathcal{S}^{1\text{-query}}]$-secure then H is a hardcore function for any one-way function.*

Combined with Construction 3.2 and Theorem 3.3 we get an instantiation of a universal hardcore function.

Proposition 4.5. *If G is a secure puncturable PRF, if iO is a secure indistinguishability obfuscator and if AIPO exists, then the hash function family H defined in Construction 3.2 is a universal hardcore function.*

Bellare, Stepanovs, and Tessaro (BST;[BST13]) have recently given a similar result. That is, they have shown that the same construction is a universal hardcore function but based their security proof on a different assumption. Namely, they base their security proof on a weaker form of differing-inputs obfuscation called diO^- where the length of the auxiliary input is restricted thereby circumventing a conditional impossibility result on differing-inputs obfuscation by Garg et al. [GGHW13]. Thus, our result can be regarded as further evidence that Construction 3.2 is, indeed, a universal hardcore function, as the result holds if iO and one-way functions exist and as least one out of diO^- or AIPO exists.

The BR93 PKE scheme. In their seminal paper on random oracles, Bellare and Roggaway proposed a very simple, yet elegant public-key encryption scheme [BR93] based on a single trapdoor function f. The public-key of the encryption scheme is set to the evaluation key of the trapdoor function, and the private key is the set to be the trapdoor. To encrypt a message m the encryption algorithm chooses a random value x and outputs $(f(x), \mathsf{H}(x) \oplus m)$, where H is a hash function. For decryption, one inverts $f(x)$ using the trapdoor and can, thus, recover message m. Bellare et al. show that we can safely instantiate the hash function with any $\mathrm{UCE}[\mathcal{S}^{\mathrm{cup}} \cap \mathcal{S}^{\mathrm{splt}} \cap \mathcal{S}^{\text{1-query}}]$ [BHK13b]. Combined with Construction 3.2 and Theorem 3.3 we get an instantiation of that scheme.

Proposition 4.6. *If f is a trapdoor one-way function, if G is a secure puncturable PRF, if iO is a secure indistinguishability obfuscator and if AIPO exists, then the BR93 PKE scheme [BR93] instantiated with f and the hash function family H defined in Construction 3.2 is IND-CPA secure.*

5 The BFM impossibility result

In a recent work, Brzuska, Farshim, and Mittelbach [BFM14] show that assuming indistinguishability obfuscation then no standard model hash construction can be $\mathrm{UCE}[\mathcal{S}^{\mathrm{cup}}]$-secure, that is, UCE secure with respect to computationally unpredictable sources. In the following section we discuss the possibility of the BFM attack being extended to also cover UCE with respect to strongly unpredictable sources and implications thereof.

The BFM attack. Let us recall the intuition behind the BFM attack. Consider the source $\mathsf{S}_{\mathrm{BFM}}$ that makes a single (random) query x to receive $\textsc{Hash}(x)$. It then prepares an indistinguishability obfuscation of the circuit $(\mathsf{H.Eval}(\cdot, x) = y)$, that is the predicate that on input a hash key hk tests, if $\mathsf{H.Eval}(\mathsf{hk}, x) = y$. If y is chosen uniformly at random and if the output length of the hash function is (much) larger than the key-space, then the probability that there exists some key hk such that $\mathsf{H.Eval}(\mathsf{hk}, x) = y$ becomes negligible. This means that the circuit is with high probability the constant zero circuit and, thus, an indistinguishability obfuscation of the circuit does not leak anything more than the obfuscation of the constant zero circuit. It follows that the source is (computationally) unpredictable. A distinguisher, given access to the above circuit can, however, easily distinguish by simply plugging the hash key hk that it got as input into the circuit and outputting whatever the circuit outputs. We next give the pseudocode of the BFM adversary:

```
Source $\mathsf{S}_{\mathrm{BFM}}(1^\lambda)$
    $x \leftarrow\$ \{0,1\}^{\mathsf{H.il}(\lambda)}$
    $y \leftarrow \textsc{Hash}(x)$
    $C \leftarrow (\mathsf{H.Eval}(\cdot, x) = y)$
    $\tilde{C} \leftarrow\$ \mathsf{iO}(C)$
    return $L := \tilde{C}$
```

```
Distinguisher $\mathsf{D}_{\mathrm{BFM}}(1^\lambda, \mathsf{hk}, L)$
    $\tilde{C} \leftarrow L$
    $b' \leftarrow \tilde{C}(\mathsf{hk})$
    return $b'$
```

In the above intuition we relied upon the output length of the hash function being significantly larger than the key-size. To bootstrap their attack to hash functions for which this does not hold BFM simply extended the source to make multiple queries until the combined length of the received hash values is sufficiently longer than the size of the key. Bellare, Hoang, and Keelvedhi [BHK13c] point out that the attack can also be extended by applying a pseudorandom generator to the output of the hash construction. The idea here is, that if H is a UCE[$\mathcal{S}^{\mathrm{cup}}$]-secure function then so is $\mathsf{H}'(\cdot,\cdot) := \mathsf{PRG}(\mathsf{H}(\cdot,\cdot))$ where PRG is a pseudorandom generator.

Implausibility of extending BFM. While it is straight forward to prove that if H is a UCE[$\mathcal{S}^{\mathrm{cup}}$]-secure function then so is $\mathsf{H}'(\cdot,\cdot) := \mathsf{PRG}(\mathsf{H}(\cdot,\cdot))$ this is not the case if we restrict our sources to be strongly unpredictable, that is, to source class $\mathcal{S}^{\text{s-cup}}$. The reason is that in the reduction from a predictor to the PRG security the predictor requires the single oracle answer y as additional input which in the reduction would correspond to the seed of the PRG value. This, however, means that the reduction given either an image under the PRG or a uniformly random value must be able to compute the corresponding seed (if it exists) thereby breaking the PRG security on its own.

Similarly, using multiple queries seems not to allow extending the BFM attack to break our construction. In our construction the key is an obfuscation of a puncturable PRF. In order to use the puncturing technique the size of this circuit must be chosen according to the number of potential puncture points. Thus, the key size of our construction will always be strictly larger than the combined output length that can be achieved using the allowed number of queries.

Implications of an extended attack. Finally, we want to discuss the implications of a successful extension of the BFM attacks. In this case we would have the following implications:

1. $\mathsf{iO} \implies \neg\mathrm{UCE}[\mathcal{S}^{\text{s-cup}}]$
2. $\mathsf{iO} + \mathsf{AIPO} \implies \mathrm{UCE}[\mathcal{S}^{\text{s-cup}}]$

Combining the two would result in the statement

$$\mathsf{iO} \implies \neg\mathsf{AIPO}$$

that is, if indistinguishability obfuscation exists then point function obfuscation secure in the presence of auxiliary inputs does not exist and vice versa. This would be a rather surprising result as currently we hope that both forms of obfuscation exists. While iO has been used in numerous works lately [SW13, BCP14, ABG+13, GGHR14, HSW14, BZ13, BST13], point function obfuscation secure in the presence of auxilliary inputs has been used, for example, to circumvent black-box impossibility results and construct 3-round proofs with negligible soundness error satisfying the zero-knowledge notions *weak ZK* and *witness hiding* [GK96] and very recently to construct CCA secure public key encryption schemes [MH14].

An intriguing direction for further research is, thus, the study of point obfuscation with auxiliary inputs in the lights of the new results regarding indistinguishability obfuscation. Finally, let us note that in case AIPOs, indeed, do not exist that our result could be salvaged by considering a statistical version of strong unpredictability. This fix, was also proposed by BFM (and independently by BHK) to salvage a large number of applications for UCEs [BFM14].

Acknowledgments

We would like to especially thank Paul Baecher, Victoria Fehr and Giorgia Azzurra Marson for many helpful comments and discussions throughout the various stages of this work. This work was supported by CASED (`www.cased.de`).

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 3, 9, and 25.)

[AGIS14] Prabhanjan Ananth, Divya Gupta, Yuval Ishai, and Amit Sahai. Optimizing obfuscation: Avoiding barrington's theorem. Cryptology ePrint Archive, Report 2014/222, 2014. `http://eprint.iacr.org/`. (Cited on page 30.)

[BC10] Nir Bitansky and Ran Canetti. On strong simulation and composable point obfuscation. In Tal Rabin, editor, *Advances in Cryptology – CRYPTO 2010*, volume 6223 of *Lecture Notes in Computer Science*, pages 520–537, Santa Barbara, CA, USA, August 15–19, 2010. Springer, Berlin, Germany. (Cited on pages 6, 7, 8, and 11.)

[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 3, 9, 16, 19, 25, 30, and 33.)

[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, *CRYPTO 2014*, Lecture Notes in Computer Science, pages ??–??, Santa Barbara, CA, USA, August 17–21, 2014. Springer, Berlin, Germany. (Cited on pages 5, 7, 13, 24, and 25.)

[BGI$^+$01] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. In Joe Kilian, editor, *Advances in Cryptology – CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 1–18, Santa Barbara, CA, USA, August 19–23, 2001. Springer, Berlin, Germany. (Cited on pages 3, 7, 8, and 9.)

[BGI$^+$12] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. *J. ACM*, 59(2):6:1–6:48, May 2012. (Cited on pages 3, 7, and 8.)

[BGI14] Elette Boyle, Shafi Goldwasser, and Ioana Ivan. Functional signatures and pseudorandom functions. In Hugo Krawczyk, editor, *PKC 2014: 17th International Workshop on Theory and Practice in Public Key Cryptography*, volume 8383 of *Lecture Notes in Computer Science*, pages 501–519, Buenos Aires, Argentina, March 26–28, 2014. Springer, Berlin, Germany. (Cited on pages 13 and 32.)

[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 page 30.)

[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 4, 5, 11, and 12.)

[BHK13b] Mihir Bellare, Viet Tung Hoang, and Sriram Keelveedhi. Instantiating random oracles via UCEs. Cryptology ePrint Archive, Report 2013/424, 2013. `http://eprint.iacr.org/2013/424`. (Cited on pages 5, 11, 12, 14, 15, 22, 23, and 24.)

[BHK13c] Mihir Bellare, Viet Tung Hoang, and Sriram Keelveedhi. Personal communication. Sep, 2013. (Cited on page 25.)

[BP12] Nir Bitansky and Omer Paneth. Point obfuscation and 3-round zero-knowledge. In Ronald Cramer, editor, *TCC 2012: 9th Theory of Cryptography Conference*, volume 7194 of *Lecture Notes in Computer Science*, pages 190–208, Taormina, Sicily, Italy, March 19–21, 2012. Springer, Berlin, Germany. (Cited on pages 3, 7, 10, and 31.)

[BP13] Elette Boyle and Rafael Pass. Limits of extractability assumptions with distributional auxiliary input. Cryptology ePrint Archive, Report 2013/703, 2013. `http://eprint.iacr.org/2013/703`. (Cited on page 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 24.)

[BR13] Zvika Brakerski and Guy N. Rothblum. Obfuscating conjunctions. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology – CRYPTO 2013, Part II*, volume 8043 of *Lecture Notes in Computer Science*, pages 416–434, Santa Barbara, CA, USA, August 18–22, 2013. Springer, Berlin, Germany. (Cited on page 30.)

[BR14a] Zvika Brakerski and Guy N. Rothblum. Black-box obfuscation for d-CNFs. In Moni Naor, editor, *ITCS 2014: 5th Innovations in Theoretical Computer Science*, pages 235–250, Princeton, NJ, USA, January 12–14, 2014. Association for Computing Machinery. (Cited on page 30.)

[BR14b] 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 page 30.)

[BST13] Mihir Bellare, Igors Stepanovs, and Stefano Tessaro. Poly-many hardcore bits for any one-way function. Cryptology ePrint Archive, Report 2013/873, 2013. `http://eprint.iacr.org/`. (Cited on pages 3, 4, 6, 11, 24, and 25.)

[BW13] Dan Boneh and Brent Waters. Constrained pseudorandom functions and their applications. In Kazue Sako and Palash Sarkar, editors, *Advances in Cryptology – ASIACRYPT 2013, Part II*, volume 8270 of *Lecture Notes in Computer Science*, pages 280–300, Bengalore, India, December 1–5, 2013. Springer, Berlin, Germany. (Cited on pages 13 and 32.)

[BZ13] Dan Boneh and Mark Zhandry. Multiparty key exchange, efficient traitor tracing, and more from indistinguishability obfuscation. Cryptology ePrint Archive, Report 2013/642, 2013. `http://eprint.iacr.org/2013/642`. (Cited on pages 3, 6, and 25.)

[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 pages 3, 7, 10, 11, and 31.)

[CD08] Ran Canetti and Ronny Ramzi Dakdouk. Obfuscating point functions with multibit output. In Nigel P. Smart, editor, *Advances in Cryptology – EUROCRYPT 2008*, volume 4965 of *Lecture Notes in Computer Science*, pages 489–508, Istanbul, Turkey, April 13–17, 2008. Springer, Berlin, Germany. (Cited on pages 6 and 7.)

[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 page 11.)

[GGG+14] Shafi Goldwasser, S. Dov Gordon, Vipul Goyal, Abhishek Jain, Jonathan Katz, Feng-Hao Liu, Amit Sahai, Elaine Shi, and Hong-Sheng Zhou. Multi-input functional encryption. In Phong Q. Nguyen and Elisabeth Oswald, editors, *Advances in Cryptology – EUROCRYPT 2014*, volume 8441 of *Lecture Notes in Computer Science*, pages 578–602, Copenhagen, Denmark, May 11–15, 2014. Springer, Berlin, Germany. (Cited on page 3.)

[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, 8, and 30.)

[GGHR14] Sanjam Garg, Craig Gentry, Shai Halevi, and Mariana Raykova. Two-round secure MPC from indistinguishability obfuscation. In Yehuda Lindell, editor, *TCC 2014: 11th Theory of Cryptography Conference*, volume 8349 of *Lecture Notes in Computer Science*, pages 74–94, San Diego, CA, USA, February 24–26, 2014. Springer, Berlin, Germany. (Cited on pages 3, 25, and 30.)

[GGHW13] Sanjam Garg, Craig Gentry, Shai Halevi, and Daniel Wichs. On the implausibility of differing-inputs obfuscation and extractable witness encryption with auxiliary input. Cryptology ePrint Archive, Report 2013/860, 2013. `http://eprint.iacr.org/2013/860`. (Cited on pages 4 and 24.)

[GGM84] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions (extended abstract). In *25th Annual Symposium on Foundations of Computer Science*, pages 464–479, Singer Island, Florida, October 24–26, 1984. IEEE Computer Society Press. (Cited on page 13.)

[GK96] Oded Goldreich and Hugo Krawczyk. On the composition of zero-knowledge proof systems. *SIAM J. Comput.*, 25(1):169–192, February 1996. (Cited on page 25.)

[GK05] Shafi Goldwasser and Yael Tauman Kalai. On the impossibility of obfuscation with auxiliary input. In *46th Annual Symposium on Foundations of Computer Science*, pages 553–562, Pittsburgh, PA, USA, October 23–25, 2005. IEEE Computer Society Press. (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/`. (Cited on page 30.)

[Gol90] Oded Goldreich. A note on computational indistinguishability. *Information Processing Letters*, pages 277–281, 1990. (Cited on page 32.)

[GOR11] Vipul Goyal, Adam O'Neill, and Vanishree Rao. Correlated-input secure hash functions. In Yuval Ishai, editor, *TCC 2011: 8th Theory of Cryptography Conference*, volume 6597 of *Lecture Notes in Computer Science*, pages 182–200, Providence, RI, USA, March 28–30, 2011. Springer, Berlin, Germany. (Cited on pages 3 and 22.)

[HILL99] Johan Håstad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. A pseudorandom generator from any one-way function. *SIAM Journal on Computing*, 28(4):1364–1396, 1999. (Cited on page 32.)

[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 pages 3, 6, 25, and 30.)

[IL89] Russell Impagliazzo and Michael Luby. One-way functions are essential for complexity based cryptography (extended abstract). In *30th Annual Symposium on Foundations of Computer Science*, pages 230–235, Research Triangle Park, North Carolina, October 30 – November 1, 1989. IEEE Computer Society Press. (Cited on page 32.)

[KPTZ13] Aggelos Kiayias, Stavros Papadopoulos, Nikos Triandopoulos, and Thomas Zacharias. Delegatable pseudorandom functions and applications. In Ahmad-Reza Sadeghi, Virgil D. Gligor, and Moti Yung, editors, *ACM CCS 13: 20th Conference on Computer and Communications Security*, pages 669–684, Berlin, Germany, November 4–8, 2013. ACM Press. (Cited on pages 13 and 32.)

[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 10.)

[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 pages 11 and 25.)

[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 5.)

[PST13] Rafael Pass, Karn Seth, and Sidharth Telang. Indistinguishability obfuscation from semantically-secure multilinear encodings. Cryptology ePrint Archive, Report 2013/781, 2013. `http://eprint.iacr.org/`. (Cited on page 30.)

[SW13] Amit Sahai and Brent Waters. How to use indistinguishability obfuscation: Deniable encryption, and more. Cryptology ePrint Archive, Report 2013/454, 2013. `http://eprint.iacr.org/2013/454`. (Cited on pages 3, 4, 6, 13, 25, and 30.)

[Wee05] Hoeteck Wee. On obfuscating point functions. In Harold N. Gabow and Ronald Fagin, editors, *37th Annual ACM Symposium on Theory of Computing*, pages 523–532, Baltimore, Maryland, USA, May 22–24, 2005. ACM Press. (Cited on page 31.)

[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 pages 3 and 4.)

A Constructions and candidates for obfuscation schemes

In the following we present existing candidates for indistinguishability obfuscation (which also yield restricted forms of differing-inputs obfuscation with a beautiful result of Boyle, Chung and Pass [BCP14]). The constructions itself are, however, not relevant for this work. The assumptions they are based on, on the other hand, are very relevant as by extension our results imply that some of these (either the ones related to indistinguishability obfuscation or the assumptions underlying the constructions for point obfuscation) cannot hold.

Indistinguishability Obfuscation. In a breakthrough paper, Garg et al. [GGH$^+$13] present a candidate construction for indistinguishability obfuscation. Their candidate is based on an intractability assumption related to multi-linear maps and their construction yields an indistinguishability obfuscator for all circuits in $\mathcal{NC}^1$. They go on to show that, if additionally assuming a perfectly correct leveled fully homomorphic encryption scheme and a perfectly sound non-interactive witness-indistinguishable proof system that their obfuscator can be bootstrapped to yield an indistinguishability obfuscator for all circuits in $\mathcal{P}$/poly. In recent works, Brakerski and Rothblum [BR14b] and Barak et al. [BGK$^+$14] have further simplified the construction and showed that it is secure against all generic multi-linear attacks. A very recent result by Ananth et al. [AGIS14] yields a more efficient construction. Complementary, Pass, Seth and Telang [PST13] show how to base an adapted construction on a novel assumption they call *semantically-secure multilinear encodings*.

With their candidate construction and their insightful application of using iO to construct functional encryption for general circuits Garg et al. [GGH$^+$13] revived the interest in the study of obfuscation schemes and in particular in the study of indistinguishability obfuscation (see, for example, [SW13, BR13, HSW14, BR14a, BR14b, GGHR14, BGK$^+$14] and the references therein).

In a very recent work Gentry et al. [GLSW14] show that iO can be based on instance-independent assumptions and give a construction based on the “Multilinear Subgroup Elimination Assumption”.

Candidates for Point Obfuscation with Auxiliary Input (AIPO). The study of point function obfuscation started with Canetti [Can97] who gives a construction that satisfies Definition 2.7 under a strong variant of the DDH assumption. We here present the construction in the formulation of [BP12] and then present the assumption it is based on.

Construction A.1 (AIPO obfuscator due to [Can97]). *Let $\mathcal{G} := \{\mathbb{G}_\lambda\}_{\lambda\in\mathbb{N}}$ be a group ensemble, where each $\mathbb{G}_\lambda$ is a group of prime order $p_\lambda \in (2^{\lambda-1}, 2^\lambda)$. We define an obfuscator* AIPO *for points in the domain $\mathbb{Z}_{p_\lambda}$ as follows: $I_x \overset{\mathsf{AIPO}}{\mapsto} C(r, r^x)$, where $r \leftarrow\!\!\$\ \mathbb{G}_\lambda$ is a random generator of $\mathbb{G}_\lambda$, and $C(r, r^x)$ is a circuit which on input i, checks whether $r^x = r^i$.*

Assumption A.2 ([Can97],[BP12]). *There exists an ensemble of prime order groups $\mathcal{G} := \{\mathbb{G}_\lambda\}_{\lambda\in\mathbb{N}}$ such that for any unpredictable distribution $\mathcal{D} = \{D_\lambda = (Z_\lambda, Y_\lambda)\}_{\lambda\in\mathbb{N}}$ with support $\{0,1\}^{\mathrm{poly}(\lambda)} \times \mathbb{Z}_{p_\lambda}$, it holds that for all* PPT *algorithms $\mathcal{A}$ there exists a negligible function* negl *such that*

$$\left|\Pr_{r \leftarrow\$\ \mathbb{G}_\lambda, \leftarrow\$\ (z,x) \leftarrow\$\ D_\lambda}[\mathcal{A}(z, r, r^x) = 1] - \Pr_{r \leftarrow\$\ \mathbb{G}_\lambda, z \leftarrow\$\ Z_\lambda, u \leftarrow\$\ \mathbb{Z}_{p_\lambda}}[\mathcal{A}(z, r, r^u) = 1]\right| \leq \mathrm{negl}(\lambda)$$

A second candidate construction for AIPO is due to Bitansky and Paneth [BP12] who adapt a point obfuscation construction by Wee [Wee05] to allow for auxiliary input. Their construction is based on an assumption on the existence of strong pseudorandom permutations. Let us recall the underlying assumption (which generalizes the original assumption due to Wee [Wee05]) before recalling the construction.

Assumption A.3 ([BP12]). *There exists an ensemble of permutation families $\mathcal{F} = \{\mathcal{F}_\lambda = \{f\}\}$ such that for any unpredictable distribution ensemble $\mathcal{D} = \{D_\lambda = (Z_\lambda, Y_\lambda)\}_{\lambda\in\mathbb{N}}$, the following two distribution ensembles are also unpredictable:*

- $((Z_\lambda, f(Y_\lambda), f); Y_\lambda)$
- $((Z_\lambda, f); f(Y_\lambda))$,

where in both $f \leftarrow\!\!\$\ \mathcal{F}_\lambda$ (independently of D_λ).

Based on Assumption A.3, Bitansky and Paneth show that the following construction yields an AIPO obfuscator satisfying Definition 2.7 [BP12].

Construction A.4 ([BP12]). *Let $\mathcal{F}$ be a family of permutations as given by Assumption A.3. AIPO obfuscator* AIPO *works as follows: given a point $y \in \{0,1\}^\lambda$,* AIPO *samples 3λ permutations $\{f_i\}_{i\in[3\lambda]}$ from $\mathcal{F}_\lambda$ and 3λ strings $\{r_i\}_{i\in[3\lambda]}$ from $\{0,1\}^\lambda$. For every $i \in [3\lambda]$, let $f^i := f_i \circ f_{i-1} \circ \ldots \circ f_1$ (where $\circ$ denotes composition). Obfuscator* AIPO *outputs a circuit C_y that has hardcoded into it the randomness of* AIPO*, $\{f_i, r_i\}_{i\in[3\lambda]}$ and the bits $\{b_i := \langle r_i, f^i(y)\rangle\}_{i\in[3\lambda]}$, where $\langle ., .\rangle$ denotes the inner product over $\mathbb{GF}_2$. Circuit C_y outputs 1 on a point x if for all $i \in [3\lambda] : b_i = \langle r_i, f^i(x)\rangle$; and 0 otherwise.*

B AIPO implies one-way functions

In this section, we show that AIPOs imply one-way functions.

Lemma B.1 (AIPO implies one-way functions). *Point Function Obfuscation (that is secure under auxiliary inputs) implies puncturable PRFs.*

Proof. We show that, like most cryptographic primitives, AIPO implies one-way functions. As one-way functions imply PRGs [HILL99] and as PRGs imply puncturable PRFs [BW13, BGI14, KPTZ13], that suffices to prove that AIPO implies puncturable PRFs.

Two distributions that are statistically close and computationally far imply a distributional one-way function [Gol90], and a distributionally one-way function implies a standard one-way function [IL89]. The security property of AIPO implies that the obfuscation of I_x for a random x, where $x_1 = 0$ is indistinguishable from the obfucation of I_u for a random u. Hence, we have two computationally indistinguishable distributions. Let us argue that they are also statistically far. With probability $\frac{1}{2}$, the first bit of u does not equal 0 and hence, the obfuscation of u is outside of the support of the distributions over I_x. Hence, the two random variables have statistical distance at least $\frac{1}{2}$ which concludes the proof. □

C Proof of Theorem 3.5

The main differences compared to the proof of Theorem 3.3 is that we now puncture on q many points and also construct q many point obfuscations and hence, in the analogue to Claim 3.4 we need to use composable point obfuscations. We now describe the five games and then give the proof of the analogue to Claim 3.4. We present the pseudocode for the games in Figure 9.

Game_1**:** The first game is the original UCE[$\mathcal{S}^{\text{s-sup}} \cap \mathcal{S}^{q\text{-query}}$]-game. Here, the hash key hk is an obfuscation of the circuit $C_1[k](x) := G.\mathsf{Eval}(k, x)$ where k is a key for the puncturable PRF.

Game_2**:** Let $x_1^*, \ldots, x_q^*$ denote the q queries that the source S makes to its HASH oracle and let $y_i^* := G.\mathsf{Eval}(k, x_i^*)$ (for $i = 1, \ldots, q$). Game_2 is similar to Game_1 except that we puncture the PRF on x_1^* to x_i^*. Namely, the hash key hk does not consist of an obfuscation of $C_1[k]$ anymore, but rather of an obfuscation of the circuit $C_2[k^*, p_1, \ldots, p_q, y_1^*, \ldots, y_q^*]$. The circuits $C_1[k]$ and $C_2[k^*, p_1, \ldots, p_q, y_1^*, \ldots, y_q^*]$ are functionally equivalent. However, instead of the normal PRF key, C_2 uses a punctured PRF key k^* which is punctured at values x_i^* (for $i = 1, \ldots, q$). Here, p_i is computed as the point obfuscation of the point function $I_{x_i^*}$. On input a value x, circuit $C_2[k^*, p_1, \ldots, p_q, y_1^*, \ldots, y_q^*]$ checks whether there exists $i \in \{1, \ldots, q\}$ such that $p_i(x) = \perp$. If so, it returns y_i^*, otherwise it outputs $G.\mathsf{Eval}(k^*, x)$.

Game_3**:** The game is equivalent to Game_2 except that oracle HASH now samples y_i^* uniformly at random instead of invoking $G.\mathsf{Eval}(k, .)$. Note that $C_2[k^*, p_1, \ldots, p_q, y_1^*, \ldots, y_q^*]$ is parametrized by y_i^*.

Game_4**:** The game is equivalent to the previous game except that we now use an obfuscation of circuit $C_3[k, p_1, \ldots, p_q, y_1^*, \ldots, y_q^*]$ as hash key hk. Circuit $C_3[k, p_1, \ldots, p_q, y_1^*, \ldots, y_q^*]$ is identical to circuit $C_2[k^*, p_1, \ldots, p_q, y_1^*, \ldots, y_q^*]$, except that it uses the original PRF key k instead of the punctured key k^*. Note that circuits $C_3[k, p_1, \ldots, p_q, y_1^*, \ldots, y_q^*]$ and $C_2[k^*, p_1, \ldots, p_q, y_1^*, \ldots, y_q^*]$ have identical input-output behaviour.

Game_5**:** The game is equivalent to the previous game except that now an obfuscation of circuit $C_4[k]$ is used as hash key hk. Circuit $C_4[k]$ is our original circuit again, that is, $C_4[k](\cdot) := G.\mathsf{Eval}(k, \cdot)$. Game_5 is our intended target. It is the UCE-security game for our construction in the random oracle world (that is, oracle HASH implements a random oracle).

It remains to give the analogue of Claim 3.4.

$$\begin{tikzcd}\mathsf{Game}_1(\lambda) \arrow[r, bend left, "\mathsf{iO}"] & \mathsf{Game}_2(\lambda) \arrow[r, bend left, "\mathsf{PRF}"] & \mathsf{Game}_3(\lambda) \arrow[r, bend left, "\mathsf{iO}"] & \mathsf{Game}_4(\lambda) \arrow[r, bend left, "q\text{-comp. VGB} + \mathsf{iO} + \text{[BCP14]}"] & \mathsf{Game}_5(\lambda)\end{tikzcd}$$

```
Game_1(λ)
$X^*, Y^* \leftarrow []$
$i \leftarrow 0$
$k \leftarrow\$ G.\mathsf{KGen}(1^\lambda)$
$L \leftarrow\$ \mathsf{S}^{\textsc{Hash}}(1^\lambda)$
$\mathsf{hk} \leftarrow\$ \mathsf{iO}(C_1[k])$
$b' \leftarrow\$ \mathsf{D}(1^\lambda, \mathsf{hk}, L)$
return $(1 = b')$

Hash(x)
$i \leftarrow i + 1$
$X^*[i] \leftarrow x$
$Y^*[i] \leftarrow G.\mathsf{Eval}(k, x)$
return $Y^*[i]$
```

```
Game_2(λ)
$X^*, Y^* \leftarrow []$
$i \leftarrow 0$
$k \leftarrow\$ G.\mathsf{KGen}(1^\lambda)$
$L \leftarrow\$ \mathsf{S}^{\textsc{Hash}}(1^\lambda)$
$P \leftarrow []$
for $i = 1, \ldots, q$ do
    $P[i] \leftarrow\$ \mathcal{O}(X^*[i])$
$k^* \leftarrow G.\mathsf{Puncture}(k, X^*)$
$\mathsf{hk} \leftarrow\$ \mathsf{iO}(C_2[k^*, P, Y^*])$
$b' \leftarrow\$ \mathsf{D}(1^\lambda, \mathsf{hk}, L)$
return $(1 = b')$

Hash(x)
$i \leftarrow i + 1$
$X^*[i] \leftarrow x$
$Y^*[i] \leftarrow G.\mathsf{Eval}(k, x)$
return $Y^*[i]$
```

```
Game_3(λ)
$X^*, Y^* \leftarrow []$
$i \leftarrow 0$
$k \leftarrow\$ G.\mathsf{KGen}(1^\lambda)$
$L \leftarrow\$ \mathsf{S}^{\textsc{Hash}}(1^\lambda)$
$P \leftarrow []$
for $i = 1, \ldots, q$ do
    $P[i] \leftarrow\$ \mathcal{O}(X^*[i])$
$k^* \leftarrow G.\mathsf{Puncture}(k, X^*)$
$\mathsf{hk} \leftarrow\$ \mathsf{iO}(C_2[k^*, P, Y^*])$
$b' \leftarrow\$ \mathsf{D}(1^\lambda, \mathsf{hk}, L)$
return $(1 = b')$

Hash(x)
$i \leftarrow i + 1$
$X^*[i] \leftarrow x$
$Y^*[i] \leftarrow\$ \{0,1\}^{\mathsf{H.ol}(\lambda)}$
return $Y^*[i]$
```

```
Game_4(λ)
$X^*, Y^* \leftarrow []$
$i \leftarrow 0$
$k \leftarrow\$ G.\mathsf{KGen}(1^\lambda)$
$L \leftarrow\$ \mathsf{S}^{\textsc{Hash}}(1^\lambda)$
$P \leftarrow []$
for $i = 1, \ldots, q$ do
    $P[i] \leftarrow\$ \mathcal{O}(X^*[i])$
$\mathsf{hk} \leftarrow\$ \mathsf{iO}(C_3[k, P, Y^*])$
$b' \leftarrow\$ \mathsf{D}(1^\lambda, \mathsf{hk}, L)$
return $(1 = b')$

Hash(x)
$i \leftarrow i + 1$
$X^*[i] \leftarrow x$
$Y^*[i] \leftarrow\$ \{0,1\}^{\mathsf{H.ol}(\lambda)}$
return $Y^*[i]$
```

```
Game_5(λ)
$X^*, Y^* \leftarrow []$
$i \leftarrow 0$
$k \leftarrow\$ G.\mathsf{KGen}(1^\lambda)$
$L \leftarrow\$ \mathsf{S}^{\textsc{Hash}}(1^\lambda)$
$\mathsf{hk} \leftarrow\$ \mathsf{iO}(C_4[k])$
$b' \leftarrow\$ \mathsf{D}(1^\lambda, \mathsf{hk}, L)$
return $(1 = b')$

Hash(x)
$i \leftarrow i + 1$
$X^*[i] \leftarrow x$
$Y^*[i] \leftarrow\$ \{0,1\}^{\mathsf{H.ol}(\lambda)}$
return $Y^*[i]$
```

```
Circuit $C_1[k](x)$
    return $G.\mathsf{Eval}(k, x)$
```

```
Circuit $C_2[k^*, P, Y^*](x)$
    for $i = 1, \ldots, q$
        if $P[i](x) = 1$ then
            return $Y^*[i]$
    return $G.\mathsf{Eval}(k^*, x)$
```

```
Circuit $C_3[k, P, Y^*](x)$
    for $i = 1, \ldots, q$
        if $P[i](x) = 1$ then
            return $Y^*[i]$
    return $G.\mathsf{Eval}(k, x)$
```

```
Circuit $C_4[k](x)$
    return $G.\mathsf{Eval}(k, x)$
```

Figure 9: The games used in the proof of Theorem 3.5 on the top and the corresponding circuits on the bottom. Note that we use lists X^*, Y^* and P to store queries, answers and point functions.

Claim C.1. *If p is a q-composable VGB obfuscator, then the family of circuit pairs*

$$(C_3[k, p_1, \ldots, p_q, y_1^*, \ldots, y_q^*], C_4[k], \mathsf{Sam})$$

is differing-inputs.

Proof. Assume there exists an adversary (i.e., an extractor) Ext against the differing-inputs of the above circuit family which receives as input $(C_3[k, p_1, \ldots, p_q, y_1^*, \ldots, y_q^*], C_4[k], L)$ and outputs a value τ such that $C_3[k, p_1, \ldots, p_q, y_1^*, \ldots, y_q^*](\tau) \neq C_4[k](\tau)$. Then, $p_i(\tau) = 1$ for some $i \in \{1, \ldots, q\}$ and thus, intuitively, Ext breaks the security property of the point obfuscation scheme. Let us now make this intuition formal.

We construct adversary $(\mathcal{B}_1, \mathcal{B}_2)$ where $\mathcal{B}_1$ describes a statistically unpredictable distribution. On input the security parameter, $\mathcal{B}_1$ runs source $\mathsf{S}(1^\lambda)$. Without loss of generality we assume that the source's queries are distinct. Adversray $\mathcal{B}_1$ answers the q many distinct $\textsc{Hash}$ queries x_i^* each with a uniformly random value y_i^* and then receives leakage L from S. Adversary $\mathcal{B}_1$ then draws a random string r and index $j \leftarrow\$ \{1, \ldots, q\}$. It computes $b := \left\langle r, x_j^* \right\rangle$ and finally outputs $((x_1^*, \ldots, x_q^*), (b, r, j, y_1^*, \ldots, y_q^*, L))$.

Adversary $\mathcal{B}_2$ gets as input the security parameter, the auxiliary input $(b, r, j, y_1^*, \ldots, y_q^*, L)$, as well as q obfuscations $p_1, \ldots, p_q$ which are either honest obfuscations of point functions $I_{x_i^*}$ (for $i = 1, \ldots, q$) or of q uniformly random points. It samples a random key $k \leftarrow\$ G.\mathsf{KGen}(\lambda)$ and constructs circuits $C_3[k, p_1, \ldots, p_q, y_1^*, \ldots, y_q^*]$ and $C_4[k]$. It then calls the extractor Ext on input

$(C_3[k, p_1, \ldots, p_q, y_1^*, \ldots, y_q^*], C_4[k], L)$ to receive a value τ. If Ext outputs $\tau = \perp$, then $\mathcal{B}_2$ flips a bit and returns the outcome of the bitflip. Else, if τ is such that $C_3[k, p, y^*](\tau) \neq C_4[k](\tau)$ and $p_j(\tau) = 1$, then $\mathcal{B}_2$ outputs 1 if $\langle r, \tau \rangle$ equals b and 0 otherwise. Else, if $p_j(\tau) = \perp$, then $\mathcal{B}_2$ also flips a bit and returns the outcome of the bitflip.

If the obfuscations where chosen honestly with respect to the target points $x_1^*, \ldots, x_q^*$, then circuits $C_3[k, p_1, \ldots, p_q, y_1^*, \ldots, y_q^*]$ and $C_4[k]$ differ on input τ if and only if $\tau = x_i^*$ for some $i \in \{1, \ldots, q\}$. Hence, if the differing-inputs adversary Ext outputs τ, then $\tau = x_i^*$ for some $i \in \{1, \ldots, q\}$ and, thus, with probability $1/q$ value τ is equal to x_j^* and, hence, $\mathcal{B}_2$ will output 1. If, on the other hand, the obfuscations p_i are of random points u_i, then the circuits $C_3[key, p_1, \ldots, p_q, y_1^*, \ldots, y_q^*]$ and $C_4[k]$ differ on input τ if and only if $\tau = u_i$ for some $i \in \{1, \ldots, q\}$. Hence, if the differing-inputs adversary Ext outputs τ and $\tau = u_j$ then $\mathcal{B}$ will only output 1 with probability $\frac{1}{2}$ (since $\Pr[\langle u_j, r \rangle = b] = \frac{1}{2}$). The formal analysis is equivalent to the one for Claim 3.4 with an additional loss of factor $\frac{1}{q}$ for guessing the right index.

To finish the proof of Claim C.1, we need to argue that $\mathcal{B}_1$ implements a statistically unpredictable distribution. By assumption, the source S is strongly, statistically unpredictable (i.e., $\mathsf{S} \in \mathcal{S}^{\text{s-sup}}$) and hence leakage L hides the query points even in the presence of y_i^* statistically. Thus, to see that $\mathcal{B}_1$ defines an unpredictable distribution, we need to argue that x_i^* remain unpredictable if additionally given a single bit of x_j^* and an index j. But a single bit and index can be guessed with probability $\frac{1}{2q}$. Hence, $(\mathcal{B}_1, \mathcal{B}_2)$ breaks the security of the AIPO obfuscation, which concludes the proof of Claim C.1.