Modeling Random Oracles Under Unpredictable Queries

Pooya Farshim, Arno Mittelbach

Modeling Random Oracles under Unpredictable Queries

Pooya Farshim[1] and Arno Mittelbach[2]

[1] ENS, CNRS & INRIA, PSL Research University, Paris, France
[2] Darmstadt University of Technology, Germany
pooya.farshim@gmail.com arno.mittelbach@cased.de

Abstract. In recent work, Bellare, Hoang, and Keelveedhi (CRYPTO 2013) introduced a new abstraction called Universal Computational Extractors (UCEs), and showed how they can replace random oracles (ROs) across a wide range of cryptosystems. We formulate a new framework, called Interactive Computational Extractors (ICEs), that extends UCEs by viewing them as models of ROs under unpredictable (aka. high-entropy) queries. We overcome a number of limitations of UCEs in the new framework, and in particular prove the adaptive RKA and semi-adaptive KDM securities of a highly efficient symmetric encryption scheme using ICEs under key offsets.

We show both negative and positive feasibility results for ICEs. On the negative side, we demonstrate ICE attacks on the HMAC and NMAC constructions. On the positive side we show that: 1) ROs are indeed ICE secure, thereby confirming the structural soundness of our definition and enabling a finer layered approach to protocol design in the RO model; and 2) a modified version of Liskov's Zipper Hash is ICE secure with respect to an underlying fixed-input-length RO, for appropriately restricted classes of adversaries. This brings the first result closer to practice by moving away from variable-input-length ROs. Our security proofs employ techniques from indifferentiability in multi-stage settings.

Keywords. Random oracle, Unpredictability, UCE, RKA security, KDM security, Zipper Hash, Indifferentiability, Multi-stage security.

1 Introduction

1.1 Background

Since their formal introduction by Bellare and Rogaway [BR93], random oracles (ROs) have found many applications across a wide range of cryptographic protocols. However, due to an uninstantiability result of Canetti, Goldreich, and Halevi [CGH98], which shows that certain (artificial) protocols become insecure as soon as the random oracle is replaced by *any* concrete hash function, reliance on ROs has also become somewhat debatable.

Two lines of research have been directed at dealing with such uninstantiability results. One is to construct standard-model counterparts of cryptographic primitives designed in the RO model (ROM). This approach comes with the drawback that the resulting cryptosystems often tend to be complex and achieve a lower level of security and/or efficiency. A second, more modular, approach aims to formulate abstractions of the proof-centric properties of random oracles such as extractability, programability, or non-malleability [Can97,CD09,Nie02,CD08,BCFW09]. Assuming that a hash function meets the introduced model, one proceeds to show that it can safely replace the random oracle in a protocol. These formalizations, however, have only been successful to a limited extent, and the question of finding a flexible and general framework that could be applied across a broad range of security goals and protocols remained open until recently.

1.2 UCE security

Bellare, Hoang, and Keelveedhi (BHK) [BHK13a] revisit the above questions and present a powerful framework called *Universal Computational Extractors* (UCEs) that allow to securely instantiate random oracles in an

interesting and diverse set of applications. These include, among other things, security under key-dependent-message (KDM) attacks, security under related-key attacks (RKAs), simultaneous hard-core bits, point function obfuscators, garbling schemes, proofs of storage, deterministic encryption, and message-locked encryption, thereby going far beyond what was previously possible.

Behind UCEs lies a new way to model the indistinguishability of a keyed hash function from a random oracle. Indeed, there are two direct ways to (incorrectly) model the security of a hash function:

(1) Provide the adversary with the hash key and ask it to distinguish an oracle implementing the hash function from one implementing the random oracle. This approach immediately fails as this game can be trivially won with the knowledge of the hash key by computing a hash value and checking the answer against the oracle's answer for the same query.

(2) Adopt the above approach, but now hide the hash key. This leads to PRF security—for which feasibility results are known—but is not useful in the context of hashing as the hash key is typically publicly known.

BHK overcome the above shortcomings by *splitting* the attacker into two parts and *constraining* the communication between the two. The first UCE attacker does *not* get to see the hash key, but has oracle access to either the hash function under a random key or the random oracle according to a random bit. The second attacker, on the other hand, *does* get to see the hash key, but can no longer access the oracle, and it has to guess the bit; see Figure 1 (left). The two stages of the adversary can communicate only in restricted ways since arbitrary communication would lead to an attack similar to that given above for formulation (1).

More formally, for a keyed hash function H, UCE security is defined via a two-stage game consisting of algorithms S and D, called the *source* and the *distinguisher* respectively, as follows. In the first stage, the source is given access to an oracle HASH that depending on a random bit b implements either the random oracle or the concrete hash function H under a random hash key hk. The source terminates by outputting some leakage L, which is then communicated to the second-stage distinguisher D. In addition to leakage L, the distinguisher also gets the hash key hk as input. The distinguisher's task is to guess b, i.e., guess whether the source was talking to the random oracle or the hash function. The UCE advantage of the pair (S, D) is defined as usual to be the probability of correctly guessing the bit b scaled away from one-half. We refer the reader to the original work [BHK13b] for an excellent overview of this approach to modeling hash-function security.

To see that without further restrictions UCE security cannot be achieved, consider a source that leaks one of its oracle queries together with the corresponding oracle answer to the distinguisher. The distinguisher then simply recomputes the hash value on the queried point—the distinguisher knows the hash key—and compares it to the leaked value.

In their original work, BHK [BHK13a] define two restrictions on sources: *computational unpredictability* and *computational reset security*. In the computational unpredictability game, it is required that when the source is run with a *random oracle* its leakage does not computationally reveal any of its queries. This is formalized by requiring that the probability of any efficient predictor P in guessing a query of S when given L is negligible.

The class of computationally unpredictable sources is denoted by $\mathcal{S}^{\text{cup}}$, and the resulting UCE security UCE[$\mathcal{S}^{\text{cup}}$] (aka. UCE1) of a hash function is defined by requiring the advantage of any efficient pair (S, D) with an *unpredictable* $S \in \mathcal{S}^{\text{cup}}$ in the UCE game to be negligible. Reset security imposes a weaker restriction on the source class and leads to the stronger UCE2 notion.

UCE security has been the subject of many recent studies. Brzuska, Farshim, and Mittelbach (BFM) [BFM14] show that, under new cryptographic assumptions, these restrictions are insufficient for a feasible definition. More precisely, assuming the existence of indistinguishability obfuscators [BGI+01,GGH+13], BFM show that the UCE[$\mathcal{S}^{\text{cup}}$] security of *any* hash function can be broken in polynomial time. To overcome this attack, BFM [BFM14] (and subsequently BHK in an updated version of their paper [BHK13b]) propose a *statistical* notion of unpredictability whereby the predictor can even run in unbounded time. Following the attack, BHK also refine the UCE notions based on computational unpredictability and introduce the classes of *bounded*

Fig. 1. The interactions in the UCE game (left) and the ICE game (right).

parallel and *split* sources.[3] BFM show that security against bounded parallel source is also infeasible [BFM14], and recently attacks against split sources have also been shown [BST15].

On the positive side, Brzuska and Mittelbach [BM14b,BM15] show how to construct UCEs for the class of *strongly* unpredictable and statistically unpredictable sources for bounded number of queries. Bellare, Hoang, and Keelveedhi [BHK14] develop domain extenders for UCEs, and Bellare and Hoang [BH15] construct deterministic PKEs from UCEs for statistically unpredictable sources and lossy trapdoor functions. BFM [BFM14] have shown that the existence of obfuscation-based attacks against statistically unpredictable sources violates well-known impossibility results. A number of recent works have shown how to use UCEs as RO replacements in other protocols [MH14,BK15,DGG$^+$15].

Despite the above advances, and irrespective of the restrictions imposed on sources, the UCE framework is intrinsically limited in a number of aspects: it only allows the source to place HASH queries which are *independent* of the hash key; after leakage is communicated from the source to the distinguisher no further HASH oracle queries can be made, and hence hash queries are inherently *non-adaptive*; UCEs cannot model *unkeyed* hash functions nor hash functions with weak keys where the key does not come from the uniform distribution. Motivated by these shortcomings, and the ultimate goal of basing the security of highly efficient and practical protocols on well-defined and feasible properties of random oracles, we set out to formalize an enhanced framework for the study ROM protocols.

1.3 Interactive computational extractors

Given the development of UCEs, defining an extended model which meets the above-mentioned specifications is an intricate task. Indeed, well before the emergence of obfuscation-based attacks, BHK [BHK13b, page 9] warned that extending UCEs to an interactive setting is "a dangerous path to tread." As an example, assume that we introduce a bi-directional communication channel between the distinguisher and the source so that our adaptivity targets are met. This extension can be shown to fall prey to somewhat non-trivial attacks that utilize general-purpose multi-party computation (MPC) protocols. Suppose the source S holds a random input x whose hash is y, and D holds hk. The two parties then run an MPC protocol to compute the Boolean value $y = \mathsf{H}(hk, x)$. The distinguisher finally returns this value as its guess. This attack would meet any reasonable notion of computational unpredictability since the security of the MPC protocol would ensure that the parties learn no more than what can be deduced from their individual private inputs.[4] Allowing hash queries to depend on the hash key hk is also challenging since similarly to approach (1) above access to both hk and the hash oracle would trivialize the notion. For similar reasons, formulating a UCE-like model for unkeyed hash functions is also non-trivial. As we shall see, other forms of attacks also arise that should be ruled out for a feasible model.

THE ICE FRAMEWORK. Let us call an input (hk, x), consisting of the hash key hk and a domain point x, to a hash function a *full* input. One way to view UCEs is that they adopt the indistinguishability-based approach (1) above, but restrict hash queries so that full inputs remain hidden from the attacker(s). It is

[3] Such computational UCE notions are intrinsically needed for applications such as simultaneous had-core bits and deterministic PKEs.

[4] This can be viewed as an interactive analogue of BFM's attack [BFM14].

clear that such hidden queries are not meaningful in the presence of a single adversary—any adversary knows its own queries—and hence UCEs come with two adversaries. Unpredictability together with denial of oracle access to D ensures that the x components of full inputs remain hidden from D. On the other hand, the hk components of full inputs remain hidden from S as the source is denied access to hk (and no communication from D to S is allowed). As a result, full inputs (hk, x) remain hidden from *both* parties involved in a UCE attack.

This perspective allows us to build on UCEs and extend them as follows. In our new framework, which we call *Interactive Computational Extractors* (ICEs),[5] a general mechanism for the joint generation of full inputs is enabled and adversarial restrictions that formalize what it means for full hash inputs to have high entropy are imposed.

We let two distinguishers (D_1, D_2) to take part in an attack, and allow them to communicate via a bi-directional channel. Both distinguishers get access to a challenge hash oracle, which depending on a challenge bit implements either the real hash function or a (keyed) random oracle. To enable the two parties to make hidden queries, we introduce a shared write-only tape that both D_1 and D_2 can write onto. When a distinguisher queries the hash oracle, the (real or ideal) hash of the full contents of the tape is returned. In contrast to UCEs, D_1 or D_2 can generate a hash key and perhaps modify it throughout the attack. This attack scenario is symmetric for D_1 and D_2 and, without loss of generality, the game terminates by D_2 outputting with a bit. (Our formal definition, however, comes with a slightly more general return statement.) For a class $\mathcal{C}$ of distinguishers, we define ICE[$\mathcal{C}$] security by demanding that the probability of guessing the challenge bit for any $D = (D_1, D_2) \in \mathcal{C}$ is negligibly close to 1/2. See Figure 1 (right) for a summary of this interaction.

Entropic queries. Similarly to UCEs, the ICE notion cannot be achieved without constraining the way the two distinguishers communicate. The main restriction that we introduce is analogous to *statistical* unpredictability for UCEs: we demand the statistical unpredictability of *full* inputs to the hash function, including the hash key hk, from each distinguisher's point of view. We choose a statistical, rather than a computational, notion so that our definitions do not become subject to the interactive versions of the attacks highlighted in [BFM14].[6] More precisely, we require that when the hash oracle implements a keyed random oracle, no (possibly unbounded) predictor can guess a full input (hk, x) used to compute a hash value when it is provided with a distinguisher's *view* consisting of its inputs, random coins, and all incoming messages and oracle responses.

Since our framework allows oracle access to both parties, unlike UCEs the two distinguishers can implicitly communicate via *hash patterns* as follows. Suppose D_2 wants to leak a bit d to D_1. Algorithm D_2 starts by writing a random string onto the second half of the input and hands over the attack to D_1. Algorithm D_1 writes a random value to the first half of the input, calls Hash to receive a first hash value h_1, and hands over the attack back to D_2. Now algorithm D_2, according to the value of d, either modifies the contents of the second half of the input tape or leaves them unchanged. D_1 can recover d by obtaining a second hash value h_2 and checking if $(h_1 = h_2)$. The two distinguishers can also communicate via a *bit-fixing* attack: D_2 samples many (unpredictable) random values x conditioned on its hash value beginning with bit d, which D_1 can then recover via a hash query.

In our unpredictability definition the predictor gets to see all hash responses, and hence if there are any repetitions they will be seen by the predictor. Unpredictability will therefore ensure that such repetition patterns will not leak any of the queries. Sometimes, however, we need to explicitly disallow any repeat queries to enable a security proof to go through. In such a scenario, we can ensure that there is no leakage via hash patterns either. Repeat-freeness appears in other related settings such as related-key attacks or correlated-input hashing [BK03,GOR11].

[5] In UCEs, "universal" refers to the fact that extraction should work with respect to universal (i.e., all admissible) sources. Analogously, "interactive" in ICEs refers to the fact that extraction should work for sources that can interact.

[6] This is also motivated by impossibility results for statistically secure two-party protocols.

1.4 Applications

BHK [BHK13a] use UCEs to show that the encryption scheme of Black, Rogaway, and Shrimpton (BRS) [BRS03] is secure under related-key attacks (RKAs) and key-dependent-message (KDM) attacks as long as the related keys/key-dependent messages are derived non-adaptively at the onset and without access to the hash key or previous ciphertexts.[7]

As we shall see, ICE encompasses UCE as a special case, and the BRS scheme can also be instantiated under the above models using ICEs. We can however also obtain feasibility results that are outside the reach of UCEs. A practically relevant and desirable level of RKA security is that corresponding to key offsets (the so-called xor-RKA security [LRW02,BK09]). We show that ICEs are sufficient to prove the *full* xor-RKA security of the BRS scheme. Our formal result is more general and applies to the larger class of *split* functions that take the form $\phi(K_1\|K_2) = \phi_1(K_1)\|\phi_2(K_2)$. (Such functions have been used to build RKA-secure PRFs [BC10], and also appear in other related contexts [CG14,LL12].) In addition to achieving stronger security guarantees, ICEs allow instantiating the BRS scheme using unkeyed hash functions, which is arguably closer to the original formulation of BRS.[8]

We also strengthen the attainable KDM security guarantees for BRS by showing that adversaries can choose key-dependent messages adaptively based on the hash key and also semi-adaptively depending on previous ciphertexts. We prove that ICEs are adaptively correlated-input secure [GOR11] and that they relate well to other standard security properties of random oracles, such as pseudorandomness, randomness extraction, and one-way security (see Appendix B). We leave it as open questions to see if full RKA beyond xor offsets or full KDM security can be established using extractor-like notaions.

1.5 Instantiations

BHK show that random oracles fulfill their strongest proposed UCE notion, namely UCE security with respect to computationally unpredictable sources.[9] We prove that random oracles are also ICE secure. The significance of these results are twofold [BHK13a]: (1) there are no generic attacks on ICEs and the model is structurally sound; and (2) a *layered* approach to security analysis can be enabled, whereby one first proves the security of a scheme under an ICE assumption and then applies the RO model feasibility result. The latter is akin to security analyses carried out in the generic group model.

Practical hash functions, however, are not monolithic objects and often follow an iterative procedure to convert a fixed-input-length random oracle (FIL-RO) into a variable-input-length random oracle (VIL-RO). This, in turn, raises the question whether or not the above result can be brought closer to practice by demonstrating positive feasibility results for VIL-ICEs in the FIL-RO model. A seemingly immediate way to establish this result would be to start with a hash function that is known to be *indifferentiable* from a VIL RO (e.g., the HMAC or the NMAC construction), and then apply the RO feasibility result above to conclude. This argument, however, fails as the ICE game is *multi-staged* and indifferentiability does not necessarily guarantee composition in such settings [RSS11].

Motivated by the above observations, we show both positive and negative feasibility results for ICEs. On the negative side, we show that the indifferentiable HMAC and NMAC constructions are provably ICE *in*secure in the FIL-RO model. On the positive side, and building on Mittelbach's techniques [Mit14], we prove that a keyed version of Liskov's Zipper Hash [Lis07] *is* ICE secure (as a VIL hash function) under the assumption that the underlying compression function is a FIL-RO. Zipper Hash can be seen as a variant of the classical Merkle–Damgård [Dam90,Mer90] construction where the message blocks are processed twice in the forward and backward directions. Hence our results strengthen the VIL-RO feasibility result above, and also provide formal evidence for the (intuitive) added security guarantees that multi-pass hash functions seem

[7] Recall that in RKA security the adversary can see encryptions of messages under keys $\phi(K)$ for a random K and functions ϕ of its choice. In KDM security the adversary can see encryptions of $\phi(K)$, under a random key K, for ϕ's of its choice.

[8] BRS [BRS03] analyze their scheme in the unkeyed RO model, which translates to unkeyed instantiations in practice.

[9] Note that this does not contradict the BFM attack as ROs do not have succinct descriptions.

to offer over their single-pass counterparts. For instance, combined with our RKA and KDM results, we may conclude that Zipper Hash can be safely used within the BRS scheme with no adverse affects on its security.

The above analysis can be further strengthened in at least two directions. First, one can weaken the underlying assumption and assume that the compression function underlying Zipper Hash is only a FIL-ICE (rather than a FIL-RO). To this end, BHK [BHK14] give domain extenders for UCEs. Second, and motivated by the standard-model realizations of ICEs and UCEs, we ask if these primitives can be based on plausible hardness assumptions. Brzuska and Mittelbach [BM14a,BM15] have recently shown positive results for UCEs with respect to restricted classes of sources.

2 Notation

We denote the security parameter by $\lambda \in \mathbb{N}$, which is implicitly given to all algorithms (if not explicitly stated so) in the unary representation 1^λ. By $\{0,1\}^\ell$ we denote the set of all bit strings of length ℓ and $\{0,1\}^*$ is the set of all finite-length bit strings. For $x, y \in \{0,1\}^*$ we denote their concatenation by $x \| y$, the length of x by $|x|$, the ith bit of x by $x[i]$, and the substring of x formed using bits i to j by $x[i..j]$. We denote the empty string by ε. For X a finite set, $|X|$ denotes its cardinality, and $x \leftarrow\!\!\$\, X$ denotes the action of sampling x uniformly at random from X. If Q is a list and x a string then $\mathsf{Q} : x$ denotes the list obtained by appending x to Q. Similarly, If Q_1 and Q_2 are lists, then $\mathsf{Q}_1 : \mathsf{Q}_2$ denotes the concatenated list. Unless stated otherwise, algorithms are assumed to be randomized. We call an algorithm efficient or PPT if it runs in time polynomial in the security parameter. By $y \leftarrow \mathcal{A}(x; r)$ we denote that y was output by algorithm $\mathcal{A}$ on input x and randomness r. If $\mathcal{A}$ is randomized and no randomness is specified, then we assume that $\mathcal{A}$ is run with freshly sampled uniform random coins, and write $y \leftarrow\!\!\$\, \mathcal{A}(x)$. We use $\mathsf{Coins}[A]$ to denote the polynomially long string of random coins r used by a PPT machine A. We say a function $\mathrm{negl}(\lambda)$ is negligible if $\mathrm{negl}(\lambda) \in \lambda^{-\omega(1)}$.

Hash functions. In the line with [BHK13a], we consider the following (simplified) formalization of hash functions. A hash function consists of five PPT algorithms $\mathsf{H} := (\mathsf{H.Kg}, \mathsf{H.Ev}, \mathsf{H.kl}, \mathsf{H.il}, \mathsf{H.ol})$ as follows. The key-generation algorithm $\mathsf{H.Kg}$ gets the security parameter 1^λ as input and outputs a key $hk \in \{0,1\}^{\mathsf{H.kl}(\lambda)}$, where $\mathsf{H.kl}(\lambda)$ is the key-length function. Algorithm $\mathsf{H.il}(\lambda)$ outputs the length of admissible inputs, which could take the special value $*$ denoting the variable-length input space $\{0,1\}^*$. Algorithm $\mathsf{H.ol}(\lambda)$ outputs the length of admissible outputs, which we assumed to be a fixed polynomial function of the security parameter. The deterministic evaluation algorithm $\mathsf{H.Ev}$ takes as input the security parameter 1^λ, a key hk, a point $x \in \{0,1\}^{\mathsf{H.il}(\lambda)}$, and generates a hash value $\mathsf{H.Ev}(1^\lambda, hk, x) \in \{0,1\}^{\mathsf{H.ol}(\lambda)}$. To ease notation, we often suppress the security parameter and simply write $\mathsf{H.Ev}(hk, x)$.

3 The ICE Framework

In this section we precisely define the ICE framework. We refer the reader to the introduction for a high-level overview of the model.

The ICE game. Let $\mathsf{H} = (\mathsf{H.Kg}, \mathsf{H.Ev}, \mathsf{H.kl}, \mathsf{H.il}, \mathsf{H.ol})$ be a hash function and let $D = (D_1, D_2)$ be a pair of algorithms. We define the ICE advantage of D against H as

$$\mathbf{Adv}^{\mathrm{ice}}_{\mathsf{H},D}(\lambda) := 2 \cdot \Pr\left[\mathrm{ICE}^D_{\mathsf{H}}(\lambda)\right] - 1 \ ,$$

where game $\mathrm{ICE}^D_{\mathsf{H}}(\lambda)$ is shown in Figure 2. As mentioned in the introduction, we may assume, without loss of generality, that the game termites by D_2 outputting a bit. However, in order to preserve the symmetry of the definition (which will simplify our adversarial restrictions later on) and for added generality, we let the distinguishers jointly guess the challenge bit by computing $b_1 \oplus b_2$, where b_i is D_i's guess. The interaction terminates when both distinguishers return non-$\perp$ values for b_1 and b_2. For a class $\mathcal{C}$ of distinguishers, we define $\mathrm{ICE}[\mathcal{C}]$ security by requiring the advantage of any adversary $D \in \mathcal{C}$ to be negligible in the ICE game.

We require (D_1, D_2) not to leave any superfluous blank spaces on the joint tape. That is, a Write call must ensure that before the Hash oracle is called there do not exist indices $i < j$ such that $x[i] = \varepsilon \neq x[j]$ or

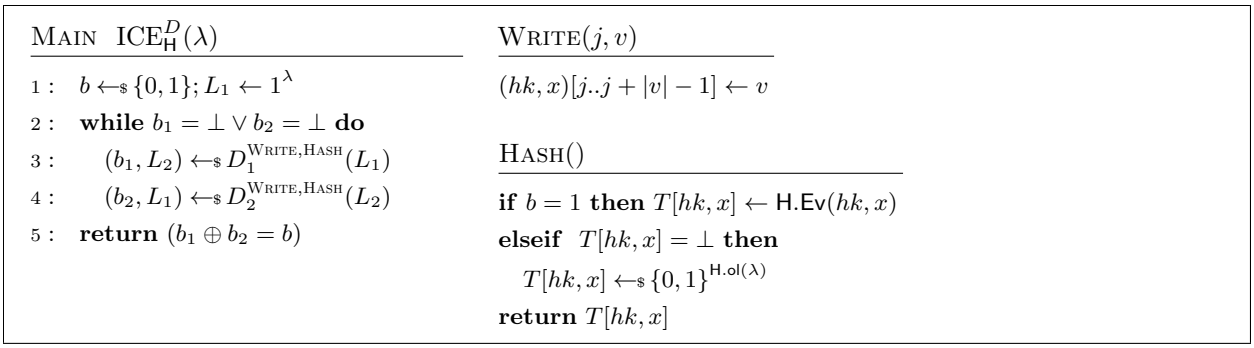

Fig. 2. The ICE game with respect to hash function H and distinguishers $D = (D_1, D_2)$. We have omitted the initialization of various variables for readability.

$hk[i] = \varepsilon \neq hk[j]$. We also demand that the full inputs (hk, x) are valid in the sense that prior to a HASH call $hk \in \{0,1\}^{\mathsf{H.kl}(\lambda)}$ and $x \in \{0,1\}^{\mathsf{H.il}(\lambda)}$. Although the distinguishers D_1 and D_2 are in general stateful algorithms, we omit the explicit handling of state values from the inputs and outputs of D_i.

RESTRICTIONS. As discussed in the introduction, the ICE model is not feasible unless additional restrictions on the distinguishers are imposed. We formulate our restrictions as joint properties of (D_1, D_2). Before presenting our main restrictions corresponding to high-entropy queries, we give a set of basic classes that will be useful in studying ICEs. As an example, for polynomials w, q, and r we define $\mathcal{C}_i^{w,q,r}$ to be the set of all (D_1, D_2) such that when (D_1, D_2) is run in the ICE game conditioned on $b = 0$ (i.e., with respect to the random oracle), the distinguisher D_i places at most $w(\lambda)$ queries to WRITE, at most $q(\lambda)$ queries to HASH, and terminates after at most $r(\lambda)$ invocations. We formalize a number of other notions below and omit the preamble "The set of all (D_1, D_2) such that when (D_1, D_2) is run in ICE with $b = 0$, we have with overwhelming probability that" from their definitions. Note that the classes below depend on $i \in \{1, 2\}$. For classes $\mathcal{C}_i^{\text{label}}$ we define $\mathcal{C}^{\text{label}} := \mathcal{C}_1^{\text{label}} \cap \mathcal{C}_2^{\text{label}}$. In the following table we present several restrictions that we will be using throughout this paper.

Class	Description
$\mathcal{C}_i^{w,q,r}$	D_i places at most $w(\lambda)$ queries to WRITE, at most $q(\lambda)$ queries to HASH, and terminates after at most $r(\lambda)$ invocations.
$\mathcal{C}_i^{\text{poly}}$	D_i makes polynomially many oracle queries.
$\mathcal{C}_i^{\text{ppt}}$	D_i runs in polynomial time on each invocation and terminates after a polynomial number of rounds.
$\mathcal{C}_i^{0}$	D_i sets $b_i := 0$ in all invocations.
$\mathcal{C}_i^{\varepsilon}$	D_i sets $L_{3-i} := \varepsilon$ in all invocations.
$\mathcal{C}_i^{\text{0-hk}}$	D_i never writes onto the hk part of the tape.
$\mathcal{C}_i^{\text{1-hk}}$	On its first invocation, D_i writes a random hk onto the hk-part of the tape. In subsequent invocations, D_i never writes onto the hk-part of the tape.
$\mathcal{C}_i^{\text{dist}}$	D_i makes distinct queries to HASH. That is, for lists Q_1 and Q_2 defined in Figure 3, the *combined* list $\mathsf{Q}_1 : \mathsf{Q}_2$ is repetition-free. Note that $\mathcal{C}_i^{\text{dist}} = \mathcal{C}_{3-i}^{\text{dist}} = \mathcal{C}^{\text{dist}}$.
$\mathcal{C}_i^{\text{sup}}$	The probability that any (possibly unbounded) predictor P can guess a full query of D_i is negligible. We call this the class of statistically unpredictable D_i. See Figure 3 for the formal definition. Class $\mathcal{C}_i^{\text{cup}}$ is the computational analogue, where P is restricted to be ppt.

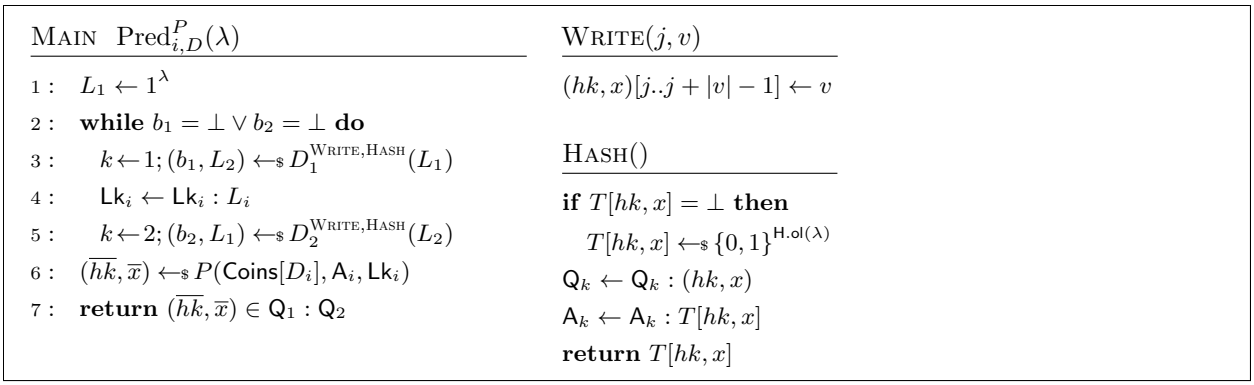

Fig. 3. The unpredictability game.

AN EXAMPLE: UCE WITHIN ICE. We describe how UCEs can be captured within the ICE framework. Since ICE is more expressive a framework, we need to (drastically) restrict the distinguishers. In modeling UCEs, we identify the UCE distinguisher with D_1 and the UCE source with D_2. All parties typically run in polynomial time and hence we restrict to $\mathcal{C}^{\text{ppt}} := \mathcal{C}_1^{\text{ppt}} \cap \mathcal{C}_2^{\text{ppt}}$. In UCEs, the source queries HASH on an unknown hash key. The distinguisher, on the other hand, gets to see the hash key. Thus, we let D_1 (which represents the distinguisher) write a random hk to the joint input and then hand the attack to D_2 on the first invocation, i.e., $D \in \mathcal{C}_1^{\text{1-hk}}$. We further restrict to $\mathcal{C}_1^{\varepsilon}$, as a UCE distinguisher does not leak. Since the UCE game only has a single round, we also restrict to $\mathcal{C}_1^{1,0,2}$ (one round is used to write the hk). Finally, the source does not take part in decision making and cannot modify the hash key: UCEs are modeled by ICE$[\mathcal{C}^{\text{uce}}]$ where

$$\mathcal{C}^{\text{uce}} := \mathcal{C}^{\text{ppt}} \cap \mathcal{C}_1^{\text{1-hk}} \cap \mathcal{C}_1^{\varepsilon} \cap \mathcal{C}_1^{1,0,2} \cap \mathcal{C}_2^{0} \cap \mathcal{C}_2^{\text{0-hk}} \ .$$

Note that the above models UCEs without any additional restrictions on the source classes. Such requirements can be added on top by appropriately restricting $\mathcal{C}^{\text{uce}}$.

UNPREDICTABILITY. We now formally define what we mean by a D that has unpredictable (aka. high-entropy) queries. We focus on a statistical notion of unpredictability [BFM14,BST15].[10] We say $D = (D_1, D_2)$ is statistically unpredictable for the distinguisher i, and write $D \in \mathcal{C}_i^{\text{sup}}$, if the advantage of any unbounded predictor P defined by

$$\mathbf{Adv}^{\text{pred}}_{i,D,P}(\lambda) := \Pr\left[\text{Pred}^P_{i,D}(\lambda)\right] \ ,$$

is negligible, where game $\text{Pred}^P_{i,D}(\lambda)$ is shown in Figure 3.

Note that the predicator only gets to see the hash responses for distinguisher D_i—these are within D_i's view—and has to guess a query made by *either* distinguisher in the concatenated list $\mathsf{Q}_1 : \mathsf{Q}_2$. It is easy to check that UCE security with respect to statistically unpredictable sources is equivalent to ICE$[\mathcal{C}^{\text{uce}} \cap \mathcal{C}^{\text{sup}}]$ security.

REMARK. Since predictor P receives the full view of a distinguisher D_i, it can perfectly simulate a run of D_i in the ICE game with respect to a *random* implementation of the hash oracle, without any need to see the view of the partner distinguisher D_{3-i}. We will rely on this observation in our proofs.

4 Example Applications

In this section we demonstrate two example use cases of ICEs. Further applications are given in Appendix B and summarized in Table 2 below. These applications serve to demonstrate that many properties of random oracles that are useful in analyses of ROM cryptosystems can be modeled in a unified way within the ICE framework.

[10] We emphasize that computational notions are still valuable as combined with our feasibility results, they would enable easier and more modular security proofs in the RO model.

Goal/Model	Class Used/Achieved
Split RKA	$\mathcal{C}^* \cap \mathcal{C}^{\text{sup}} \cap \mathcal{C}_2^0$
Split KDM	$\mathcal{C}^* \cap \mathcal{C}^{\text{sup}} \cap \mathcal{C}_1^0$
Split/claw-free CIH	$\mathcal{C}^* \cap \mathcal{C}^{\text{sup}} \cap \mathcal{C}_2^0$
Extractor	$\mathcal{C}^* \cap \mathcal{C}^{\text{sup}} \cap \mathcal{C}_1^0 \cap \mathcal{C}^{\varepsilon} \cap \mathcal{C}^{1,1,2}$
Weak PRF	$\mathcal{C}^* \cap \mathcal{C}^{\text{sup}} \cap \mathcal{C}_1^0 \cap \mathcal{C}^{\varepsilon} \cap \mathcal{C}^{\text{poly},\text{poly},1}$
poly-regular OWF	$\mathcal{C}^* \cap \mathcal{C}^{\text{sup}} \cap \mathcal{C}_1^0 \cap \mathcal{C}^{1,1,1}$
VIL-ROM	$\mathcal{C}^{\text{ppt}} \cap \mathcal{C}^{\text{cup}}$ and $\mathcal{C}^{\text{poly}} \cap \mathcal{C}^{\text{sup}}$; both contain $\mathcal{C}^* \cap \mathcal{C}^{\text{sup}}$
FIL-ROM	$\mathcal{C}^* \cap \mathcal{C}^{\text{cup}}$, which contains $\mathcal{C}^* \cap \mathcal{C}^{\text{sup}}$

Table 2. Distinguisher classes used (above) and shown feasibility for (below). Here $\mathcal{C}^* := \mathcal{C}^{\text{ppt}} \cap \mathcal{C}^{\text{dist}} \cap \mathcal{C}_1^{\text{1-hk}} \cap \mathcal{C}_2^{\text{0-hk}} \cap \mathcal{C}_2^{\varepsilon}$.

4.1 Split RKA security

We show that the symmetric encryption scheme proposed by Black, Rogaway, and Shrimpton (BRS) [BRS03] is secure against related-key attacks (RKAs) when instantiated with an ICE-secure hash function. The encryption algorithm of the BRS scheme is implemented via $\mathsf{Enc}^{\mathsf{H}}(K, M; R) := (R, M \oplus \mathsf{H}(K\|R))$, for a hash function H, randomness R and key K. Recall that in an RKA, an adversary can obtain encryptions of messages of its choice under correlated keys (e.g., under K and $K \oplus 1$). See Appendix A for the formal definition.

Split related-key derivation (RKD) functions ϕ decompose into two sub-RKD functions ϕ_1 and ϕ_2 that are applied in parallel to two (fixed) sub-strings of the key: $\phi(K_1\|K_2) = \phi_1(K_1)\|\phi_2(K_2)$.[11] Split functions capture many RKA cases of interest including the case of xoring constants into keys. Without the minimal assumption that ϕ's have unpredictable outputs (i.e., the guessing probability of the outputs of $\phi(K)$ over randomly chosen K is negligible) RKA security is not achievable [BK03]. In our proof, we will require a slightly stronger condition that the sub-RKD functions $\phi_1(K_1)$ and $\phi_2(K_2)$ are *individually* unpredictable. (See Appendix A for the formal definition.) Note that offsetting keys via xor enjoys this property as xor induces a permutation over the two halves of the key.

BHK [BHK13a], by interpreting encryption randomness as hash keys, show that BRS is selectively RKA secure using a multi-key extension of UCE[$\mathcal{S}^{\text{cup}}$]. In contrast, the adversary in our model retains its capability to adaptively query RKD functions of its choice depending both on the hash key and the ciphertexts that it has previously seen. For this result, although ICE[$\mathcal{C}^{\text{ppt}} \cap \mathcal{C}^{\text{sup}}$] is sufficient, the assumption can be fine-tuned to ICE[$\mathcal{C}$] where

$$\mathcal{C} := \mathcal{C}^{\text{ppt}} \cap \mathcal{C}^{\text{sup}} \cap \mathcal{C}^{\text{dist}} \cap \mathcal{C}_1^{\text{1-hk}} \cap \mathcal{C}_2^{\text{0-hk}} \cap \mathcal{C}_2^0 \cap \mathcal{C}_2^{\varepsilon} \ .$$

We defer the formal proof to Appendix A and give a detailed outline here.

The ICE adversary. Given an RKA adversary A, we construct an ICE adversary (D_1, D_2), where D_1 handles the left components of A's RKA queries and D_2 handles the right components as follows.

$D_1(L_1)$: On initial invocation, generate a hash key hk, a random K_1, and a random bit b. Store these values and write hk onto the hk-part of the tape. Run $A(hk)$ to get an RKA query $((\phi_1, \phi_2), M_0, M_1)$. Output $(b_1, L_2) := (\perp, \phi_2)$. Proceed as follows in subsequent invocations. Generate and store a random R and write $\phi_1(K_1)$ onto the 1st segment (out of three segments) of the x-part of the tape and R onto its 3rd segment. Query Hash to get H. Recover R and resume A on $(R, H \oplus M_b)$ to get a new RKA query $((\phi_1, \phi_2), M_0, M_1)$, or a bit b'. If A outputs a bit b', return $(b_1, L_2) := (b = b', \varepsilon)$ and terminate. Else output $(b_1, L_2) := (\perp, \phi_2)$.

[11] For simplicity we assume that these are just the left and right halves of the key. Our proof will however also apply to any two substrings of super-logarithmic lengths.

$D_2(L_2)$: When initially invoked, generate a random K_2 and store it. In all invocations (including the first), recover ϕ_2 from L_2. If $\phi_2 = \varepsilon$, return $(b_2, L_1) := (0, \varepsilon)$ and terminate. Else write $\phi_2(K_2)$ onto the 2nd segment of the x-part of the tape. Output $(b_2, L_1) := (0, \varepsilon)$.

UNPREDICTABILITY. We show that $D \in \mathcal{C}$ for class $\mathcal{C}$ as defined above. To this end, we only prove membership in $\mathcal{C}^{\text{sup}} \cap \mathcal{C}^{\text{dist}}$ as other cases follow via syntactic checks. This follows from the following two observations: (1) The HASH queries are distinct with overwhelming probability since before each query a fresh random value R is written onto the joint tape. (2) The functions ϕ_1 and ϕ_2 are run on independently chosen substrings of the key. Since they are assumed to be individually statistically unpredictable, D_1 observing independently generated random strings corresponding to hash values never gets to know the contents of the tape written by D_2, and vice versa, D_2 never gets to know what is written on to the tape by D_1.

4.2 KDM security

When the random oracle in the BRS scheme is instantiated with an ICE-secure hash function, we are able to show that the BRS scheme resists a partially adaptive form of KDM security for split key-dependent-message derivation (KDMD) functions ϕ. As for RKD functions, such KDMD functions consist of sub-KDMD functions ϕ_1 and ϕ_2 of the form $\phi(K_1 \| K_2) := \phi_1(K_1) \| \phi_2(K_2)$. The adaptivity level that we can tolerate is as follows. In an initial phase of the attack, the adversary can fully adaptively query split KDMD functions that do not depend on K_2. That is, for these functions $\phi_2(K_2)$ is constant and independent of K_2 and its value can be predicted. In a second phase of the attack, the adversary can query split KDMD functions of its choice as long as either $\phi_1(K_1)$ is constant or $\phi_1(K_1)$ was used in the first phase. (We emphasize that these functions are not required to be unpredictable.) This model is strong enough to imply IND-CPA security (without any restrictions), a case that could not be treated using UCEs.

THE ICE ADVERSARY. Let A be a KDM adversary against the BRS scheme in the model above. Our ICE$[\mathcal{C}^{\text{ppt}} \cap \mathcal{C}^{\text{sup}} \cap \mathcal{C}^{\text{dist}}]$ adversary corresponding to A is as follows, where for simplicity we have assumed the lengths of keys, randomness and messages are all ℓ. (The ICE class can be further restricted as is shown in Table 2.) In this reduction, D_1 faithfully runs the first stage of the attack, while D_2 runs its second stage. To answer KDM queries, D_2 relies on the "homomorphic" property that $H \oplus (x_1 \| x_2) = H \oplus (x_1 \| 0^{|x_2|}) \oplus (0^{|x_1|} \| x_2)$.

$D_1(L_1)$: When initially invoked, generate a random hk, K_1 and b and store them. Write hk to the hk-part and K_1 to the 1st (out of three) segments of the x-part of the tape. (The segments are of lengths $\ell/2$, $\ell/2$ and ℓ corresponding to K_1, K_2 and R respectively.) Output $(\perp, \varepsilon)$. On the second invocation, run $A(hk)$ and answer its KDM queries $((\phi_1^0, \phi_2^0), (\phi_1^1, \phi_2^1))$ as follows. Write a fresh random value R onto the 3rd segment of the x-part of the tape. Call HASH to get H, and resume A on $(R, H \oplus (\phi_1(K_1) \| M_2^*))$, where $M_2^* := \phi_2(0^{\ell/2})$ is the right K_2-independent part of the message. Continue this process until A decides to proceed to its second stage. Let st_A denote A's state. Generate sufficiently many copies $(R_1, C_1'), \ldots, (R_q, C_q')$ of each of the KDM queries made in the first phase. Let List_1 denote the corresponding list of queried ϕ_1^b. Return $(0, (b, st_A, (R_1, C_1'), \ldots, (R_q, C_q'), \mathsf{List}_1))$ and terminate.

$D_2(L_2)$: When initially invoked, generate a random K_2, store it, and write it to the 2nd segment of the x-part of the tape. Hand the attack back to D_1, by outputting $(\perp, \varepsilon)$. On the second invocation, parse L_2 appropriately as above. Resume A on st_A and answer its KDM queries $((\phi_1^0, \phi_2^0), (\phi_1^1, \phi_2^1))$ as follows. If $\phi_1^b \in \mathsf{List}_1$ pick a fresh ciphertext (R, C') corresponding to ϕ_1^b and *complete* the ciphertext preparation by setting $C \leftarrow C \oplus (0^{\ell/2} \| \phi_2^b(K_2))$. Otherwise generate a random R, write it onto the 3rd segment of the x-part of the tape, query HASH to get H, and set $C \leftarrow H \oplus (\phi_1^b(0^{\ell/2}) \| \phi_2^b(K_2))$. Resume A on $(R, C; st_A)$ and continue in this manner until A outputs a bit b'. Return $(b = b', \varepsilon)$ and terminate.

UNPREDICTABILITY. D's queries are distinct with overwhelming probability as fresh randomness R is written on the tape before each query. Throughout the attack, and when the hash oracle implements a random function, K_2 remains hidden from D_1 as D_1 only sees distinct random values as hash responses. Key K_1 also remains hidden from D_2 as the (incomplete) ciphertext components received from D_1 are random strings. Hence $D \in \mathcal{C}^{\text{ppt}} \cap \mathcal{C}^{\text{sup}} \cap \mathcal{C}^{\text{dist}}$.

5 Feasibility

In this section we start by showing that random oracles are ICE secure with respect to interesting distinguisher classes (in particular, with respect to the restrictions needed for the presented applications). We then consider the ICE security of practical hash constructions built from fix-input-length (FIL) ROs. In particular, we look at a keyed variant of Liskov's Zipper Hash [Lis07] and show that it achieves ICE security in the FIL-RO model. Interestingly, we show that both HMAC and NMAC constructions [BCK96], which were recently shown to be UCE secure in FIL-ROM [Mit14], fail to be ICE secure. This result yields a natural counterexample to the composability of HMAC in multi-stage settings, similarly to that given by Ristenpart, Shacham, and Shrimpton in [RSS11]. Furthermore, it provides a separation between ICE and UCE. Our results also demonstrate that Zipper Hash can provide a higher level of security compared to HMAC when used in multi-stage settings.

5.1 ICEs from random oracles

BHK [BHK13b] show that UCE-secure hash functions can be provably constructed in the RO model. The philosophical justifications of this result are that there are *no structural weaknesses* in the definitional framework, and more importantly, a *layered* approach to protocol design in the RO model can be enabled [BHK13b]. We show that ICEs also enjoy RO feasibility.

Let $\mathsf{H.kl}(\cdot)$ and $\mathsf{H.ol}(\cdot)$ be two arbitrary functions as in the syntax of a hash function. Let $\mathcal{R}$ be a family of variable-input-length (VIL) ROs (i.e., with domain $\{0,1\}^*$) and range $\{0,1\}^{\mathsf{H.ol}(\lambda)}$. We construct the required hash function $\mathsf{H}^{\mathcal{R}}$ by defining the key-generation algorithm $\mathsf{H.Kg}(1^\lambda)$ to return a random $hk \leftarrow\!\!\$\ \{0,1\}^{\mathsf{H.kl}(\lambda)}$ and the evaluation algorithm $\mathsf{H.Ev}^{\mathcal{R}}(hk, x)$ to return $\mathcal{R}(hk\|x)$. Our first feasibility result is as follows.

Theorem 1 (ICE **feasibility in ROM).** *The VIL hash function $\mathsf{H}^{\mathcal{R}}$ constructed above is* ICE$[\mathcal{C}]$ *secure in the VIL-RO model for $\mathcal{R}$ for the following (incomparable) classes of adversaries:*

$$\mathcal{C} := \mathcal{C}^{\mathrm{poly}} \cap \mathcal{C}^{\mathrm{sup}} \quad \textit{and} \quad \mathcal{C} := \mathcal{C}^{\mathrm{ppt}} \cap \mathcal{C}^{\mathrm{cup}} \ .$$

The proof of this theorem is similar to the proof of [BHK13b, Theorem 6.1] for UCEs, and we give the details in Appendix C. Intuitively, we rely on unpredictability of queries to simulate the random oracles used in the construction and implicit in the ICE game independently. Interestingly, distinctness of queries will not be needed in this proof and we do not restrict the classes to $\mathcal{C}^{\mathrm{dist}}$. We note that the above classes include all those needed for the applications, as listed in Table 2. We also note that this theorem generalizes the feasibility of UCEs for unpredictable sources in ROM [BHK13b] as it can be easily verified that

$$\mathcal{C}^{\mathrm{uce}} \cap \mathcal{C}_2^{\mathrm{cup}} \subseteq \mathcal{C}^{\mathrm{ppt}} \cap \mathcal{C}^{\mathrm{cup}} \quad \text{and} \quad \mathcal{C}^{\mathrm{uce}} \cap \mathcal{C}_2^{\mathrm{sup}} \subseteq \mathcal{C}^{\mathrm{ppt}} \cap \mathcal{C}^{\mathrm{sup}} \ .$$

5.2 VIL-ICEs from ideal compression

Practical variable-input-length (VIL) hash functions are not monolithic objects. They often follow iterative modes of chaining that convert a fix-input-length (FIL) compression function to one that accepts variable-length inputs. This design principle has been successfully validated via the indifferentiability framework of Maurer, Renner, and Holenstein [MRH04,CDMP05], whereby an indifferentiable hash-function construction is shown to securely compose when used in place of a random oracle. As pointed out in [RSS11], the indifferentiability framework only guarantees composition in single-stage environments. The ICE and UCE games, however, are inherently multi-staged and lie outside the reach of (plain) indifferentiability. Mittelbach [Mit14] develops new techniques to extend the reach of (plain) indifferentiability to certain classes of multi-stage games. In particular, he shows that the HMAC and NMAC constructions are UCE secure. Interestingly, we show that these results do not carry over to the ICE model: HMAC and NMAC provably fail to be ICE secure. On the other hand, we build on Mittelbach's techniques to prove that a variant of Zipper Hash [Lis07] is provably ICE secure.

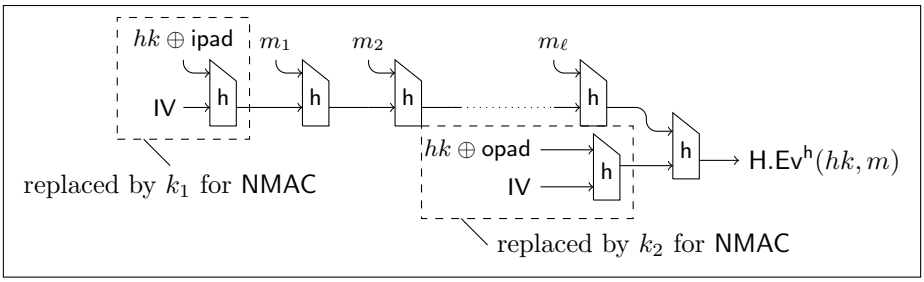

Fig. 4. The HMAC construction. If the dashed boxes are exchanged for independent keys k_1 and k_2, we obtain NMAC. Here we are ignoring padding.

Attacks on HMAC and NMAC. The HMAC and NMAC constructions are shown in Figure 4. If we denote the iterated compression function used in HMAC by h, then it is easily seen that key hk is only used on the "outer" h-evaluations. Consider an ICE distinguisher D_1 which holds hk, computes the values

$$y_1 := \mathsf{h}(hk \oplus \mathsf{ipad}, \mathsf{IV}) \qquad \text{and} \qquad y_2 := \mathsf{h}(hk \oplus \mathsf{opad}, \mathsf{IV})$$

and sends them to distinguisher D_2. Given (y_1, y_2), distinguisher D_2 can compute the HMAC values for any $x \in \{0,1\}^*$ under hk. Thus, in order to win the ICE game, D_2 simply chooses a random x and writes it on the input tape, and calls Hash to receive a value y. It then locally recomputes $\mathsf{H.Ev}^{\mathsf{h}}(hk, x)$ using the compression function h and values (y_1, y_2). If the results match, it outputs 1, and else it outputs 0. It is easily seen that this adversary wins ICE with overwhelming probability. Furthermore, given (y_1, y_2), the hash key hk remains statistically hidden from D_2 (as the number of h queries is bounded by a polynomial). Value x, being random, also remains statistically hidden from D_1. Formally, this attack breaks ICE[$\mathcal{C}$] for

$$\mathcal{C} := \mathcal{C}^{\text{ppt}} \cap \mathcal{C}^{\text{sup}} \cap \mathcal{C}^{1,1,1} \cap \mathcal{C}_1^{\text{1-hk}} \cap \mathcal{C}_1^{0} \cap \mathcal{C}_2^{\text{0-hk}} \cap \mathcal{C}_2^{\varepsilon} \ .$$

Zipper Hash. The above attack raises the question if any iterative hash function can be ICE[$\mathcal{C}$] secure for a meaningful class of distinguishers $\mathcal{C}$. We show that a hybrid construction of a keyed version of Liskov's Zipper Hash construction [Lis07] and chopped Merkle–Damgård (chop-MD) of Coron et al. [CDMP05] is ICE secure. Zipper Hash can be regarded as a basic Merkle–Damgård scheme where the message is processed twice, the second time in reversed block order. chop-MD refers to the construction where a hash value consists only of the first half of the output bits of the final compression function. Our hybrid construction results from adding the chop step to Zipper Hash. Furthermore, we consider a keyed variant of Zipper Hash by prepending the hash key to the message. We assume that key length matches block length, which means that the *first and last* evaluations of the compression function operate on the hash key. We denote this keyed variant of Zipper Hash by chop-KZIP. Figure 5 shows a schematic diagram of the construction, and Appendix D.1 gives further details.

Theorem 2 (Zipper Hash's ICE security). *The VIL hash function* $\mathsf{chop\text{-}KZIP}^{\mathsf{h}}$ *constructed above is* ICE[$\mathcal{C}$] *secure in the FIL-RO model for* $\mathsf{h} : \{0,1\}^{\mu} \times \{0,1\}^{n} \longrightarrow \{0,1\}^{n}$ *for the class*

$$\mathcal{C} := \mathcal{C}^{\text{poly}} \cap \mathcal{C}^{\text{sup}} \cap \mathcal{C}^{\text{dist}} \cap \mathcal{C}_1^{\text{1-hk}} \cap \mathcal{C}_1^{0} \cap \mathcal{C}_2^{\text{0-hk}} \cap \mathcal{C}_2^{\varepsilon} \ .$$

An analogous result holds for polynomial-time distinguishers that are only computationally unpredictable.

In Appendix D.2 we give the proof, where we also present a self-contained introduction to the unsplittability technique [Mit14]. Appendix D.3 contains the full security analysis.

Note that class $\mathcal{C}$ above contains that class used to attack HMAC and hence $\mathsf{chop\text{-}KZIP}^{\mathsf{h}}$ provably achieves a higher level of security in multi-stage games. We note that the reach of the above feasibility result includes all applications scenarios listed in Table 2. In particular, $\mathsf{chop\text{-}KZIP}^{\mathsf{h}}$ can security replace the random oracle in these applications. For this also note that we can easily drop $\mathcal{C}_1^0$ by requesting that in the last round D_1

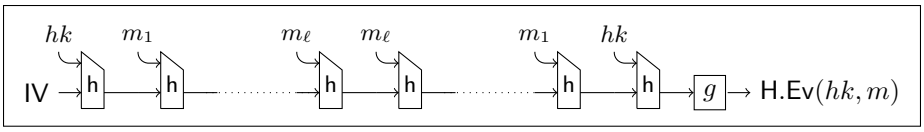

Fig. 5. The Zipper Hash construction merged with chop-MD [CDMP05] and keyed with hk. The final node g corresponds to the projection to the first half of the output of h.

outputs a guess for b which D_2 echoes. With the other restrictions present this change is without loss of generality.

This result cannot be strengthened for the (large) adversarial classes that were used in Theorem 1. To see this, consider two distinguishers that engage in a *distributed* computation of $\mathsf{chop\text{-}KZIP}^{\mathsf{h}}$ hash values as follows. Distinguisher D_1 knows hk and m_1 and D_2 knows m_2, where message $m := m_1 \| m_2$ is being hashed. Distinguisher D_1 computes an intermediate hash digest using (hk, m_1) and forwards it to D_2. Distinguisher D_2 now computes another iteration of the hash using m_2 and forwards the result to D_1. Distinguisher D_1 can now complete the hash computation using its knowledge of (hk, m_1) and the intermediate hash digest that it receives.

A straightforward generalization of this attack also rules out multi-pass variants of $\mathsf{chop\text{-}KZIP}^{\mathsf{h}}$ (where messages are processed multiple times in the forward and backward directions), including those whose number of passes is not fixed a priori and can depend on the number of message blocks. This is due to the fact that the number of rounds in an ICE attack is not fixed. This, in turn, raises the question if $\mathrm{ICE}[\mathcal{C}^{\mathrm{poly}} \cap \mathcal{C}^{\mathrm{sup}}]$ is feasible in the FIL-RO model. We conclude the paper with a candidate construction that we conjecture to reach this level of security.

Mix Hash. Let $\mathsf{h} : \{0,1\}^n \times \{0,1\}^n \longrightarrow \{0,1\}^n$ be a compression function. Let $m := m_1 \| \cdots \| m_\ell \in (\{0,1\}^n)^\ell$ be a message with ℓ blocks of length n each. Let $\mathsf{Mix}^{\mathsf{h}}(m)$ denote the transformation that maps m to $M := \|_i \|_j M_{i,j}$ where $M_{i,j} := \mathsf{h}(m_i, m_j)$ for $1 \le i < j \le \ell$. (Therefore M has $\ell(\ell-1)/2$ blocks.) Now let $hk \in \{0,1\}^n$ be a hash key and define

$$\mathsf{MixHash}^{\mathsf{h}}(hk, m) := \mathsf{HMAC}^{\mathsf{h}}(0^n, \mathsf{Mix}^{\mathsf{h}}(hk \| m)) \ .$$

Note that $\mathsf{MixHash}^{\mathsf{h}}$ places $\Theta(\ell^2)$ calls to its compression function h.[12] The design rationale behind $\mathsf{MixHash}^{\mathsf{h}}$ is as follows. All intermediate digests values $M_{i,j}$ are needed in order to successfully compute a hash value. These values, however, consist of all pairs (m_i, m_j) compressed through h. Since h is a monolithic object, $M_{i,j}$ cannot be computed in a distributed way, a strategy that was used in all previous attacks. In other words, one of the distinguishers has to know (hk, m) in full and hence will violate unpredictability. To see this, suppose D_1 does not know m_j in full and D_2 does not know m_i in full for some $i < j$. Then there is no way for these parties to learn $M_{i,j} := \mathsf{h}(m_i, m_j)$ without one of them explicitly quarrying h on (m_i, m_j). This however means that both m_i and m_j are known to the quarrying party, which leads to a contradiction. We leave a formal analysis of $\mathsf{MixHash}^{\mathsf{h}}$ in the FIL-RO model for h as future work.

Acknowledgments

The authors would like to thank Christina Brzuska for taking part in the early stages of this work. Pooya Farshim was supported in part by grant ANR-14-CE28-0003 (Project EnBid).

References

[BC10] Mihir Bellare and David Cash. Pseudorandom functions and permutations provably secure against related-key attacks. In Tal Rabin, editor, *CRYPTO 2010*, volume 6223 of *LNCS*, pages 666–684. Springer, Heidelberg, August 2010.

[12] Indeed, MixHash is a (highly) offline function: for $\alpha \in [0,1]$, it requires space roughly $n\ell\sqrt{1-\alpha}$ bits after a fraction α of the $n\ell(\ell+1)/2$ bits are processed.

[BCFW09] Alexandra Boldyreva, David Cash, Marc Fischlin, and Bogdan Warinschi. Foundations of non-malleable hash and one-way functions. In Mitsuru Matsui, editor, *ASIACRYPT 2009*, volume 5912 of *LNCS*, pages 524–541. Springer, Heidelberg, December 2009.

[BCK96] Mihir Bellare, Ran Canetti, and Hugo Krawczyk. Keying hash functions for message authentication. In Neal Koblitz, editor, *CRYPTO'96*, volume 1109 of *LNCS*, pages 1–15. Springer, Heidelberg, August 1996.

[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, Part I*, volume 8616 of *LNCS*, pages 188–205. Springer, Heidelberg, August 2014.

[BG81] C. H. Bennett and J. Gill. Relative to a random oracle A, $P^A \neq NP^A \neq coNP^A$ with probability 1. *SIAM Journal on Computing*, 10(1):96–113, 1981.

[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, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 1–18. Springer, Heidelberg, August 2001.

[BH15] Mihir Bellare and Viet Tung Hoang. Resisting randomness subversion: Fast deterministic and hedged public-key encryption in the standard model. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part II*, volume 9057 of *LNCS*, pages 627–656. Springer, Heidelberg, April 2015.

[BHK13a] Mihir Bellare, Viet Tung Hoang, and Sriram Keelveedhi. Instantiating random oracles via UCEs. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part II*, volume 8043 of *LNCS*, pages 398–415. Springer, Heidelberg, August 2013.

[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.

[BHK14] Mihir Bellare, Viet Tung Hoang, and Sriram Keelveedhi. Cryptography from compression functions: The UCE bridge to the ROM. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part I*, volume 8616 of *LNCS*, pages 169–187. Springer, Heidelberg, August 2014.

[BK03] Mihir Bellare and Tadayoshi Kohno. A theoretical treatment of related-key attacks: RKA-PRPs, RKA-PRFs, and applications. In Eli Biham, editor, *EUROCRYPT 2003*, volume 2656 of *LNCS*, pages 491–506. Springer, Heidelberg, May 2003.

[BK09] Alex Biryukov and Dmitry Khovratovich. Related-key cryptanalysis of the full AES-192 and AES-256. In Mitsuru Matsui, editor, *ASIACRYPT 2009*, volume 5912 of *LNCS*, pages 1–18. Springer, Heidelberg, December 2009.

[BK15] Mihir Bellare and Sriram Keelveedhi. Interactive message-locked encryption and secure deduplication. In Jonathan Katz, editor, *PKC 2015*, volume 9020 of *LNCS*, pages 516–538. Springer, Heidelberg, March / April 2015.

[BM14a] Christina Brzuska and Arno Mittelbach. Indistinguishability obfuscation versus multi-bit point obfuscation with auxiliary input. In Palash Sarkar and Tetsu Iwata, editors, *ASIACRYPT 2014, Part II*, volume 8874 of *LNCS*, pages 142–161. Springer, Heidelberg, December 2014.

[BM14b] Christina Brzuska and Arno Mittelbach. Using indistinguishability obfuscation via UCEs. In Palash Sarkar and Tetsu Iwata, editors, *ASIACRYPT 2014, Part II*, volume 8874 of *LNCS*, pages 122–141. Springer, Heidelberg, December 2014.

[BM15] Christina Brzuska and Arno Mittelbach. Universal computational extractors and the superfluous padding assumption for indistinguishability obfuscation. Cryptology ePrint Archive, Report 2015/581, 2015. http://eprint.iacr.org/.

[BR93] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In V. Ashby, editor, *ACM CCS 93*, pages 62–73. ACM Press, November 1993.

[BRS03] John Black, Phillip Rogaway, and Thomas Shrimpton. Encryption-scheme security in the presence of key-dependent messages. In Kaisa Nyberg and Howard M. Heys, editors, *SAC 2002*, volume 2595 of *LNCS*, pages 62–75. Springer, Heidelberg, August 2003.

[BST15] Mihir Bellare, Igors Stepanovs, and Stefano Tessaro. Contention in cryptoland: Obfuscation, leakage and UCE. Cryptology ePrint Archive, Report 2015/487, 2015. http://eprint.iacr.org/2015/487.

[Can97] Ran Canetti. Towards realizing random oracles: Hash functions that hide all partial information. In Burton S. Kaliski Jr., editor, *CRYPTO'97*, volume 1294 of *LNCS*, pages 455–469. Springer, Heidelberg, August 1997.

[CD08] Ran Canetti and Ronny Ramzi Dakdouk. Extractable perfectly one-way functions. In Luca Aceto, Ivan Damgård, Leslie Ann Goldberg, Magnús M. Halldórsson, Anna Ingólfsdóttir, and Igor Walukiewicz, editors, *ICALP 2008, Part II*, volume 5126 of *LNCS*, pages 449–460. Springer, Heidelberg, July 2008.

[CD09] Ran Canetti and Ronny Ramzi Dakdouk. Towards a theory of extractable functions. In Omer Reingold, editor, *TCC 2009*, volume 5444 of *LNCS*, pages 595–613. Springer, Heidelberg, March 2009.

[CDMP05] Jean-Sébastien Coron, Yevgeniy Dodis, Cécile Malinaud, and Prashant Puniya. Merkle-Damgård revisited: How to construct a hash function. In Victor Shoup, editor, *CRYPTO 2005*, volume 3621 of *LNCS*, pages 430–448. Springer, Heidelberg, August 2005.

[CG14] Mahdi Cheraghchi and Venkatesan Guruswami. Non-malleable coding against bit-wise and split-state tampering. In Yehuda Lindell, editor, *TCC 2014*, volume 8349 of *LNCS*, pages 440–464. Springer, Heidelberg, February 2014.

[CGH98] Ran Canetti, Oded Goldreich, and Shai Halevi. The random oracle methodology, revisited (preliminary version). In *30th ACM STOC*, pages 209–218. ACM Press, May 1998.

[Dam90] Ivan Damgård. A design principle for hash functions. In Gilles Brassard, editor, *CRYPTO'89*, volume 435 of *LNCS*, pages 416–427. Springer, Heidelberg, August 1990.

[DGG+15] Yevgeniy Dodis, Chaya Ganesh, Alexander Golovnev, Ari Juels, and Thomas Ristenpart. A formal treatment of backdoored pseudorandom generators. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part I*, volume 9056 of *LNCS*, pages 101–126. Springer, Heidelberg, April 2015.

[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 FOCS*, pages 40–49. IEEE Computer Society Press, October 2013.

[GOR11] Vipul Goyal, Adam O'Neill, and Vanishree Rao. Correlated-input secure hash functions. In Yuval Ishai, editor, *TCC 2011*, volume 6597 of *LNCS*, pages 182–200. Springer, Heidelberg, March 2011.

[Lis07] Moses Liskov. Constructing an ideal hash function from weak ideal compression functions. In Eli Biham and Amr M. Youssef, editors, *SAC 2006*, volume 4356 of *LNCS*, pages 358–375. Springer, Heidelberg, August 2007.

[LL12] Feng-Hao Liu and Anna Lysyanskaya. Tamper and leakage resilience in the split-state model. In Reihaneh Safavi-Naini and Ran Canetti, editors, *CRYPTO 2012*, volume 7417 of *LNCS*, pages 517–532. Springer, Heidelberg, August 2012.

[LRW02] Moses Liskov, Ronald L. Rivest, and David Wagner. Tweakable block ciphers. In Moti Yung, editor, *CRYPTO 2002*, volume 2442 of *LNCS*, pages 31–46. Springer, Heidelberg, August 2002.

[Mer90] Ralph C. Merkle. One way hash functions and DES. In Gilles Brassard, editor, *CRYPTO'89*, volume 435 of *LNCS*, pages 428–446. Springer, Heidelberg, August 1990.

[MH14] Takahiro Matsuda and Goichiro Hanaoka. Chosen ciphertext security via UCE. In Hugo Krawczyk, editor, *PKC 2014*, volume 8383 of *LNCS*, pages 56–76. Springer, Heidelberg, March 2014.

[Mit13] Arno Mittelbach. Salvaging indifferentiability in a multi-stage setting. Cryptology ePrint Archive, Report 2013/286, 2013. http://eprint.iacr.org/2013/286.

[Mit14] Arno Mittelbach. Salvaging indifferentiability in a multi-stage setting. In Phong Q. Nguyen and Elisabeth Oswald, editors, *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 603–621. Springer, Heidelberg, May 2014.

[MRH04] Ueli M. Maurer, Renato Renner, and Clemens Holenstein. Indifferentiability, impossibility results on reductions, and applications to the random oracle methodology. In Moni Naor, editor, *TCC 2004*, volume 2951 of *LNCS*, pages 21–39. Springer, Heidelberg, February 2004.

[Nie02] Jesper Buus Nielsen. Separating random oracle proofs from complexity theoretic proofs: The non-committing encryption case. In Moti Yung, editor, *CRYPTO 2002*, volume 2442 of *LNCS*, pages 111–126. Springer, Heidelberg, August 2002.

[RSS11] Thomas Ristenpart, Hovav Shacham, and Thomas Shrimpton. Careful with composition: Limitations of the indifferentiability framework. In Kenneth G. Paterson, editor, *EUROCRYPT 2011*, volume 6632 of *LNCS*, pages 487–506. Springer, Heidelberg, May 2011.

A Split RKA Security

Symmetric encryption. A symmetric key encryption scheme consists of six PPT algorithms $\mathsf{SE} := (\mathsf{SE.kl}, \mathsf{SE.il}, \mathsf{SE.cl}, \mathsf{SE.Kg}, \mathsf{SE.Enc}, \mathsf{SE.Dec})$ as follows. Algorithms $\mathsf{SE.kl}, \mathsf{SE.il}, \mathsf{SE.cl}$ are deterministic and on input 1^λ specify the key, message and ciphertexts lengths. The key generation algorithm $\mathsf{SE.Kg}(1^\lambda)$ generates keys K of length $\mathsf{SE.kl}(1^\lambda)$. The encryption algorithm $\mathsf{SE.Enc}$ takes as input a key K and a message M of length $\mathsf{SE.il}(1^\lambda)$ and returns a ciphertext C of length $\mathsf{SE.cl}(1^\lambda)$. The decryption algorithm $\mathsf{SE.Dec}$ is deterministic and takes as input a key K and a ciphertext C and returns a message M. Correctness requires that for all K that are output by $\mathsf{SE.Kg}(1^\lambda)$ and for all messages M of length $\mathsf{SE.il}(1^\lambda)$, and all ciphertexts $C \leftarrow_\$ \mathsf{SE.Enc}(K, M)$ we have that $M = \mathsf{SE.Dec}(K, C)$.

MAIN $\mathrm{RKA}^{A}_{\Phi,\mathsf{SE}}(\lambda)$	MAIN $\mathrm{SPUP}^{P}_{\Phi}(\lambda)$
$b \leftarrow\!\!\$\ \{0,1\}$ $K \leftarrow\!\!\$\ \{0,1\}^{\mathsf{SE.kl}(\lambda)}$ $b' \leftarrow\!\!\$\ A^{\mathrm{LR}}(1^\lambda)$ **return** $(b' = b)$	$(\overline{K}_1, \overline{K}_2, \phi_1, \phi_2) \leftarrow\!\!\$\ P(1^\lambda)$ $(K_1, K_2) \leftarrow\!\!\$\ \{0,1\}^{\mathsf{SE.kl}(\lambda)/2} \times \{0,1\}^{\mathsf{SE.kl}(\lambda)/2}$ **return** $(\overline{K}_1 = \phi_1(K_1) \vee \overline{K}_2 = \phi_2(K_2))$
$\underline{\mathrm{LR}(\phi, M_0, M_1)}$ $K' \leftarrow \phi(K)$ $C \leftarrow\!\!\$\ \mathsf{SE.Enc}(K', M_b)$ **return** C	

Fig. 6. The RKA and split unpredictability games. A RKA adversary is legitimate if it calls RKEnc only with functions $\phi = (\phi_1, \phi_2)$ in $\Phi = \Phi_1 \times \Phi_2$. A SPUP adversary in is legitimate if it outputs a $(\phi_1, \phi_2) \in \Phi_1 \times \Phi_2$.

Split RKD functions. A related-key derivation (RKD) function is a circuit ϕ which maps keys to keys in some key space. We denote by a family of RKD function sets parameterized by λ by Φ. (We will suppress the explicit dependency on λ to ease notation.) An RKD family Φ is *split* if there are RKD families Φ_1 and Φ_2 with half the key size such that for any $\phi \in \Phi$ there is a $(\phi_1, \phi_2) \in \Phi_1 \times \Phi_2$ such that for all $K = K_1 \| K_2$, where K_1 and K_2 are of length half the key size (which for simplicity we assume to be even), we have

$$\phi(K_1 \| K_2) = \phi_1(K_1) \| \phi_2(K_2) \ .$$

We assume that given a split RKD function ϕ its split representation consisting of (ϕ_1, ϕ_2) can be efficiently computed. Note it might not necessarily be the case that $\Phi = \Phi_1 \times \Phi_2$.

RKA security. Let SE be a symmetric encryption scheme, and let Φ be a family of RKD functions with key space identical to that of SE. We define the Φ-RKA advantage of an adversary A against SE by

$$\mathbf{Adv}^{\mathrm{rka}}_{\mathsf{SE},\Phi,A}(\lambda) := 2 \cdot \Pr\left[\mathrm{RKA}^{A}_{\mathsf{SE},\Phi}(\lambda)\right] - 1 \ ,$$

where game $\mathrm{RKA}^{A}_{\mathsf{SE},\Phi}(\lambda)$ is shown in Figure 6. We define Φ-RKA security by requiring this advantage to be negligible for all PPT adversaries A.

Split unpredictability. A family of split RKD functions Φ is called split-unpredictable if the advantage of any (possibly unbounded) predictor P against Φ defined by

$$\mathbf{Adv}^{\mathrm{spup}}_{\Phi,P}(\lambda) := \Pr\left[\mathrm{SPUP}^{P}_{\Phi}(\lambda)\right] \ ,$$

is negligible, where game $\mathrm{SPUP}^{P}_{\Phi}(\lambda)$ is shown in Figure 6. Standard (thats is, non-split) unpredictability is defined similarly.

Theorem 3 (Split RKA security of the BRS scheme). *Let* SE *denote the BRS symmetric encryption scheme, where the hash oracle is instantiated with* H*. Let* Φ *be a split RKD function family. Then for any* Φ-RKA *adversary* A *against* SE *there exists an* ICE *adversary* $D = (D_1, D_2) \in \mathcal{C}^{\mathrm{ppt}} \cap \mathcal{C}^{\mathrm{sup}} \cap \mathcal{C}^{\mathrm{dist}}$ *such that*

$$\mathbf{Adv}^{\mathrm{rka}}_{\mathsf{SE},\Phi,A}(\lambda) \leq 2 \cdot \mathbf{Adv}^{\mathrm{ice}}_{\mathsf{H},D}(\lambda) + \frac{Q(\lambda)^2}{2^{\mathsf{SE.rl}(\lambda)}} \ .$$

Furthermore, for any predictor P *against adversary* D *constructed above, there are predictors* P'_i *for* $i \in \{1,2\}$ *such that*

$$\mathbf{Adv}^{\mathrm{pred}}_{i,D,P}(\lambda) \leq Q(\lambda) \cdot \mathbf{Adv}^{\mathrm{spup}}_{\Phi,P'_i}(\lambda) + \frac{Q(\lambda)^2}{2^{\mathsf{SE.rl}(\lambda)+1}} \ ,$$

where $Q(\lambda)$ *denotes the number of* LR *queries of* A*.*

$\underline{D_1^{\text{WRITE,HASH}}(L_1; st_1)}$

if $(st_1 = \varepsilon)$ **then**
  $hk \leftarrow\!\!\$\ \{0,1\}^{\mathsf{H.kl}(\lambda)}$
  $K_1 \leftarrow\!\!\$\ \{0,1\}^{\mathsf{SE.kl}(\lambda)/2}$
  $b \leftarrow\!\!\$\ \{0,1\}$
  $(\phi_1, \phi_2, M_0, M_1; st_A) \leftarrow\!\!\$\ A(hk)$
  $st_1 \leftarrow (hk, K_1, b, \phi_1, M_0, M_1, st_A)$
  return $(\bot, \phi_2; st_1)$
if $(st_1 \neq \varepsilon)$ **then**
  $(hk, K_1, b, \phi_1, M_0, M_1, st_A) \leftarrow st_1$
  $R \leftarrow\!\!\$\ \{0,1\}^{\mathsf{SE.rl}(\lambda)}$
  $\text{WRITE}(\mathsf{H.kl}(\lambda) + 1, \phi_1(K_1))$
  $\text{WRITE}(\mathsf{H.kl}(\lambda) + \mathsf{SE.kl}(\lambda) + 1, R)$
  $H \leftarrow\!\!\$\ \text{HASH}()$
  $(aux; st_A) \leftarrow A(R, H \oplus M_b; st_A)$
  if $(aux \in \{0,1\})$ **then**
    return $(aux = b, \varepsilon; \bot)$
  $(\phi_1, \phi_2, M_0, M_1) \leftarrow aux$
  $st_1 \leftarrow (hk, K_1, b, \phi_1, M_0, M_1, st_A)$
  return $(\bot, \phi_2; st_1)$

$\underline{D_2^{\text{WRITE,HASH}}(L_2; st_2)}$

if $(st_2 = \varepsilon)$ **then**
  $K_2 \leftarrow\!\!\$\ \{0,1\}^{\mathsf{SE.kl}(\lambda)/2}$
  $st_2 \leftarrow K_2$
if $(L_2 = \varepsilon)$ **then**
  return $(0, \varepsilon; \bot)$
$K_2 \leftarrow st_2$
$\phi_2 \leftarrow L_2$
$\text{WRITE}(\mathsf{H.kl}(\lambda) + \mathsf{SE.rl}(\lambda)/2 + 1, \phi_2(K_2))$
return $(0, \varepsilon; st_2)$

Fig. 7. Distinguishers D_1 and D_2 corresponding to an RKA adversary A. State value $st_i = \bot$ indicates termination.

Proof. For an intuition of the proof we refer the reader to Section 4. Given A we construct an ICE adversary (D_1, D_2) as shown in Figure 7. We denote the internal state of D_i by st_i, which is initialized to ε. We assume, without loss of generality, that A places at least one RKA query.

Let d denote the hidden bit in the ICE game. When $d = 1$ adversary A is run in an environment that is identical to the RKA game for BRS with respect to hash function H and challenge bit b. Since the output of the ICE game is $d' := (b' = b)$, where b' denotes A's output, we have

$$\Pr\left[\text{ICE}_{\mathsf{H}}^{D}(\lambda) \;\middle|\; d = 1\right] = \Pr\left[\text{RKA}_{\mathsf{SE},\Phi}^{A}(\lambda)\right] \ .$$

On the other hand, when $d = 0$, subject to the condition that all R-values generated by D_1 are distinct, adversary A receives uniform random strings. Hence b remains information theoretically hidden from A:

$$\Pr\left[\text{ICE}_{\mathsf{H}}^{D}(\lambda) \;\middle|\; d = 0\right] \leq \frac{1}{2} + \frac{Q(\lambda)^2}{2^{\mathsf{SE.rl}(\lambda)+1}} \ .$$

Subtracting we get

$$2 \cdot \mathbf{Adv}_{\mathsf{H},D}^{\text{ice}}(\lambda) \geq \mathbf{Adv}_{\mathsf{SE},\Phi,A}^{\text{rka}}(\lambda) - \frac{Q(\lambda)^2}{2^{\mathsf{SE.rl}(\lambda)}} \ .$$

This proves the first part of the theorem.

We next prove that $D \in \mathcal{C}_i^{\text{sup}}$ for $i = 1, 2$. (Membership in $\mathcal{C}^{\text{ppt}} \cap \mathcal{C}^{\text{dist}}$ is easy to check.) Let P be a predictor against D in game $\text{Pred}_{i,D}^{P}(\lambda)$. We construct a predictor P_i' in game Φ-SPUP. Intuitively, algorithm P_i' will use D to construct a list of candidate sub-RKD functions whose output it will then predict using P.[13]

We consider a slight modification of $\text{Pred}_{i,D}^{P}(\lambda)$, where HASH queries are answered by returning uniform and independent strings in $\{0,1\}^{\mathsf{SE.rl}(\lambda)} \times \{0,1\}^{\mathsf{H.ol}(\lambda)}$ in a forgetful manner. Modulo the distinctness of the sampled R values, this game is identical to the real unpredictability game. Hence the two games are within statistical distance $Q(\lambda)^2/2^{\mathsf{SE.rl}(\lambda)+1}$.

We introduce a final modification. If $i = 1$, when computing the list of queries Q_1 and Q_2 the game uses $\phi_2(\tilde{K}_2)$ for an independent key $\tilde{K}_2$. Analogously, when $i = 2$, when computing Q_1 and Q_2 the game uses

[13] Note that a direct simulation of (D, P) by P_i' would not work at this point since P_i' will guess the output of a sub-RKD with respect to a *known* key K.

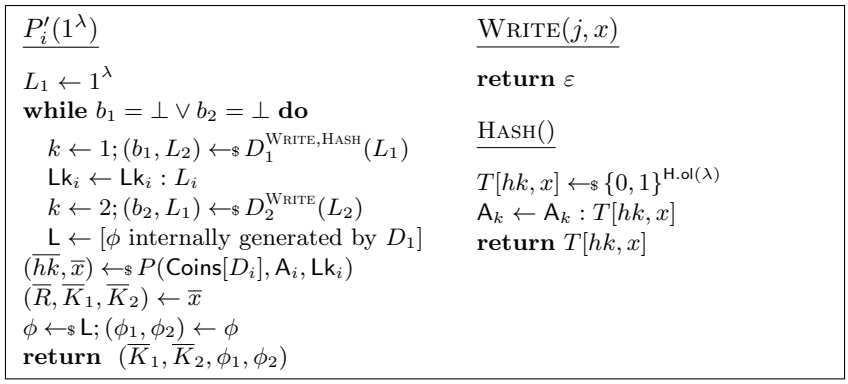

Fig. 8. Φ-SPUP predictor P_i' corresponding to ICE adversary $D = (D_1, D_2)$ and predictor P. Here L denotes the list of RKD functions (ϕ_1, ϕ_2) that are internally generated by D as it runs A.

$\phi_1(\tilde{K}_1)$ for an independent key $\tilde{K}_1$. In each case, these modifications do no affect P's success probability. To see this, note that for each i, the distributions of the inputs of P even in the presence of the list of queries $\mathsf{Q}_1 : \mathsf{Q}_2$ in the two cases are identical (as P does not get to see K_{3-i}).

We now construct the predictors for $i = 1, 2$. Algorithm P_i' runs (D, P) by simulating its HASH queries forgetfully as above and ignoring its WRITE queries. It forms a list of all RKD functions that are internally generated. It generates random $\mathsf{Coins}[D_i]$, forms A_2 via simulated hash values, and Lk_i via the list of quarried RKD functions. When P returns $(\overline{hk}, \overline{x})$, algorithm P_i' returns $(\overline{K}_1, \overline{K}_2, \phi_1, \phi_2)$ for a randomly sampled split RKD function (ϕ_1, ϕ_2) from the set of all queried RKD functions, and $(\overline{K}_1, \overline{K}_2)$ recovered from $\overline{x}$. See Figure 8 for the details.

For each i, when P is successful, it guesses the contents of the tape formed via an RKD function with respect to an independent key $\tilde{K}_{3-i}$ With probability $1/Q(\lambda)$ algorithm P_i' picks the RKD query that resulted in the predicted contents of the tape. It follows that for $i = 1, 2$ we have

$$\Pr\left[\text{SPUP}_\Phi^{P_i'}(\lambda)\right] \geq \frac{1}{Q(\lambda)} \cdot \left(\Pr\left[\text{Pred}_{i,D}^{P}(\lambda)\right] - \frac{Q(\lambda)^2}{2^{\mathsf{SE.rl}(\lambda)+1}}\right) .$$

The second part of the theorem follows.

□

B Other Applications

In this section we give an overview of how the ICE framework can be applied in a number of other cryptographic settings. Proofs of our results below can be formalized following the approach of Appendix A.

B.1 Correlated-input hashing

In correlated-input hashing, an adversary is able to obtain real or ideal hash values of points $\phi(x)$, where functions ϕ are chosen by the adversary from a pre-specified set, and x is a randomly chosen value which is kept hidden from the adversary. The task of the adversary is to guess if the real or ideal hash is being used. An adversary similar to the one given for the RKA security of the BRS scheme can be used to show that a hash function that meets the ICE definition of security also achieves security under correlated inputs as introduced by Goyal, O'Neill, and Rao [GOR11]. As before we require that the corresponding correlated-input derivation (CID) functions to split into two parallel components. In this setting, however, not only do we need the sub-CID functions to be individually output unpredictable, but also require that the (full) outputs of any two CID functions ϕ and ϕ' collide with negligible probability over a random choice of (common) input x. This property is also known as *claw-freeness*. The ICE distinguishers interact as follows.

$D_1(L_1)$: When initially invoked, generate a random hk and write it. Also generate a random x_1 and store it. Run $A(hk)$ to get a first query (ϕ_1, ϕ_2). Write $\phi_1(x_1)$ and output $(0, \phi_2)$. In subsequent invocations, query HASH to get H and resume A on H to get a new (ϕ_1, ϕ_2) or a bit b'. In the latter case return (b', ε) and terminate. In the first case write $\phi_1(x_1)$ and output $(0, \phi_2)$.

$D_2(L_2)$: When initially invoked, generate a random x_2 and store it. In all invocations, if $L_2 = \varepsilon$ return $(0, \varepsilon)$ and terminate. Else recover ϕ_2 from L_2 and write $\phi_2(x_2)$. Output $(0, \varepsilon)$.

The claw-freeness property implies that only with negligible probability the HASH oracle will be called on repeated inputs. Moreover, the split unpredictability condition ensures that the contents of the first segment of the input tape are hidden from D_2 and the contents of the second segment of the input tape are hidden from D_1 when these are run with respect to a random oracle. The argument is similar to the RKA setting, but claw-freeness will be used to forgetfully simulate the HASH oracle. Hence $D \in \mathcal{C}^{\text{sup}} \cap \mathcal{C}^{\text{dist}}$.

B.2 Randomness extraction

In a secure randomness extractor no adversary A should be able to distinguish a uniformly chosen random string from the output of the extractor when applied to a high-entropy input generated by a source S. Given a pair (S, A), we construct an ICE adversary as follows.

$D_1(L_1)$: When initially invoked, generate a random hk. Write hk to the tape. Output $(0, \varepsilon)$. On the second invocation, query HASH to get H. Run $A(hk, H)$ to get a bit b'. Return (b', ε) and terminate.

$D_2(L_2)$: Run $S(1^\lambda)$ to get x. Write x to the tape. Return $(0, \varepsilon)$ and terminate.

Note that there is a single HASH query and hence there are no repeat queries. Observe that the hash key hk is statistically hidden from D_2. Since S is assumed to generate (statistically) high-entropy values, the single query to HASH also remains statistically hidden from D_1 when HASH implements a random oracle. We conclude that $D \in \mathcal{C}^{\text{sup}} \cap \mathcal{C}^{\text{dist}}$.

B.3 Weak pseudorandomness

A weak pseudorandom function F needs to be indistinguishable from a random function by adversaries that are given input/output samples $(x, F_K(x))$ for a randomly chosen key K and randomly chosen points x. Any ICE-secure hash function is also a weak pseudorandom function.

$D_1(L_1)$: Generate a random hk and write it. Return $(0, \varepsilon)$ and terminate.

$D_2(L_2)$: Run the weak-PRF adversary $A(1^\lambda)$. When a sample is requested, pick a random value x and write it, call HASH to get H, and resume $A(H)$. When A outputs a bit b', return (b', ε) and terminate.

Statistical unpredictability follows as hk remains hidden from D_2 with respect to a random HASH, and queries x being random remain hidden from D_1. Note also that with overwhelming probability there are no repeat queries to HASH. Hence $D \in \mathcal{C}^{\text{sup}} \cap \mathcal{C}^{\text{dist}}$.

B.4 One-way hashing

We show that any ICE-secure hash function with a polynomial upper bound on its regularity (i.e., a polynomial number of pre-images for any given image) is one way.

$D_1(L_1)$: Choose a random hk. Choose a random x_1. Write hk and x_1. Return $(0, hk)$ and terminate.

$D_2(L_2)$: Choose a random x_2 and write it. Query HASH to get a value H. Run $A(hk, H)$ to get a candidate pre-image $(\tilde{x}_1 \| \tilde{x}_2)$. Return $(\tilde{x}_2 = x_2, \varepsilon)$ and terminate.

The above reduction relies on the fact that due to polynomial regularity, a successful A, when run with respect to the hash function, will recover *the* pre-image used in challenge generation with high probability. On the other hand, when A is run with respect to the random oracle, it will recover x_2 with only a negligible probability.

There is a single query to HASH, the input x_1 chosen by D_1 remains hidden from D_2 with respect to a random oracle, and the input x_2 chosen by D_2 remains hidden from D_1. The adversary is in $\mathcal{C}^{\text{sup}} \cap \mathcal{C}^{\text{dist}}$.

C Proof of Theorem 1: ICE Feasibility in ROM

Let $\mathsf{H.kl}(\cdot)$ and $\mathsf{H.ol}(\cdot)$ be two arbitrary functions as in the syntax of a hash function. Let $\mathcal{R}$ be a family of variable-input-length (VIL) ROs (i.e., with domain $\{0,1\}^*$) and range $\{0,1\}^{\mathsf{H.ol}(\lambda)}$. We construct the required hash function $\mathsf{H}^{\mathcal{R}}$ by defining $\mathsf{H.Kg}(1^\lambda)$ to return a random $hk \leftarrow\!\!\$\, \{0,1\}^{\mathsf{H.kl}(\lambda)}$ and the evaluation algorithm $\mathsf{H.Ev}^{\mathcal{R}}(hk, x)$ to return $\mathcal{R}(hk\|x)$.

Theorem 1 (restated from page 11). *The VIL hash function $\mathsf{H}^{\mathcal{R}}$ constructed above is* ICE$[\mathcal{C}]$ *secure in the VIL-RO model for $\mathcal{R}$ for the following (incomparable) classes of adversaries:*

$$\mathcal{C} := \mathcal{C}^{\text{poly}} \cap \mathcal{C}^{\text{sup}} \quad \textit{and} \quad \mathcal{C} := \mathcal{C}^{\text{ppt}} \cap \mathcal{C}^{\text{cup}} \ .$$

Proof. We prove the theorem for the first class corresponding to poly-query and statistically unpredictable distinguishers. The proofs for computational distinguishers that are computationally unpredictable are analogous. The only difference is that the predictors that we construct run the distinguishers, and hence their assumed computational complexities should match.

Let

$$D = (D_1, D_2) \in \mathcal{C}^{\text{poly}} \cap \mathcal{C}^{\text{sup}} \ .$$

Consider games G_1–G_3 in Figure 9, where game G_1 does not include the boxed statements. Let d denote the challenge bit in the ICE security game. Game G_1 is identical to the ICE game with $d = 1$, where Hash implements the construction. Game G_3 is identical to the ICE game with $d = 0$, where Hash is a lazily sampled random function independent of the random oracle used in the construction. In all games, Ro_1 and Ro_2 are the interfaces for the random oracle given to D_1 and D_2 respectively. We have that

$$\begin{aligned}\Pr\left[G_1^D(\lambda)\right] &= \Pr\left[\text{ICE}_{\mathsf{H}}^D(\lambda) \;\middle|\; d = 1\right] \\ \Pr\left[G_3^D(\lambda)\right] &= 1 - \Pr\left[\text{ICE}_{\mathsf{H}}^D(\lambda) \;\middle|\; d = 0\right] \ .\end{aligned}$$

Thus

$$\mathbf{Adv}^{\text{ice}}_{\mathsf{H},D}(\lambda) = \Pr\left[G_1^D(\lambda)\right] - \Pr\left[G_3^D(\lambda)\right] \ .$$

Game G_2 (shown in Figure 9 on top with boxed statements included) simply adds sets for bookkeeping and two bad flags. Therefore,

$$\Pr\left[G_1^D(\lambda)\right] = \Pr\left[G_2^D(\lambda)\right] \ .$$

Game G_3 (shown in Figure 9 bottom) exchanges the random oracle for adversaries D_1 and D_2 by an independent oracle; that is, the answers are no longer consistent with those of the Hash oracle (note the different tables T and H used). As long as a full query to Hash (with the hash key prepended) never collides with one to Ro_1 or Ro_2, this modification does not change the game.[14] This collision event is captured by a flag bad. Since G_2 and G_3 are identical until bad,

$$\Pr\left[G_2^D(\lambda)\right] - \Pr\left[G_3^D(\lambda)\right] \le \Pr\left[G_3^D(\lambda) \text{ sets } \mathsf{bad}\right] \ .$$

Flag bad can be set in two ways:

Event E_1: An Ro_1 query of D_1 matches a query of either D_1 or D_2 to Hash.
Event E_2: An Ro_2 query of D_2 matches a query of either D_1 or D_2 to Hash.

[14] We emphasize that we do not simulate the queries of D_1 and D_2 to Hash independently. Doing so would arguably lead to an easier simulation of the oracles, but comes at the cost of introducing the extra restriction that the distinguishers belong to $\mathcal{C}^{\text{dist}}$. Interestingly, similar restrictions are necessarily in related contexts (e.g., [BK03]). The reason we can avoid $\mathcal{C}^{\text{dist}}$ is that the predictor explicitly sees these hash values. When not provided with these values, the distinctness of queries would ensure that these hash values are random and independent, and hence of little use to a predictor.

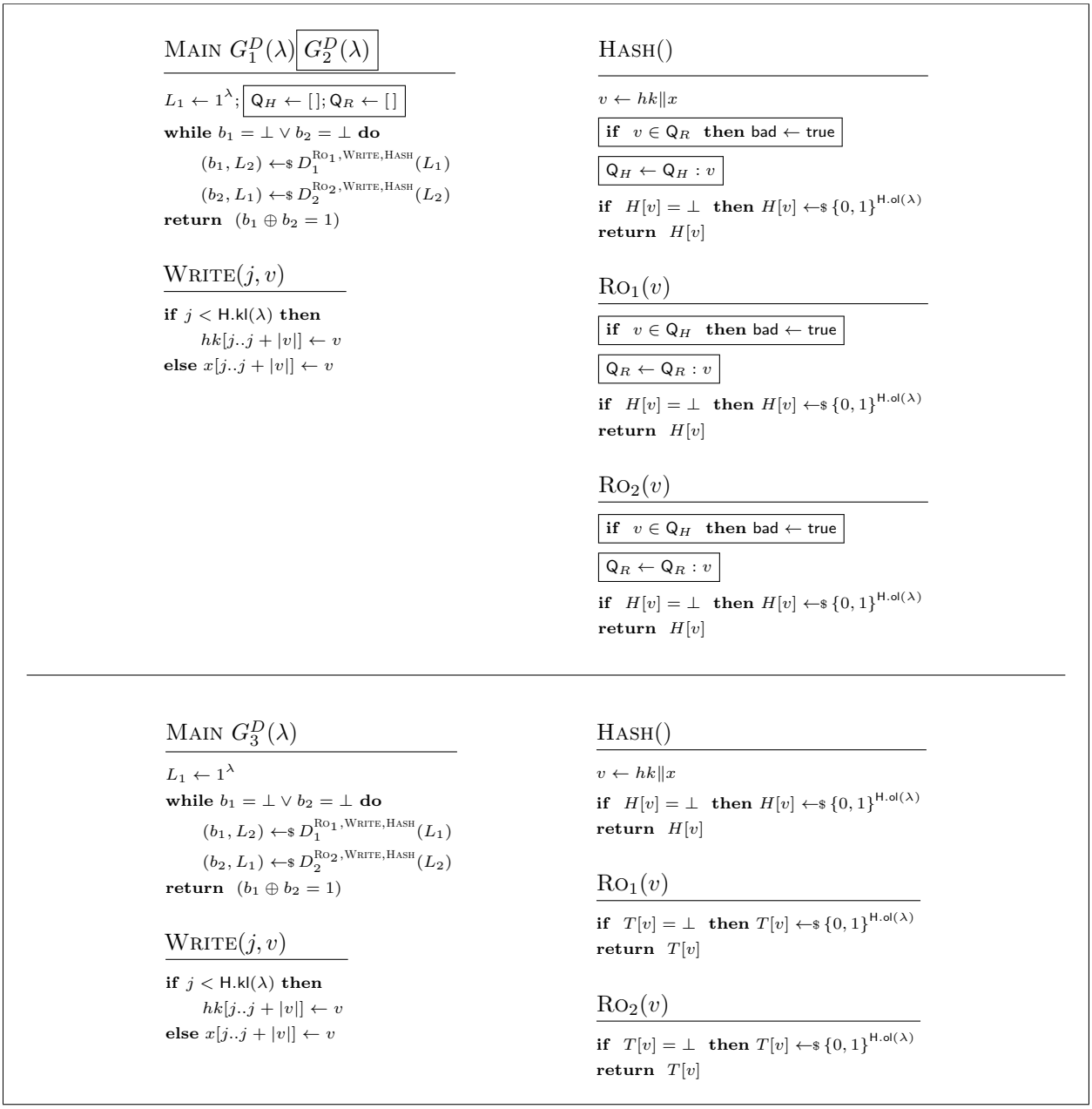

Fig. 9. Games for the random-oracle feasibility of ICEs.

We bound the proabrbility of each of these events in G_3 via the unpredictability of D_1 in the first case, and the unpredictability of D_2 in the second case.

THE PREDICTORS. We bound the probability that flag bad gets set via a D_1 query to RO_1 by constructing a predictor $P_1^{\mathcal{R}}$ against D_1 (event E_1 above). Predictor $P_1^{\mathcal{R}}$ uses a full view of D_1 provided in its inputs and $\mathcal{R}$ to perfectly simulate a run of D_1 in the ICE game with respect to an (independent) random implementation of the hash oracle (see remark on page 8). Hence $P_1^{\mathcal{R}}$ runs D_1 in an environment that is identical to G_3. Algorithm $P_1^{\mathcal{R}}$ picks a random query v of D_1 to RO_1 and outputs it. Whenever flag bad is set due to event E_1, algorithm $P_1^{\mathcal{R}}$ wins the prediction game as long as it correctly guesses the query that set bad. Hence, if we let $Q_1(\lambda)$ denote an upper bound on the number of RO_1 queries of D_1, we may conclude that

$$\Pr\left[G_3^D(\lambda) \text{ sets } \mathsf{bad} \text{ in } \text{RO}_1\right] \leq Q_1(\lambda) \cdot \Pr\left[\text{Pred}_{1,D}^{P_1}(\lambda)\right] \ .$$

The second predictor $P_2^{\mathcal{R}}$ corresponding to event E_2 is constructed and analyzed analogously:

$$\Pr\left[G_3^D(\lambda) \text{ sets } \mathsf{bad} \text{ in } \text{RO}_2\right] \leq Q_2(\lambda) \cdot \Pr\left[\text{Pred}_{2,D}^{P_2}(\lambda)\right] \ .$$

It follows that

$$\mathbf{Adv}^{\text{ice}}_{\mathsf{H},D}(\lambda) \leq \Pr\left[G_3^D(\lambda) \text{ sets } \mathsf{bad}\right] \leq Q_1(\lambda) \cdot \mathbf{Adv}^{\text{pred}}_{1,D,P_1}(\lambda) + Q_2(\lambda) \cdot \mathbf{Adv}^{\text{pred}}_{2,D,P_2}(\lambda) \ .$$

The theorem follows for $D \in \mathcal{C}^{\text{poly}} \cap \mathcal{C}^{\text{sup}}$.

□

D Proof of Theorem 2: VIL-ICEs from FIL-RO

In the following section we consider instantiatoions of ICEs via a hybrid between Zipper Hash [Lis07] and chop-MD [CDMP05] and analyzed in the ideal compression function model (where the compression function is assumed to be a fixed-input-length random oracle).

D.1 Chopped & keyed Zipper Hash

Liskov's Zipper Hash can be seen as a variant of the Merkle–Damgård construction [Mer90,Dam90]. The basic Merkle–Damgård construction $\mathsf{H}^{\mathsf{h}}(m_1, \ldots, m_\ell)$ is computed recursively as $\mathsf{H}^{\mathsf{h}}(m_1, \ldots, m_\ell) := \mathsf{h}(m_\ell, x_{\ell-1})$ where $x_0 := \mathsf{IV}$ is some initialization vector and $x_i := \mathsf{h}(m_i, x_{i-1})$ is computed as the compression function evaluated on the current message block and the last chaining value. In the Zipper Hash construction, the message is not passed once, but twice, the second time in reversed block order. In Liskov's original formulation of Zipper Hash [Lis07] two independent compression functions were used, one for the first message pass, and one for the second. For our proof, this is not necessary, and we consider only a single (idealized) compression function. We denote this construction with ZH^{h}.

The chop Merkle–Damgård (chop-MD) construction is essentially the basic Merkle–Damgård construction appended by a projection map that drops half of the bits of the final compression function evaluation [CDMP05]. We consider a hybrid construction: Zipper Hash with a final projection.

Zipper Hash is specified as an unkeyed hash function. We will, however, consider a keyed variant of Zipper Hash, and define $\mathsf{chop\text{-}KZIP}^{\mathsf{h}}$ as

$$\mathsf{chop\text{-}KZIP}^{\mathsf{h}}(hk, M) := g(\mathsf{ZH}^{\mathsf{h}}(hk \| M)) \ ,$$

where we assume that hk is padded to the length of one message block and g is a projection to the first half of its input. As a consequence, during the evaluation of $\mathsf{chop\text{-}KZIP}^{\mathsf{h}}(hk, M)$, the first call to the compression function as well as the last call, both take the key hk as input, which is crucial to proving that the construction is ICE secure. We give a schematic diagram of an evaluation of the resulting hybrid called *Chopped & Keyed Zipper Hash* ($\mathsf{chop\text{-}KZIP}$) in Figure 10. Note, that $\mathsf{chop\text{-}KZIP}$ is also a *key-prefixed hash construction* in the terminology of Mittelbach [Mit14].

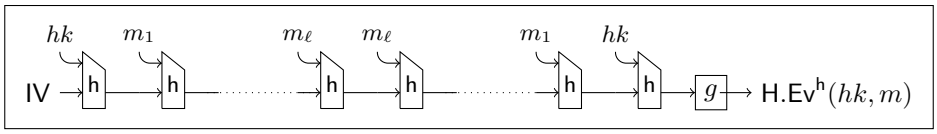

Fig. 10. The Zipper Hash construction [Lis07] merged with chop-MD [CDMP05] evaluated on a message $m_1 \| \ldots \| m_\ell := m$ where $|m| := \ell k$ is a multiple of the block size λ and hash key hk. The final node g corresponds to the projection of the first half of the output of h, that is, $g : \{0,1\}^n \to \{0,1\}^{n/2}$.

D.2 Proof for restricted ICE[$\mathcal{C} \cap \mathcal{C}^{\text{skup}}$]

To show that the keyed function $\mathsf{chop\text{-}KZIP}^{\mathsf{h}}$ is ICE[$\mathcal{C}$] secure for the class

$$\mathcal{C} := \mathcal{C}^{\text{poly}} \cap \mathcal{C}^{\text{cup}} \cap \mathcal{C}^{\text{dist}} \cap \mathcal{C}_1^0 \cap \mathcal{C}_1^{\text{1-hk}} \cap \mathcal{C}_2^{\text{0-hk}} \cap \mathcal{C}_2^{\varepsilon} \ ,$$

and an ideal compression function h, we first look at a more restricted class where the hash key hk remains statistically hidden from distinguisher D_2. We denote the corresponding class of adversaries $\mathcal{C}^{\text{skup}}$. We start with the following proposition.

Proposition 1. *The* $\mathsf{chop\text{-}KZIP}^{\mathsf{h}}$ *construction described above is an* ICE[$\mathcal{C} \cap \mathcal{C}^{\text{skup}}$]*-secure hash function for* $\mathcal{C}$ *as above and whenever* $\mathsf{h} : \{0,1\}^\mu \times \{0,1\}^n \longrightarrow \{0,1\}^n$ *is a FIL-RO.*

Mittelbach [Mit14] presents a composition theorem for multi-stage games which intuitively says that if a game is *unsplittable* for an iterative hash construction then security in the random-oracle model implies security for the iterative hash construction where the underlying compression function is ideal and assuming that the hash construction is indifferentiable from a random oracle. Thus, in order to prove the proposition 1 it is sufficient to show that:

(1) $\mathsf{chop\text{-}KZIP}$ is an iterative hash construction (within the framework of Mittelbach [Mit14]);
(2) $\mathsf{chop\text{-}KZIP}$ is indifferentiable from a random oracle;
(3) and the ICE game is unsplittable for $\mathsf{chop\text{-}KZIP}$for class $\mathcal{C} \cap \mathcal{C}^{\text{skup}}$.

When introducing Zipper Hash, Liskov also gave a proof of indifferentiability [Lis07]. It is easily seen that the function remains indifferentiable under composition with transformation g (dropping half of the bits). We consider (1) and (3) next.

$\mathsf{chop\text{-}KZIP}$ *is iterative.* Mittelbach [Mit14] gives a framework for hash functions H^{h} that are built by iterating a fixed-input-length compression function h. We describe a simplified version of the framework here and refer to [Mit14] for details. The idea is that the computation of a hash function can be described by a directed acyclic graph that is independent of the compression function h. In other words, for a given key hk and message M, there is an algorithm that constructs an execution graph which, basically, consists of message-block nodes (for message blocks m_i in message M) an IV-node (for the initialization vector), h-nodes representing compression function evaluations and a single g-node representing the final transformation. Furthermore, a universal algorithm Eval given oracle access to h and the unique execution graph for (hk, M) can compute value $\mathsf{H}^{\mathsf{h}}(hk, M)$. In Figure 10 we give the execution graph for a $\mathsf{chop\text{-}KZIP}$ evaluation for message $m_1 \| \ldots \| m_\ell = M$ and key hk (for ease of presentation we ignore padding).[15] To compute the corresponding hash value relative to function h the generic algorithm $\mathsf{Eval}^{\mathsf{h}}$ recursively performs the following steps on the execution graph: Search for a node that has no ingoing edges. If it is a message-block node or an IV-node then label all outgoing edges with the corresponding value (message-block or IV) and remove the node from the graph. If the node is an h-node or the single g-node such that all ingoing edges are labeled, then compute the corresponding function taking the labels of the ingoing edges as input, label all outgoing edges with the result and remove the node from the graph. Finally, return the result from the final g computation.

Zipper Hash, chop-MD and also $\mathsf{chop\text{-}KZIP}$ can be casted in this way and, indeed, Zipper Hash and chop-MD are examples used in [Mit14].

[15] In [Mit14] execution graphs are defined to have one message-block node per message-block m_i in M. Here, we relax this requirement to support the presentation. We refer to [Mit14] for a detailed description of how Zipper Hash is represented in that framework.

Unsplittability. It remains to show unsplittability for ICE[$\mathcal{C} \cap \mathcal{C}^{\text{skup}}$]. Unsplittability is defined relative to certain events regarding queries of adversaries to the compression function h. Consider a specific game $G^{\mathsf{H}^{\mathsf{h}},\mathcal{A}_1,\ldots,\mathcal{A}_m}$ with respct to an iterative hash construction H^{h} and adversarial procedures $\mathcal{A}_1,\ldots,\mathcal{A}_m$. In our specific case this would be the ICE game with adversary D. Now consider an execution of game $G^{\mathsf{H}^{\mathsf{h}},\mathcal{A}_1,\ldots,\mathcal{A}_m}$ on some random coins r. Then, we can formalize predicates on the compression function queries by the adversaries, for example, we say that the second h-query by adversary $\mathcal{A}_3$ has a certain property. A query is called an *initial query* if it is of the form (m, IV), that is, the second input to the compression function is the initialization vector. A query (m, x) to h is called a *chained query* if previously during the game there are h queries $(m_0, x_0), \ldots, (m_\ell, x_\ell)$ such that

$$x_0 = \mathsf{IV} \quad \text{and} \quad \forall i = 1, \ldots, \ell - 1 : x_{i+1} = \mathsf{h}(m_i, x_i) \quad \text{and} \quad x = \mathsf{h}(m_\ell, x_\ell) \ .$$

Note that here we consider all queries that were made during the game by the adversaries before the point in time where (m, x) is queried. A query is called a *result query*, if it is a chained query and the structure of the chain corresponds to a correct evaluation of Zipper Hash, that is: $hk, m_1, \ldots, m_\ell, m_\ell, \ldots, m_1, hk$ for some value hk and message blocks m_1 to m_ℓ. We say that an h-query (m, x) by an adversary $\mathcal{A}_i$ is *bad* if it is a chained query, but that it is not chained with respect to only the queries by adversary $\mathcal{A}_i$ that occurred before (m, x). Finally, a query is a *bad result query* if it is a result query and a bad query. We refer to [Mit14] for formal definitions.

We now recall the definition of unsplittability. Let H^{h} be an iterative hash function and let h be an ideal compression function. We say game G, where only adversarial procedures access h directly,[16] is *unsplittable* for H^{h} if for every efficient adversary $\mathcal{A}_1, \ldots, \mathcal{A}_m$ there exist efficient algorithms $\mathcal{A}_1^*, \ldots, \mathcal{A}_m^*$ and negligible function negl such that

$$\mathbf{Adv}^{G}_{\mathsf{H}^{\mathsf{h}},\mathcal{A}_1,\ldots,\mathcal{A}_m}(\lambda) \leq \mathbf{Adv}^{G}_{\mathsf{H}^{\mathsf{h}},\mathcal{A}_1^*,\ldots,\mathcal{A}_m^*}(\lambda) + \text{negl}(\lambda) \ ,$$

where $\mathbf{Adv}^{G}_{\mathsf{H}^{\mathsf{h}},\mathcal{A}_1,\ldots,\mathcal{A}_m}$ denotes the success probability of $(\mathcal{A}_1, \ldots, \mathcal{A}_m)$ in game G with hash construction H^{h}. Moreover, it holds for game $G^{\mathsf{H}^{\mathsf{h}},\mathcal{A}_1^*,\ldots,\mathcal{A}_m^*}$ that bad result queries occur only with negligible probability.

Unsplittability allows composition. We present a self-contained explanation of how unsplittability can be used to argue composition [Mit14].

Figure 11 (top) shows the ICE game with respect to $\mathsf{chop\text{-}KZIP}$ and the ICE game with respect to the random oracle is shown on the bottom. We next, give the intuition on how unsplittability allows us to go from an adversary in the Zipper Hash setting to an adversary in the random-oracle setting.

Let function $\mathsf{H}^{\mathcal{R}}$ denote the hash function with $\mathsf{H}.\mathsf{Ev}^{\mathcal{R}}(hk, m) := \mathcal{R}(hk \| m)$ which is ICE secure in the random-oracle model (see Theorem 1) for a superclass of adversaries considered here. From any efficient ICE adversary D we will, in several steps of which one is using the unsplittability property, construct an adapted adversary D^* such that

$$\Pr\left[\text{ICE}^{D}_{\mathsf{chop\text{-}KZIP}^{\mathsf{h}}}(\lambda)\right] \leq \Pr\left[\text{ICE}^{D^*}_{\mathsf{H}^{\mathcal{R}}}(\lambda)\right] + \text{negl}(\lambda)$$

and thus

$$\mathbf{Adv}^{\text{ice}}_{\mathsf{chop\text{-}KZIP}^{\mathsf{h}},D}(\lambda) \approx \mathbf{Adv}^{\text{ice}}_{\mathsf{H}^{\mathcal{R}},D^*}(\lambda)$$

by the composition theorem of Mittelbach. Note that on the left we are considering the ICE game with respect to $\mathsf{chop\text{-}KZIP}$ and the ideal compression function h (see also top of Figure 11), while on the right we have the ICE game with respect to hash construction $\mathsf{H}^{\mathcal{R}}$ in the random-oracle model (Figure 11, bottom), for which we know by Theorem 1 that

$$\mathbf{Adv}^{\text{ice}}_{\mathsf{H}^{\mathcal{R}},D^*}(\lambda) \leq \text{negl}(\lambda) \ .$$

[16] This corresponds to a functionality respecting game, in the terminology of Ristenpart, Shacham and Shrimpton [RSS11].

We will construct adversary D^* as

$$D_1^* := D_1'^{\mathsf{Sim}} \qquad\qquad D_2^* := D_2'^{\mathsf{Sim}}$$

where adversary D' will be the adversary guaranteed by the unsplittability property, and simulator $\mathsf{Sim}^{\mathcal{R}}$ is an indifferentiability simulator that uses the random oracle to simulate h for D'.

To construct the simulator $\mathsf{Sim}^{\mathcal{R}}$ we will use the generic indifferentiability simulator from [Mit14]. This generic simulator exists and is a *good* indifferentiability simulator, if the hash function in question is an iterative hash construction and is indifferentiable from a random oracle. As discussed, both properties hold in case of chop-KZIP.

We briefly sketch how the generic simulator works, and refer to [Mit14] for a detailed discussion. Simulator $\mathsf{Sim}^{\mathcal{R}}$ keeps an internal list of all queries it receives. (Note that each adversary runs an independent copy of $\mathsf{Sim}^{\mathcal{R}}$.) On query (m, x) simulator $\mathsf{Sim}^{\mathcal{R}}$ checks if this query is "part of a valid execution" of chop-KZIP, that is, whether the internal list of queries contains a chain of queries starting from an initial query and leads to this query.

The simulator distinguishes between result queries and non-result queries. If it sees all the queries in the chain of a result query, it is easily seen that the simulator can reconstruct key hk and message M and, thus, recognize a result query. If so it uses its access to the random oracle to return $\mathcal{R}(hk\|M)$. Else, if a query (m, x) is not a result query, then the simulator responds with a randomly chosen value. We stress that this random value does not depend on the state of the simulator but only on the query and the simulator's random coins (which is important for consistency across different instances of the simulator).

For a single-stage game, the story ends here. The single adversary composed with the simulator will be as good when playing in the random-oracle setting as it is when playing in the chop-KZIP setting. For multi-stage games, however, this does not directly work, as queries by different adversarial stages (in our case D_1 and D_2) must be answered consistently. This difference between multi-stage and single-stage games was first noted by Ristenpart, Shacham and Shrimpton [RSS11]. That is (1) the same query must get the same answer, and (2) if D_1 starts to compute a hash value, sends an intermediate compression function result to D_2, and D_2 continues the evaluation (note that this corresponds to a bad query) we need to argue that the simulator can still detect the final result query.

Unsplittability, solves problem (2). Given an adversary D we can build an adversary D' that has the same advantage in the ICE game, but does not make bad result queries. To solve problem (1), Mittelbach [Mit14] uses a derandomization technique due to Bennet and Gill [BG81] that allows to derandomize the simulator relative to the adversary using the random oracle. Hence any non-potential hash query is answered with a random value that is now generated deterministically (relative to the random oracle) and, thus, the same query (m, x) will be answered identically by any instance of simulator $\mathsf{Sim}^{\mathcal{R}}$.

This yields that if we compose adversary D' with the derandomized simulator then this adversary will be (almost) as good in the random-oracle setting as adversary D in the chop-KZIP setting.

ICE$[\mathcal{C} \cap \mathcal{C}^{\text{skup}}]$ *is unsplittable for* chop-KZIP. We now turn to showing that ICE$[\mathcal{C} \cap \mathcal{C}^{\text{skup}}]$ is unsplittable for chop-KZIP. Assume that adversary D during the ICE game makes a bad result query, that is, D_1, or D_2 makes a query to the compression function without having queried all intermediary queries in the corresponding chain. We have to show that in this case an adversary D' exists which is at least as good but, with overwhelming probability, does not make bad result queries. We will show that the original adversary is already of this form.

As h is ideal, a bad query means that an intermediary value must have been communicated from one stage to another (also see [Mit13, Lemma 3.3]). Note that there is direct communication from D_1 to D_2 and indirect communication from D_2 to D_1 via the HASH oracle (i.e., we restrict the second distinguisher to be in $\mathcal{C}_2^{\varepsilon}$). For the second case, note that via calls to HASH distinguisher D_1 receives value chop-KZIP(hk, M) for whatever message M was written on the shared tape. Since, chop-KZIP only returns half of the bits of the final compression function computation distinguisher D_1 does not learn a full result of a compression function. Together with the restriction, that distinguishers must always query the HASH oracle on distinct messages we can, thus, conclude that only D_2 can make *bad* queries.

Now assume that D_2 makes a *bad* result query. Then, by the definition of result query for chop-KZIP, distinguisher D_2 uses hash key hk. We define efficient predictor P as follows. Predictor P takes as input the view of D_2 and outputs one of its h-queries at random. If $Q(\lambda)$ is an upper bound on the number of h-queries by D_2 then predictor P outputs hash key hk with probability $1/Q(\lambda)$ times the probability that D_2 makes a bad result query. This, however, directly contradicts the assumption that adversary D is in class $\mathcal{C}^{\text{skup}}$, the class of adversaries where hash key hk remains statistically hidden from D_2.

It follows that ICE[$\mathcal{C} \cap \mathcal{C}^{\text{skup}}$] is unsplittable for chop-KZIP and the proposition follows with the composition theorem for unsplittable games [Mit13, Theorem 4.2].

D.3 chop-KZIP is ICE[$\mathcal{C}$] secure

In this section we extend the previous discussion to general $\mathcal{C}$ adversaries where the second distinguisher may or may not learn the hash key.

Theorem 2 (restated from page 12). *The VIL hash function* $\mathsf{chop\text{-}KZIP}^{\mathsf{h}}$ *constructed above is* ICE[$\mathcal{C}$] *secure in the FIL-RO model for* $\mathsf{h} : \{0,1\}^{\mu} \times \{0,1\}^{n} \longrightarrow \{0,1\}^{n}$ *for the class*

$$\mathcal{C} := \mathcal{C}^{\text{poly}} \cap \mathcal{C}^{\text{sup}} \cap \mathcal{C}^{\text{dist}} \cap \mathcal{C}_1^{\text{1-hk}} \cap \mathcal{C}_1^{0} \cap \mathcal{C}_2^{\text{0-hk}} \cap \mathcal{C}_2^{\varepsilon} \ .$$

An analogous result holds for polynomial-time distinguishers that are only computationally unpredictable.

Proof. Given the above discussion it remains to show that distinguisher D_2 does not make any *bad* result queries. Let us assume the contrary, i.e., we fix the randomness r of game ICE[$\mathcal{C}$] and assume that distinguisher D_2 makes a bad result query (m_b, x_b) to compression function h. Then, by the definition of a bad query, the query is not properly chained, that is, it is chained with respect to all h-queries by D_1 *and* D_2, but it is not chained with respect to only the h-queries by D_2. Let $\mathsf{qry}^{\mathsf{h}}_{D_i}[r]$ denote the sequence of h-queries by distinguisher D_i (for $i \in \{1,2\}$) up to and (possibly) including the bad result query (m_b, x_b). We distinguish two types of bad result queries BQ-TYPE:

BQ-TYPE$_1$: The type BQ-TYPE$_1$ corresponds to the bad query appearing in the first message pass for a message $M = m_1 \| \ldots, m_\ell$. In Figure 12 we illustrate how the hash function evaluation is divided over the two distinguishers. Formally, there exists an index $j \leq \ell + 1$ such that

$$(m_0, x_0), \ldots, (m_j, x_j) \in \mathsf{qry}^{\mathsf{h}}_{D_1}[r]$$

and

$$(m_{j+1}, x_{j+1}), \ldots, (m_\ell, x_\ell), (m_\ell, x_{\ell+1}), \ldots, (m_0, x_{2\ell}) \in \mathsf{qry}^{\mathsf{h}}_{D_2}[r]$$

for $x_i = \mathsf{h}(m_{i_1}, x_{i-1})$ and $(m_0, x_0) := (hk, \mathsf{IV})$.

BQ-TYPE$_2$: Type BQ-TYPE$_2$ corresponds to the bad query appearing in the second message pass. Formally, there exists index $\ell + 1 \leq j \leq 2\ell$ such that

$$(m_0, x_0), \ldots, (m_\ell, x_\ell), (m_\ell, x_{\ell+1}), \ldots, (m_j, x_j) \in \mathsf{qry}^{\mathsf{h}}_{D_1}[r]$$

and

$$(m_{j+1}, x_{j+1}), \ldots, (m_0, x_{2\ell}) \in \mathsf{qry}^{\mathsf{h}}_{D_2}[r]$$

for $x_i = \mathsf{h}(m_{i_1}, x_{i-1})$ and $(m_0, x_0) := (hk, \mathsf{IV})$.

In the following we construct an adapted distinguisher that does not make any bad result queries. For this we argue that the distinguisher does not make a bad result query in either of the above two cases: We consider the ICE game together with the following adversary (D_1', D_2'). Before we describe the distinguishers in detail let us describe the underlying idea.

Result queries are always of the form (hk, x), that is the key is used in the first slot (cf. Figure 12). In order for D_2' to recognize such queries (and perform adequate steps) it must, thus, be able to recognize key hk. We

$\underline{\mathrm{ICE}^{D}_{\mathsf{chop\text{-}KZIP}^{\mathsf{h}}}(\lambda)}$

$b \leftarrow\$ \{0,1\}; b' \leftarrow \bot; L_1 \leftarrow 1^{\lambda}$
while $b_1 = \bot \vee b_2 = \bot$ **do**
  $(b_1, L_2) \leftarrow\$ D_1^{\mathsf{h}_1, \textsc{Write}, \textsc{Hash}}(L_1)$
  $(b_2, L_1) \leftarrow\$ D_2^{\mathsf{h}_2, \textsc{Write}, \textsc{Hash}}(L_2)$
return $(b_1 \oplus b_2 = b)$

$\underline{\mathsf{h}_1(x)}$

if $T_{\mathsf{h}}[x] = \bot$ **then** $T_{\mathsf{h}}[x] \leftarrow\$ \{0,1\}^{n/2}$
return $T_{\mathsf{h}}[x]$

$\underline{\mathsf{h}_2(x)}$

if $T_{\mathsf{h}}[x] = \bot$ **then** $T_{\mathsf{h}}[x] \leftarrow\$ \{0,1\}^{n/2}$
return $T_{\mathsf{h}}[x]$

$\underline{\mathsf{h}_{\mathsf{H}}(x)}$

if $T_{\mathsf{h}}[x] = \bot$ **then** $T_{\mathsf{h}}[x] \leftarrow\$ \{0,1\}^{n/2}$
return $T_{\mathsf{h}}[x]$

$\underline{\textsc{Write}(j, v)}$

if $j < \mathsf{H.kl}(\lambda)$ **then**
  $hk[j..j+|v|] \leftarrow v$
else $x[j..j+|v|] \leftarrow v$

$\underline{\textsc{Hash}()}$

if $T[x] = \bot$ **then**
  if $b = 1$ **then**
    $T[x] \leftarrow \mathsf{chop\text{-}KZIP}^{\mathsf{h}_{\mathsf{H}}}(hk, x)$
  else $T[x] \leftarrow\$ \{0,1\}^{n/2}$
return $T[x]$

$\underline{\mathrm{ICE}^{D}_{\mathsf{H}^{\mathcal{R}}}(\lambda)}$

$b \leftarrow\$ \{0,1\}; b' \leftarrow \bot; L_1 \leftarrow 1^{\lambda}$
while $b_1 = \bot \vee b_2 = \bot$ **do**
  $(b_1, L_2) \leftarrow\$ D_1^{\mathsf{h}_1, \textsc{Write}, \textsc{Hash}}(L_1)$
  $(b_2, L_1) \leftarrow\$ D_2^{\mathsf{h}_2, \textsc{Write}, \textsc{Hash}}(L_2)$
return $(b_1 \oplus b_2 = b)$

$\underline{\mathsf{h}_1(x)}$

return $\mathsf{Sim}(x)$

$\underline{\mathsf{h}_2(x)}$

return $\mathsf{Sim}(x)$

$\underline{\mathsf{Sim}^{\mathcal{R}}(x)}$

run generic chop-KZIP simulator [Mit14,Mit13] derandomized via [BG81] on input x and wrt. random oracle $\mathcal{R}$.

$\underline{\textsc{Write}(j, v)}$

if $j < \mathsf{H.kl}(\lambda)$ **then**
  $hk[j..j+|v|] \leftarrow v$
else $x[j..j+|v|] \leftarrow v$

$\underline{\textsc{Hash}()}$

if $T[x] = \bot$ **then**
  if $b = 1$ **then**
    $T[x] \leftarrow \mathcal{R}(hk \| x)$
  else $T[x] \leftarrow\$ \{0,1\}^{n/2}$
return $T[x]$

$\underline{\mathcal{R}(x)}$

if $R[x] = \bot$ **then** $R[x] \leftarrow\$ \{0,1\}^{n/2}$
return $R[x]$

Fig. 11. On the top we depict the ICE game relative to chop-KZIP in the ideal-compression-function model; that is, where compression function h which is used by chop-KZIP is assumed to be a fixed-input-length random oracle to which all parties have access to. On the bottom we give the target ICE game relative to hash construction $\mathsf{H}^{\mathcal{R}}$ with $\mathsf{H.Ev}^{\mathcal{R}}(hk, m) := \mathcal{R}(hk \| m)$ where $\mathcal{R}$ is a random oracle. The simulator $\mathsf{Sim}^{\mathcal{R}}$ simulates h for the adversary. Note that h takes values in $\{0,1\}^n$ and due to chopping chop-KZIP has values in $\{0,1\}^{n/2}$.

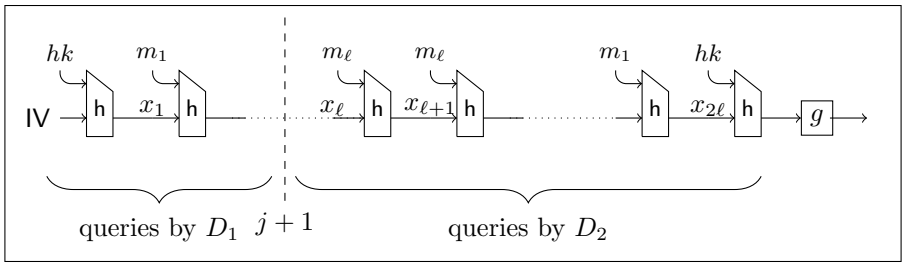

Fig. 12. Splitting of an execution of $\mathsf{chop\text{-}KZIP.Ev}^{\mathsf{h}}(hk, M)$ for some message $M = m_1 \| \ldots \| m_\ell$ where the first distinguisher computes the first $j+1$ compression function calls up to and including $\mathsf{h}(m_j, x_j)$, and the second distinguisher computes the remainder.

can, however, not simply forward the key from D_1' to D_2' since this might lead to the distinguishers making predictable HASH queries. Instead D_1' forwards a random value $\mathsf{TestKey}$ together with value $\mathsf{h}(hk, \mathsf{TestKey})$. This allows D_2' to test if a given query from D_2 contains key hk and at the same time hides the key from D_2' unless it is communicated by D_1.

Now that D_2' can recognize potential result queries, D_2' can ensure that the above described types of bad result queries do not occur. For type $\mathsf{BQ\text{-}TYPE}_1$ note that the entire second message pass is performed by D_2. On seeing a result query D_2 can, thus, extract the corresponding message M and recompute $\mathsf{chop\text{-}KZIP.Ev}^{\mathsf{h}}(hk, M)$ thereby avoiding bad result queries of type $\mathsf{BQ\text{-}TYPE}_1$. For bad result queries of type $\mathsf{BQ\text{-}TYPE}_2$ we use a similar trick as before and let D_1' forward the necessary information to recognize and properly answer such queries without needing to call h (and thereby avoiding the bad result query). For this, we let D_1' choose two additional random values $\mathsf{TestVal}$ and $\mathsf{BlindVal}$ (which it also forwards to D_2'). Furthermore, D_1' keeps track of all h queries by D_1 and constructs all (partial) chains. For each of those partial chains it then reconstructs the corresponding message M—message M consists j message blocks if the chain consists of $j+1$ queries; note that the first query contains key hk—and computes $\mathsf{chop\text{-}KZIP.Ev}^{\mathsf{h}}(hk, M)$. Let query (hk, x_{2j}) be the final h query in that computation. Distinguisher D_1' computes values

$$\mathsf{h}(\mathsf{TestVal}, x_{2j}) \qquad \text{and} \qquad \mathsf{h}(\mathsf{BlindVal}, x_{2j}) \oplus \mathsf{h}(hk, x_{2j})$$

which it also forwards to distinguisher D_2'. Given these two values, distinguisher D_2' can check if a potential result query (i.e., a query of the form (hk, x)) is equivalent to a precomputed result query by D_1' by comparing $\mathsf{h}(\mathsf{TestVal}, x)$ to the values received from D_1'. If so, it can recover value $\mathsf{h}(hk, x_{2j})$ from $\mathsf{h}(\mathsf{BlindVal}, x_{2j}) \oplus \mathsf{h}(hk, x_{2j})$ without having to make the bad result query. Further note, that the additionally leaked values statistically hide key hk as well as x_{2j} and $\mathsf{h}(hk, x_{2j})$ as long as the number of oracle h queries is bounded by a polynomial.

We now describe the distinguishers in detail.

Distinguisher D_1'. On the first invocation distinguisher D_1' chooses values

$$\mathsf{TestKey} \in \{0,1\}^n \qquad \text{and} \qquad \mathsf{BlindVal}, \mathsf{TestVal} \in \{0,1\}^\mu \ .$$

Distinguisher D_1' runs distinguisher D_1 forwarding oracle calls to its own oracles and keeping track of all h queries in set Q. When distinguisher D_1 finishes and outputs leakage L, then distinguisher D_1' stops and outputs the following leakage L'. From all queries to h it constructs all valid (partial) chains C as follows: remove query (hk, IV) from Q and add $\mathsf{chain}_1 := [(hk, \mathsf{IV})]$ to C, where chain_1 is a sequence of tuples containing only tuple (hk, IV). Repeat the following steps until Q is empty, or no new chain was constructed in the last step. For each $\mathsf{chain}_i \in C$ let (m_i, x_i) be the last tuple in the chain. If there exists query $(m, x) \in Q$ such that $\mathsf{h}(m_i, x_i) = x$ then create a new chain $\mathsf{chain} \leftarrow [\mathsf{chain}_i \| (m, x)]$ whereby $[\mathsf{chain}_i \| (m, x)]$ we mean the chain that contains all tuples from chain_i and as (additional) final tuple in the sequence tuple (m, x). Add chain to C.

In a next step distinguisher D_1' further processes each chain. Let $T \leftarrow [\,]$ be an empty table. For each $\mathsf{chain} \in C$ the following steps are performed. If $|\mathsf{chain}| = 1$ (that is the chain only consists of query (hk, IV))

then nothing is done. Else, set $j := |\mathsf{chain}| - 1$. The distinguisher extracts $M = m_1, \ldots, m_j$ from chain chain where m_i corresponds to the first entry of the ith tuple. It then computes $\mathsf{chop\text{-}KZIP.Ev}^{\mathsf{h}}(hk, M)$. Let (hk, x_{2j}) be the final h query in the computation of $\mathsf{chop\text{-}KZIP.Ev}^{\mathsf{h}}(hk, M)$. It then sets

$$T[\mathsf{h}(\mathsf{TestVal}, x_{2j})] \leftarrow \mathsf{h}(\mathsf{BlindVal}, x_{2j}) \oplus \mathsf{h}(hk, x_{2j})$$

and prepares its leakage as

$$L' \leftarrow \Big(\mathsf{TestKey}, \mathsf{TestVal}, \mathsf{BlindVal}, \mathsf{h}(hk, \mathsf{TestKey}), T, L\Big) \ .$$

Distinguisher D_2'. Distinguishers D_2' parses the leakage as

$$(\mathsf{TestKey}, \mathsf{TestVal}, \mathsf{BlindVal}, \mathsf{h}(hk, \mathsf{TestKey}), T, L) \ ,$$

and runs distinguisher D_2. It forwards oracle queries to its own oracles keeping track of h-calls in set Q and furthermore handling h-calls as follows: on an h query (m, x) by distinguisher D_2, distinguisher D_2' computes $\mathsf{h}(m, \mathsf{TestKey})$ and compares it to $\mathsf{h}(hk, \mathsf{TestKey})$. If the two values do not match, then it forwards the query to h and returns the result to D_2. If, on the other hand $\mathsf{h}(m, \mathsf{TestKey}) = \mathsf{h}(hk, \mathsf{TestKey})$, then distinguisher D_2' further distinguishes two cases: $T[\mathsf{h}(\mathsf{TestVal}, x)] = \perp$ and $T[\mathsf{h}(\mathsf{TestVal}, x)] \neq \perp$. In the first case, where table T does not contain an entry for value $\mathsf{h}(\mathsf{TestVal}, x)$ distinguisher D_2' reconstructs all (tail) chains from Q that end in query (m, x). This can be done with a similar process as the chain reconstruction by D_1'. For each chain it extracts the corresponding message M and computes $\mathsf{chop\text{-}KZIP.Ev}^{\mathsf{h}}(hk, M)$ where we denote by (hk, x_{2j}) the final h query. If for any of these $x_{2j} = x$ then D_2' returns the result of $\mathsf{h}(hk, x_{2j})$ to D_2. If this is not the case for all reconstructed chains it queries h on (hk, x) and forwards the result to D_2.

In case table T contains a value for $\mathsf{h}(\mathsf{TestVal}, x)$, that is, $T[\mathsf{h}(\mathsf{TestVal}, x)] \neq \perp$, then distinguisher D_2' returns

$$T[\mathsf{h}(\mathsf{TestVal}, x) \oplus \mathsf{h}(\mathsf{BlindVal}, x)$$

to D_2. When D_2 stops and outputs a bit b, then so does D_2.

Analysis. We need to argue that (1) the adapted adversary D' has a similar advantage as the original adversary, (2) bad result queries in the ICE game with adversary D' and hash construction $\mathsf{chop\text{-}KZIP}^{\mathsf{h}}$ (for ideal h) occur only with negligible probability and (3) $D' \in \mathcal{C}$, that is, the HASH queries by D_1' and D_2' are statistically unpredictable.

For (1) note that unless D_1 or D_2 make an h query containing any of the random values $\mathsf{TestVal}$, $\mathsf{BlindVal}$, $\mathsf{TestKey}$ and furthermore no h-collision occurs then the view of D_1 and D_2 simulated by D_1' and D_2' is perfect. The collision probability is negligible since h is ideal and since values $\mathsf{TestVal}, \mathsf{BlindVal}, \mathsf{TestKey}$ are chosen uniformly at random and not given to D_1 nor D_2 the probability of either distinguisher guessing any of the three constants is also negligible.

For (2) note that the probability of guessing a value within a hash chain for an iterative hash function without honestly computing the chain is negligible (see Lemma 3.3 and its strengthened version in [Mit13]). As intuition note that h is ideal and thus a value $\mathsf{h}(m, x)$ has n bits of min-entropy. As the result of each h query goes into the next query in a chain this entropy is, so to speak, passed on. It follows that for potential bad result queries of type $\mathsf{BQ\text{-}TYPE}_1$ distinguisher D_2' will with overwhelming probability recover the correct message M. Similarly for potential bad result queries of type $\mathsf{BQ\text{-}TYPE}_2$ table T will with overwhelming probability contain the necessary information to answer the query without making the bad result query.

Finally, for (3) note that all that is additionally leaked by D_1' are the randomly chosen values $\mathsf{TestVal}$, $\mathsf{BlindVal}$, $\mathsf{TestKey}$ and the result of specially crafted h-queries of the forms:

$$\mathsf{h}(hk, \mathsf{TestKey}) \quad \text{and} \quad \mathsf{h}(\mathsf{TestVal}, x_{2j}) \quad \text{and} \quad \mathsf{h}(\mathsf{BlindVal}, x_{2j}) \oplus \mathsf{h}(hk, x_{2j})$$

All these values are statistically hidden from D_1 and D_2 since values $\mathsf{TestVal}$, $\mathsf{BlindVal}$, $\mathsf{TestKey}$ remain hidden. Furthermore, values hk, x_{2j} and $\mathsf{h}(hk, x_{2j})$ remain hidden unless values hk or x_{2j} can be reconstructed given the view of distinguisher D_2. For this note that also the unbounded predictors can only make polynomially many h queries. □