A preliminary version of this paper appeared in:
T. Ristenpart, H. Shacham, and T. Shrimpton. Careful with Composition: Limitations of the Indifferentiability Framework. *Advances in Cryptology – EUROCRYPT '11*, LNCS vol. 6632, pp. 487–506, Kenneth Paterson ed., Springer, 2011.

Careful with Composition: Limitations of Indifferentiability and Universal Composability

Thomas Ristenpart* Hovav Shacham† Thomas Shrimpton‡

June 22, 2011

Abstract

We exhibit a hash-based storage auditing scheme which is provably secure in the random-oracle model (ROM), but easily broken when one instead uses typical indifferentiable hash constructions. This contradicts the widely accepted belief that the indifferentiability composition theorem applies to *any* cryptosystem. We characterize the uncovered limitation of the indifferentiability framework by showing that the formalizations used thus far implicitly exclude security notions captured by experiments that have multiple, disjoint adversarial stages. Examples include deterministic public-key encryption (PKE), password-based cryptography, hash function nonmalleability, key-dependent message security, and more. We formalize a stronger notion, reset indifferentiability, that enables an indifferentiability-style composition theorem covering such multi-stage security notions, but then show that practical hash constructions cannot be reset indifferentiable. We discuss how these limitations also affect the universal composability framework. We finish by showing the chosen-distribution attack security (which requires a multi-stage game) of some important public-key encryption schemes built using a hash construction paradigm introduced by Dodis, Ristenpart, and Shrimpton.

1 Introduction

The indifferentiability framework of Maurer, Renner, and Holenstein (MRH) [51] supports modular proofs of security for cryptosystems. It can be viewed, informally, as a special case of the Universal Composition (UC) framework [27, 53], which inspired it. A crucial application of the indifferentiability framework has been to allow proofs in the random oracle model (ROM) [19] to be transferred to other idealized models of computation, where a monolithic random oracle is replaced by a hash function constructed from (say) an ideal compression function. This happens via an elegant composition theorem, the usual interpretation of which is: A proof of security for an arbitrary cryptosystem using functionality F (e.g., a random oracle) continues to hold when the cryptosystem instead uses a second functionality F' (e.g., a hash function built from an ideal compression function), so long as F' is *indifferentiable* from F.

* Department of Computer Sciences, University of Wisconsin–Madison, USA. Email: rist@cs.wisc.edu URL: http://pages.cs.wisc.edu/~rist/

† Department of Computer Science and Engineering, University of California, San Diego, USA. Email: hovav@cs.ucsd.edu URL: http://www.cs.ucsd.edu/~hovav/

‡ Department of Computer Science, Portland State University, Portland, USA. Email: teshrim@cs.pdx.edu URL: http://www.cs.pdx.edu/~teshrim/

In this paper, we show that this interpretation is too generous. We uncover an application (in the context of secure distributed storage) for which composition fails completely. For this application there is a simple scheme provably secure in the ROM, and yet easily broken when using typical indifferentiable hash constructions. We then begin an exploration of the fall out.

RANDOM ORACLES AND INDIFFERENTIABILITY. Let us give a bit more background on why indifferentiability has proved so useful. A wide range of practical, in-use cryptographic schemes enjoy proofs of security in the ROM [19]; for some schemes, ROM proofs are the only ones known. But most in-use hash-function constructions are not suitable for modeling as a RO, even when assuming the primitive underlying the hash function is ideal (e.g., an ideal compression function), because they admit length-extension attacks [56]. These attacks abuse the structure of the iterative modes-of-operation underlying hash functions such as MD5, SHA-1, and SHA-2. And the weakness they expose has led to practical insecurities [36]. Of course, we can build hash functions that resist known length-extension attacks, but it remains unclear whether the resulting functions would also prevent other, unforeseen structure-abusing attacks.

Coron et al. [31] instead suggest an approach to design hash functions that "behave like" random oracles in a provable sense. Specifically, this requires that a hash function will provide security anywhere a random oracle would. The MRH composition theorem seems to give exactly this, taking $F = \mathrm{RO}$ and $F' = H^f$, the latter being a hash function constructed from an (ideal) primitive f. Thus the needed hash function property is that H^f be indifferentiable from a RO. Importantly, this approach preserves proofs of security as well: the MRH theorem transports a cryptosystem's proof of security in the ROM to a proof of security when using an indifferentiable hash function. A number of recent works prove constructions to be indifferentiable from a RO (e.g., [5, 18, 21, 30, 34, 35, 41]), including many candidates for the NIST SHA-3 competition. Given all this, the consensus opinion appears to be that indifferentiability exactly captures "behaving like" a RO, rules out structure-abusing attacks, and that once a cryptosystem is proven in the ROM it is secure using any compatible indifferentiable hash construction. We now describe an application that shows this consensus opinion to be wrong.

HASH-BASED STORAGE AUDITING. In the design of secure distributed systems, the following problem arises: How can parties in a system verify that a storage server is actually storing the files that it should be? A malicious server might tamper arbitrarily with the data entrusted to it; a rational one might discard the file to save space if detection is unlikely. This problem has received much attention since being formalized in 2007 [7, 43]. The particular example we consider in this paper is inspired by a proof-of-storage challenge-response protocol proposed as part of an earlier system, SafeStore [46]. Consider the following protocol. The client sends a random challenge C to the server; the server proves possession of the file M by computing $Z \leftarrow \mathsf{Hash}(M \parallel C)$ using a cryptographic hash function Hash and sending Z to the client, who performs the same computation using her copy of M and compares the result to that sent by the server.

Suppose, for simplicity, that both the file M and the challenge C are d bits long, and consider the case that $\mathsf{Hash} = H^f$, where f is an ideal compression function outputting strings of length $n < d$ bits and H returns the first $n/2$ bits of $f(f(IV, M), C)$. (IV is a fixed constant string.) This construction was shown indifferentiable from a RO in [31]. Thus, the MRH composition theorem combined with the fact that the protocol is secure in the ROM assuredly proves that the protocol is secure when using H^f. Quite baffling, then, is the observation that the server can cheat! The server simply computes $Y \leftarrow f(IV, M)$ when it first gets M, and then deletes M and stores the (shorter) Y. To answer a challenge C, the server computes $Z \leftarrow f(Y, C)$ and returns the first half of Z as its response. The client's check will succeed even though M is not stored.

The attack abuses a structural feature of typical hash functions that we call online computability. A hash function has this property when it can process its input in successive blocks, storing only a small amount of internal state between blocks. This property is desirable in practice and all indifferentiable hash constructions suggested for practical use have it (see, e.g., [5, 18, 21, 30, 34, 41]). As our example shows, however, online computability can be abused.

Let us pause to take stock of the situation. In Section 4 we prove that the SafeStore-inspired auditing scheme is, indeed, secure in the ROM. The proof of indifferentiability for our $\mathsf{Hash} = H^f$ provided by Coron et al. [31] and the proof of the MRH composition theorem are also both correct. But the server is still somehow able to abuse the structure of H^f. So what is going on here?

CHARACTERIZING THE PROBLEM. The gap is that the *MRH theorem does not apply*. The problem is subtle. Carefully revisiting the MRH theorem and its proof, we find that (loosely speaking) they only apply when a cryptosystem's security is measured relative to a security game using a single, stateful adversary. For example, left-or-right indistinguishability [38] for encryption schemes and unforgeability under chosen message attacks [39] each use just a single, stateful adversary. But the security of the challenge-response auditing protocol we just described is fundamentally two-stage. In the first stage, the adversary (the server) receives the message M, derives from M some state st that is smaller than the size of M, and forgets M. In a second stage it attempts to answer challenges using just st. This is an example of what we call a multi-stage game, a notion we will make formal.

In prior treatments of indifferentiability, the restriction to single-stage games is implicit in the underlying formalization of cryptosystems and adversaries. This restriction has not been mentioned in the literature, and our sense is that no researchers (until now) realized it. For completeness, we restate the MRH indifferentiability composition theorem and give its proof for single-stage games (see Section 3). We note that the same implicit limitations exist in the Universal Composability (UC) framework [27, 53]. While it has previously been observed in the context of collusion-free protocols [3, 4, 44] that UC does not cater to multiple, disjoint adversaries, we are the first to observe the broad implications of this restriction. We discuss UC in more detail in Section 7.

REPERCUSSIONS. We do not necessarily expect that practitioners would (or have) deployed the hash-based auditing scheme above. One can simply use $H^f(C \,\|\, M)$ to achieve (provable) security, and in fact this is the actual protocol used in SafeStore [46]. But the flaw this example uncovers is that the common interpretation of composition might actually *encourage* use of an insecure auditing mechanism. This is exactly the opposite of how provable security should guide protocol design.

All of this casts doubt on the security of any scheme relative to a multi-stage game. The scheme may well have provable security in the ROM, but this does not imply the inexistence of dangerous structure-abusing attacks, even when using indifferentiable hash constructions. And unfortunately the use of multi-stage games is widespread. The recent security notions for deterministic [10, 14, 24], hedged [11], and efficiently searchable [10] public-key encryption (PKE) are all multi-stage. Key-dependent message security [22] is multi-stage. Related-key attack security [16] when one allows ideal-primitive-dependent related key functions [2] is multi-stage. When formalizing password-based cryptography (e.g. [17, 57]) to allow arbitrary, hash-dependent password sampling algorithms, one uses multi-stage games. A recently proposed hash function nonmalleability security notion [23] is multi-stage. Interestingly, this is the only notion (we are aware of) that formalizes security against length-extension attacks, and so although we expect them to, we do not have *proof* that current indifferentiable hash constructions resist length-extension attacks.

STORAGE-AUGMENTED PRIMITIVES. So, we cannot generically use composition to modularly argue security in the context of multi-stage games. But it could be that indifferentiability remains a sufficient property to establish security in settings beyond hash-based challenge-response auditing. One might hope to prove, without relying on the MRH composition theorem, that a ROM proof of (say) a deterministic PKE scheme holds still when using any indifferentiable hash construction. This seems reasonable since for the applications just listed, online computability of the hash function does not obviously compromise security.

We give a simple mechanism to show that such proofs do not exist: augmenting an underlying ideal primitive f with the extra functionality of adversarial storage easily yields counterexamples. This is because the ability to store state in the underlying ideal primitive does not affect proving indifferentiability, but it does enable trivial attacks in all multi-stage settings that we know of. The conclusion, discussed further in Section 5, is that indifferentiability does not imply security in the multi-stage settings mentioned above.

We note that this does leave open the possibility for general results showing that indifferentiability for a certain restricted class of underlying primitives f is sufficient for security in some multi-stage settings. We leave finding such general positive results as an important open question.

RESET INDIFFERENTIABILITY. We present a new notion, reset indifferentiability, that does admit a composition theorem covering both single-stage and multi-stage games. In the indifferentiability framework, functionalities have both an honest and an adversarial interface, e.g. $F.hon$, $F.adv$ and $F'.hon, F'.adv$. Functionality F' is indifferentiable from F if there exists a simulator $\mathcal{S}$ such that no distinguisher can determine when it has access to oracles $F.hon$ and $F.adv$ or to $F'.hon$ and $\mathcal{S}^{F'.adv}$. Reset indifferentiability asks that no distinguisher can differentiate those two sets of oracles, but when the distinguisher can reset the simulator to its initial state at arbitrary times. Randomized simulators use freshly-chosen coins after each reset.

The inability to distinguish when resets are allowed enables proving a composition theorem for multi-stage games because the resets allow one to restart the simulator for each stage. We show that establishing (regular) indifferentiability via a stateless simulator immediately yields a proof of reset indifferentiability. However, this is of limited applicability since most indifferentiability results, including all those known for hash constructions, are not stateless. Moreover, our results thus far already rule out typical hash constructions from being reset indifferentiable since they have online computability. Still, that leaves open if other constructions perform better.

We show that no practical domain extension transforms can be reset indifferentiable from a RO. The result is quite general, ruling out all known single-pass constructions. We leave open the problem of proving the existence (or inexistence) of a domain extender, even an impractical one (i.e., one that makes two or more passes over the message), that is reset indifferentiable.

DIRECT PROOFS. Having lost composition as a general way to transport ROM proofs of security for multi-stage games to the setting where one uses a hash constructed from an ideal primitive, we take up consideration of a specific security goal from public-key encryption. We prove a theorem establishing the chosen-distribution attack (CDA) security for a number of related, ROM-secure, PKE schemes when these are used with any indifferentiable hash function built according to a design paradigm introduced by Dodis, Ristenpart and Shrimpton [35]. The CDA security notion [11] captures message privacy of a PKE scheme when messages and randomness are (jointly) unpredictable, but otherwise adversarially controlled. In particular, this notion is the goal in the context of deterministic PKE [10, 14, 24], hedged PKE (which provides message privacy even in the face of poor randomness) [11, 55], and efficiently searchable encryption (an extension of deterministic PKE) [10]. As expected, this direct proof of security is complex because we have to work directly in the model of the ideal primitive underlying the hash function. This case study shows that direct security results are possible, restoring confidence that in some multi-stage settings security holds with proposed indifferentiable hash constructions.

FURTHER ISSUES WITH COMPOSITION. We have focused on hash functions and the ROM, but the limitation of composition to single-stage games affects any use of indifferentiability. Moreover, the limitations extend to other composition frameworks such as universal composability (UC) [27], which have mechanics similar to indifferentiability. We discuss UC in Section 7.

Finally, in the course of understanding the hash-based auditing counter-example, we uncovered other potentially subtle ways in which composition may fail to help one establish security. We catalogue these issues in Section 8.

DISCUSSION. We emphasize that we are *not* recommending that indifferentiability be dropped as a target of hash function design. The class of single-stage games encompasses many important security goals, and even after our results composition remains, when it applies, an elegant way to analyze security. Rather, the message here is that one must be very careful when using composition.

2 Preliminaries

NOTATION. When $\mathcal{X}$ is a non-empty finite set, we write $x \leftarrow\!\!\$\ \mathcal{X}$ to mean that a value is sampled uniformly at random from $\mathcal{X}$ and assigned to x. We overload this notation to extend to probabilistic algorithms, so that $x \leftarrow\!\!\$\ \mathcal{A}$ means that x is assigned a value according to the distribution induced by algorithm $\mathcal{A}$. (More intuitively, $\mathcal{A}$ is run and x is assigned its output.) When X and Y are strings, we write $X \parallel Y$ to mean the string created by appending Y to X. When $n > 0$ is an integer we write $\{0,1\}^n$ for the set of all n-bit strings, and $(\{0,1\}^n)^+$ for the set of all strings M such that $|M|$ is a positive multiple of n.

A CODE-BASED GAMES FRAMEWORK. We formalize a version of the code-based games framework of Bellare and Rogaway [20] for representing security experiments, indifferentiability, and the like. We find code-based games useful for formalizing security definitions, in particular, because they allow us to specify execution semantics (i.e. what runs what, and in what order). We use procedures, variables, and typical programming statements (operators, loops, procedure calls, etc.). Types are understood from context, and the names of syntactic objects must be distinct (e.g., a variable and procedure cannot have the same name). Variables are implicitly set initially to default values, i.e. integer variables are set to 0, arrays are everywhere $\perp$, etc.

A *procedure* is a sequence of statements together with zero or more inputs (variables) and zero or more outputs (variables). An *unspecified procedure* is one whose pseudocode, inputs, and outputs are understood from context. An *adversary* is an example of an unspecifired procedure. Calling a procedure P means providing it with inputs and running its sequence of statements. During its execution P may itself call other procedures. Say that the code of P expects to be able to call k distinct procedures. We will write $P^{Q_1,Q_2,\ldots,Q_k}$ to denote that these calls are handled by $Q_1, Q_2, \ldots, Q_k$ and implicitly assume (for all $i \in [k]$) that there are no syntactic mismatches between the calls that P makes to Q_i and the inputs of Q_i, as well as between the return values of Q_i and the return values expected by P. We stress that P does not call Q_i by name, but rather calls to a procedure that is instantiated by Q_i.

We assume that all procedures eventually halt, returning their outputs, at which point execution returns to the calling procedure. Procedures P_1 and P_2 are said to *export the same interface* if their inputs and outputs agree in number and type. This will typically be clear from context.

Variables are by default local, meaning they can only be used within a single procedure. The variables used within a procedure maintain their state between calls. A *collection of procedures* is a set of one or more procedures that may instead share their variables. We denote a collection of procedures by using a common prefix ending with a period, e.g. $(P.x, P.y, \ldots)$ and we use the common prefix P to refer to the collection. We will sometimes refer to the unique suffixes, e.g. x, y, as interfaces of P.

A *main procedure* is a distinguished procedure that takes no inputs and has some output. We mark it by **main**. No procedure may call **main**, and **main** can access all variables of other specified procedures. (But not other unspecified procedures.)

FUNCTIONALITIES AND GAMES. Collections of procedures will sometimes implement particular abstract functionalities, for example that of some idealized primitive (e.g. a random oracle). A functionality is a collection $F = (F.hon, F.adv)$; the names of these interfaces, hon and adv are suggestive as we will see in a moment. When games and adversaries are given access to a functionality a model of computation is induced, for example when the functionality is that of a random oracle, we have the random-oracle model. Thus one can think of functionalities and models somewhat interchangeably. For this work we specifically designate two models. First $\mathsf{RO} = (\mathsf{RO}.hon, \mathsf{RO}.adv)$, shown on the left-hand side of Figure 1, implements a random oracle (with two interfaces) and will give rise to the random-oracle model. Second, let $P = (P.hon, P.adv)$ implement some (ideal) primitive that underlies some understood construction H. Then $\mathsf{IP} = (\mathsf{IP}.hon, \mathsf{IP}.adv)$ shown on the right-side of Figure 1 gives rise to an (ideal) primitive model. For notational compactness, each time we use IP we will specify a construction H and a primitive P and assume these are the ones referred to in Figure 1.

For any two functionalities F_1, F_2, we denote by (F_1, F_2) the functionality that exposes a procedure that

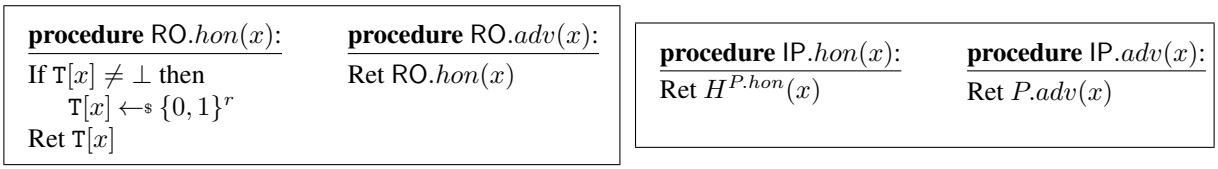

Figure 1: Procedures implementing the functionality of the random oracle model (ROM) **(left)** and the ideal primitive model (IPM) **(right)**. The number r is set as appropriate for a given context.

allows querying $(F_1.hon, F_2.hon)$ and a procedure that gives access to $(F_1.adv, F_2.adv)$.

A *game* G consists of a single main procedure, denoted "**main** G", together with a set of zero or more other specified procedures. (See for example Figure 2.) A game can make use of a functionality F and a number of adversarial procedures $\mathcal{A}_1, \ldots, \mathcal{A}_m$ together referred to as the *adversary*. We denote this by $G^{F,\mathcal{A}_1,\ldots,\mathcal{A}_m}$. For any $F_1, \mathcal{A}_1, \ldots, \mathcal{A}_m$ and $F_1', \mathcal{A}_1', \ldots, \mathcal{A}_m'$ such that $F_1.hon, F_2.hon$ are interface compatible and $\mathcal{A}_i, \mathcal{A}_i'$ are interface compatible for $1 \le i \le m$, we can write $G^{F_1,\mathcal{A}_1,\ldots,\mathcal{A}_m}$ to mean running game G with one set of external procedures and $G^{F_2,\mathcal{A}_1',\ldots,\mathcal{A}_m'}$ to mean running the same game but now with the second set of external procedures. Running a game $G^{F,\mathcal{A}_1,\ldots,\mathcal{A}_m}$ means executing the sequence of statements of the game's **main** procedure and the output of G is the value returned by **main**. We denote by $G^{F,\mathcal{A}_1,\ldots,\mathcal{A}_m} \Rightarrow y$ the event that the game's output is y, taken over the probability space defined by the coins used to execute G and the coins used in each invocation of the procedures $F.hon, F.adv, \mathcal{A}_1, \ldots, \mathcal{A}_m$. Should G and the adversary not use $F.hon, F.adv$ then we instead write $G^{\mathcal{A}_1,\ldots,\mathcal{A}_m} \Rightarrow y$. As examples, games that do not use a functionality F are given in Figure 2 while games that do are given in Figures 3 and 5.

For any fixed functionality F and adversary $\mathcal{A}_1, \ldots, \mathcal{A}_m$, two games G and H are *equivalent* if

$$\Pr\left[\, G^{F,\mathcal{A}_1,\ldots,\mathcal{A}_m} \Rightarrow y \,\right] = \Pr\left[\, H^{F,\mathcal{A}_1,\ldots,\mathcal{A}_m} \Rightarrow y \,\right]$$

for all values y.

RESOURCES. For simplicity, we fix the convention that each statement of a procedure runs in unit time. The running time of a procedure, then, is the maximum number of statements executed, where the maximum is taken over all possible inputs and over all coins used by the procedure. The number of queries of a procedure is the maximum number of procedure calls it makes in one execution, again with the maximum taken over all possible inputs and all possible coins used by the procedure.

ENTROPY MEASURES. The min-entropy of a random variable X is $H_\infty(X) = -\log\left(\max_x \Pr[\, X = x \,]\right)$ measures the unpredictability of X, and $2^{-H_\infty(X)}$ is an upperbound on the probability that any adversary can guess the value of X. When X and Y are two (possibly correlated) random variables the *average min-entropy* of Dodis et al. [33]

$$\tilde{H}_\infty(X \mid Y) = -\log\left(\sum_y \left(\max_x \Pr[\, X = x \mid Y = y \,]\right) \Pr[\, Y = y \,]\right)$$

correspondingly measures the unpredictability of X given knowledge of Y, and $2^{-\tilde{H}_\infty(X \mid Y)}$ upper bounds the chance that computationally unbounded adversary can guess X after Y is revealed to it. Lemma 2.2 from [33], which relates average min-entropy and min-entropy, is particularly useful for our work.

Lemma 2.1 (Lemma 2.2 [33]) If Y can take on 2^r possible values, then $\tilde{H}_\infty(X \mid Y) \ge H_\infty(X) - r$.

3 Indifferentiability and Composition for Single-stage Games

We describe the indifferentiablity framework [51] using games, unlike prior treatments that used random systems [50, 51] or interactive Turing machines [31]. We feel that using explicit code-based games makes understanding the limitations of indifferentiability easier, because it will enable expressing these limitations

main Real $b' \leftarrow\!\!\$\ \mathcal{D}^{\mathsf{Func},\mathsf{Prim}}$ Ret b'	**procedure** $\mathsf{Func}(m)$: Ret $F_1.hon(m)$	**procedure** $\mathsf{Prim}(u)$: Ret $F_1.adv(u)$
main $\text{Ideal}_{\mathcal{S}}$ $b' \leftarrow\!\!\$\ \mathcal{D}^{\mathsf{Func},\mathsf{Prim}}$ Ret b'	**procedure** $\mathsf{Func}(m)$: Ret $F_2.hon(m)$	**procedure** $\mathsf{Prim}(u)$: Ret $\mathcal{S}^{F_2.adv}(u)$

Figure 2: The games that define indifferentiability. Adversary $\mathcal{D}$ and functionalities F_1, F_2 are unspecified. The simulator $\mathcal{S}$ is a parameter of the game.

as syntactic conditions on the class of games considered. In addition to defining indifferentiability, we will provide a concrete version of the composition theorem given in [51] and characterize its limitations.

INDIFFERENTIABILITY. Fix two functionalities F_1 and F_2. When thinking of indifferentiability from random oracles, for example, we use $F_1 = \mathsf{IP}$ (for some understood H, P) and $F_2 = \mathsf{RO}$. A distinguisher $\mathcal{D}$ is an adversary that outputs a bit. A simulator is a procedure, usually denoted $\mathcal{S}$. Figure 2 defines two games Real and Ideal. Fix some value y (e.g., $y = 1$). The indifferentiability advantage of $\mathcal{D}$ is defined as

$$\mathbf{Adv}^{\text{indiff}}_{F_1,F_2,\mathcal{S}}(\mathcal{D}) = \Pr\left[\,\text{Real}^{F_1,\mathcal{D}} \Rightarrow y\,\right] - \Pr\left[\,\text{Ideal}^{F_2,\mathcal{D}}_{\mathcal{S}} \Rightarrow y\,\right].$$

We use a concrete security approach, i.e. not providing a strict definition of achieving indifferentiability. However, informally we will say that a functionality F_1 is indifferentiable from a functionality F_2 if for any "reasonable" adversary $\mathcal{D}$ there exists an "efficient" simulator $\mathcal{S}$ such that $\mathbf{Adv}^{\text{indiff}}_{F_1,F_2,\mathcal{S}}(\mathcal{D})$ is "small". The meanings of "reasonable", "efficient", and "small" will be clear from context.

To get an asymptotic notion, we can assume an implicit security parameter k throughout, and then use the definition of [31]: F_1 is indifferentiable from F_2 if there exists a PT simulator $\mathcal{S}$ such that for any PT $\mathcal{D}$ it is the case that $\mathbf{Adv}^{\text{indiff}}_{F_1,F_2,\mathcal{S}}(\mathcal{D})$ is negligible in the security parameter. Note that in [51] a different quantifier ordering was used. It said that for all PT $\mathcal{D}$ there must exist a PT simulator $\mathcal{S}$ such that $\mathbf{Adv}^{\text{indiff}}_{F_1,F_2,\mathcal{S}}(\mathcal{D})$ is negligible in the security parameter. We refer to the [51] notion as weak indifferentiability and to the [31] notion as strong indifferentiability. We will focus on strong indifferentiability here since it implies weak.

TYPES OF GAMES. Before giving the composition theorem from [51] we first give definitions for various restrictions on games. Let $\mathcal{G}$ be the set of all games G, i.e. any that fits our definitions from the previous section. Then a game G is *functionality respecting* if the game's main and specified procedures do not call $F.adv$. This means in particular that only adversarial procedure(s) may call $F.adv$. Let $\mathcal{L}G \subset \mathcal{G}$ be the set of all functionality respecting games G.

We go on to partition $\mathcal{L}G$ into two parts. The first is all games $G \in \mathcal{L}G$ that use only a single adversarial procedure. We call these *single stage games*. Let $\mathcal{SG} \subset \mathcal{L}G$ be the set of all single stage games that are functionality respecting. The second part is all other games in $\mathcal{L}G$, meaning ones that use $m \geq 2$ adversarial procedures. Let $\mathcal{MG} \subset \mathcal{L}G$ be the set of all *multi-stage games* that are functionality respecting. Then $\mathcal{L}G = \mathcal{SG} \cup \mathcal{MG}$.

The set $\mathcal{SG}$ includes the games defining indifferentiability above, the classic notions of encryption security such as IND-CPA [38] (see Figure 8) or IND-CCA [52], unforgeability under chosen message attack UF-CMA [39], and many others. The set $\mathcal{MG}$ includes the games defining chosen distribution attack security for public-key encryption [11] (see Figure 5), non-malleability of hash functions [23], password-based key exchange [17], key-dependent message security [22], related-key attack security [2, 16], and others.

We point out that a game $G \in \mathcal{MG}$ could be equivalent to a game $H \in \mathcal{SG}$. (The definition of equivalent is given in the previous section.) For example, common formulations of encryption indistinguishability security [13] are written using several adversarial procedures that pass an arbitrary state string between theme. It is clear these can be equivalently formulated as a single stage game. For others, such as those listed above as examples of multi-stage games, there does not appear to be such a translation. We therefore say

that an m-stage game G is *stage minimal* if it is not equivalent to any game H with less than m adversarial procedures.

COMPOSITION. One goal of indifferentiability is to allow the security analysis of a cryptographic scheme when using one functionality to imply security holds when using another. This is enabled by the following, which is a concrete security version of the original composition theorem of Maurer, Renner, and Holenstein [51].

Theorem 3.1 Let $G \in \mathcal{SG}$. Let F_1, F_2 be two functionalities with compatible honest interfaces. Let $\mathcal{A}$ be an adversary with one oracle. Let $\mathcal{S}$ be a simulator that exports the same interface as $F_1.adv$. Then there exist adversary $\mathcal{B}$ and distinguisher $\mathcal{D}$ such that for all values y

$$\Pr\left[\, G^{F_1,\mathcal{A}} \Rightarrow y \,\right] \leq \Pr\left[\, G^{F_2,\mathcal{B}} \Rightarrow y \,\right] + \mathbf{Adv}^{\text{indiff}}_{F_1,F_2,\mathcal{S}}(\mathcal{D}) \ .$$

Moreover

$$t_{\mathcal{B}} \leq t_{\mathcal{A}} + q_{\mathcal{A}} \cdot t_{\mathcal{S}} \qquad q_{\mathcal{B}} \leq q_{\mathcal{A}} \cdot q_{\mathcal{S}} \qquad t_{\mathcal{D}} \leq t_G + q_{G,1} \cdot t_{\mathcal{A}} \qquad q_{\mathcal{D}} \leq q_{G,0} + q_{G,1} \cdot q_{\mathcal{A}}$$

where $t_{\mathcal{A}}, t_{\mathcal{B}}, t_{\mathcal{D}}$ are the maximum running times of $\mathcal{A}, \mathcal{B}, \mathcal{D}$; $q_{\mathcal{A}}, q_{\mathcal{B}}$ are the maximum number of queries made by $\mathcal{A}$ and $\mathcal{B}$ in a single execution; and $q_{G,0}, q_{G,1}$ are the maximum number of queries made by G to the honest interface and to the adversarial procedure. □

The proof of Theorem 3.1 is readily established by adapting the proof of [51, Th. 1]. We provide a proof here to help support our upcoming discussion.

Proof: Fix any value y. Let $F = (F.hon, F.adv)$ be some unspecified functionality that export the same interface as $(F_1.hon, F_1.adv)$. Let indifferentiability adversary $\mathcal{D}$ be defined as follows. Adversary $\mathcal{D}$ runs game G. Whenever G calls its honest interface, adversary $\mathcal{D}$ queries $F.hon$ and returns the result. Whenever G calls $\mathcal{A}$, adversary $\mathcal{D}$ runs $\mathcal{A}$ for G using $F.adv$ to answer any queries made by $\mathcal{A}$. Finally $\mathcal{D}$ outputs whatever G outputs. Then by construction $q_{\mathcal{D}} \leq q_{G,0} + q_{G,1} q_{\mathcal{A}}$; $t_{\mathcal{D}} \leq t_G + q_{G,1} t_{\mathcal{A}}$; and

$$\Pr\left[\, \text{Real}^{\mathcal{D}} \Rightarrow y \,\right] = \Pr\left[\, G^{F_1,\mathcal{A}} \Rightarrow y \,\right] \tag{1}$$

in the case that $F = F_1$. Now we define adversary $\mathcal{B}$ as follows. Adversary $\mathcal{B}$ runs $\mathcal{A}$. When $\mathcal{A}$ queries its oracle, adversary $\mathcal{B}$ runs $\mathcal{S}$ using its $F_2.adv$ oracle to answer any queries $\mathcal{S}$ makes. Adversary $\mathcal{B}$ outputs whatever $\mathcal{A}$ outputs. By construction, then, we have that $q_{\mathcal{B}} \leq q_{\mathcal{A}} \cdot q_{\mathcal{S}}$; $t_{\mathcal{B}} \leq t_{\mathcal{A}} + q_{\mathcal{A}} \cdot t_{\mathcal{S}}$; and

$$\Pr\left[\, \text{Ideal}^{\mathcal{D}}_{\mathcal{S}} \Rightarrow y \,\right] = \Pr\left[\, G^{F_2,\mathcal{A}^{\mathcal{S}}} \Rightarrow y \,\right] = \Pr\left[\, G^{F_2,\mathcal{B}} \Rightarrow y \,\right] \tag{2}$$

in the case that $F = F_2$. By substituting according to Equations 1 and 2 into the definition of indifferentiability advantage we derive the advantage relation of the theorem statement. ▮

INAPPLICABILITY TO MULTI-STAGE GAMES. The theorem above explicitly restricts attention to games that only use a single adversarial procedure. At first glance, this restriction may seem artificial. Suppose a game $G \in \mathcal{MG}$ expects access to adversarial procedures $\mathcal{A}_1, \ldots, \mathcal{A}_m$. now consider extending Theorem 3.1 to account for G. Recall that these adversarial procedures do not necessarily share state. In the proof, a key step is defining the adversary $\mathcal{B}$. Following that proof, for this generalization we could define adversarial procedures $\mathcal{B}_1, \ldots, \mathcal{B}_m$ by $\mathcal{B}_i = \mathcal{A}_i^{\mathcal{S}}$ for all i. One may think a proof has been arrived at. However $\mathcal{S}$ is only guaranteed to simulate properly when it maintains its state across all invocations throughout the course of the indifferentiability game. Technically, then, the problem is that the analogue of equation (2) for this proof attempt would fail:

$$\Pr\left[\, G^{F_2,\mathcal{B}_1,\ldots,\mathcal{B}_m} \Rightarrow y \,\right] = \Pr\left[\, G^{F_2,\mathcal{A}_1^{\mathcal{S}},\ldots,\mathcal{A}_m^{\mathcal{S}}} \Rightarrow y \,\right] \neq \Pr\left[\, \text{Ideal}^{\mathcal{D}}_{\mathcal{S}} \Rightarrow y \,\right] \ .$$

This is true regardless of how we define $\mathcal{D}$. In the next section, we provide a counterexample showing that there is no hope of a proof for this generalization.

DISCUSSION. The above makes explicit that existing indifferentiability-style composition only works for

main $\mathrm{CRP}^{F,\mathcal{A}_1,\mathcal{A}_2}_{p,n,s}$

$M \xleftarrow{\$} \{0,1\}^p$
$st \xleftarrow{\$} \mathcal{A}_1^{F.adv}(M)$
If $|st| > n$ then Ret false
$C \xleftarrow{\$} \{0,1\}^s$
$Z \xleftarrow{\$} \mathcal{A}_2^{F.adv}(st, C)$
Ret $\big(Z = F.hon(M \parallel C)\big)$

Figure 3: Game capturing our challenge-response hash function property.

security notions definable via single-stage games. (As mentioned above, a multi-stage game that is equivalent to a single-stage game can also be treated.) We note that some multi-stage games pass some small amount of state between adversarial procedures. See for example the hash auditing security property formalized in Figure 3. Here, however, the state is not arbitrary —its length is a fixed constant— and so this game is not equivalent to a single stage game.

We do note, however, that we may extend Theorem 3.1 to cover multi-stage games that directly share some limited amount of state, but an amount sufficient to enable composition. That is, the shared state must be large enough to transport the state of $\mathcal{S}$ between $\mathcal{B}_i$ calls (in addition to whatever other state an adversary might use). We do not know of any examples of such multi-stage games, and so do not spell out the details of such an extension.

OTHER SUBTLETIES OF COMPOSITION. There are other subtleties of composition that might lead to its inapplicability. We discuss these further in Section 8.

4 A Counterexample to Multi-stage Composition

In this section we define a simple hash function property that is met by a RO, but not met by a broad class of hash functions *proven* to be indifferentiable from a RO. Together these results give a counterexample disproving the desired generalization of Theorem 3.1 to multi-stage games.

HASH-BASED STORAGE AUDITING. The property we study, denoted CRP, is motivated by challenge-response auditing protocols for secure distributed storage [46]. Consider that a client wishes to store some data M on a remote server. It will later verify that M is in fact being stored by sending a random challenge C to the server, and then checking that the server's response matches the hash $H(M \parallel C)$. Intuitively, if H is a random oracle, there is no way for the server to "cheat": It must actually store M, or guess the challenge in advance, if it is to respond correctly. (Drawing the challenges from a sufficiently large space or repeating the protocol will make the chance that the server guesses the challenges arbitrarily small.) In particular, if the server stores some state st instead of M, and $|st| \ll |M|$, then we expect the server will fail to respond properly. The CRP experiment in Figure 3 captures a slightly simplified version of this example.

Informally, a CRP-secure hash function H should not admit the storage of a short string (much shorter than the file M) that later allows the server to answer auditing challenges C, except with negligible probability. This guarantees that a rational server interested in saving storage space but subject to auditing will not store some short digest in place of the file. For completeness, we show in Appendix A that CRP is not equivalent to the standard notions of hash function security.

The following theorem shows that, as expected, a random oracle possesses property CRP.

Theorem 4.1 Fix $p, n, s > 0$. Let $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ be an adversary that makes a total of q calls. Then

$$\Pr\left[\,\mathrm{CRP}^{\mathsf{RO},\mathcal{A}_1,\mathcal{A}_2}_{p,n,s} \Rightarrow \mathsf{true}\,\right] \leq \frac{q}{2^{p-n}} + \frac{1}{2^r} + \frac{q}{2^s}$$

where RO provides the functionality of a random oracle with range $\{0,1\}^r$. $\square$

Proof: Let Q be the event that the string $M \,\|\, C$ is queried by either $\mathcal{A}_1$ or $\mathcal{A}_2$ during execution of CRP. Then by conditioning on Q we have

$$\Pr\left[\,\mathrm{CRP}^{\mathsf{RO},\mathcal{A}_1,\mathcal{A}_2} \Rightarrow \mathsf{true}\,\right] \le \Pr\left[\,\mathsf{Q}\,\right] + \Pr\left[\,\mathrm{CRP}^{\mathsf{RO},\mathcal{A}_1,\mathcal{A}_2} \Rightarrow \mathsf{true} \mid \neg\mathsf{Q}\,\right] \le \Pr\left[\,\mathsf{Q}\,\right] + \frac{1}{2^r}$$

where the final inequality follows because if $M\|C$ is not queried, the output of $\mathsf{RO}.hon(M\|C)$ is uniformly random and independent of the rest of the experiment. Let Q_1 be the event that $\mathcal{A}_1$ queries $M \,\|\, C$, and Q_2 the event that $\mathcal{A}_2$ queries $M \,\|\, C$. Clearly $\Pr\left[\,\mathsf{Q}_1\,\right] \le q/2^s$ since C is not sampled until after $\mathcal{A}_1$ finishes its execution. We assume that $\mathcal{A}_1$ outputs a string st of length exactly n bits. This is without loss of generality, since any st of fewer bits can be padded, and if $|st| > n$ the game returns false. Thus $\mathcal{A}_1^{\mathsf{RO}.adv}(M)$ is a random variable taking on at most 2^n values, and by Lemma 2.1 we have $\tilde{H}_\infty(M \mid \mathcal{A}_1^{\mathsf{RO}.adv}(M)) \ge p - n$. It follows that $\Pr\left[\,\mathsf{Q}_2\,\right] \le q/2^{p-n}$, and a union bound yields the claim. ∎

ONLINE COMPUTABILITY AND CRP. We now define a structural property of hash functions, which we refer to as online computability. Consider a hash function $H^f\colon \{0,1\}^* \to \{0,1\}^r$ using some underlying primitive f. Then we say that H^f is (p,n,s)-*online computable* if for $p,n,s > 0$ there exist functions $H_1^f\colon \{0,1\}^p \to \{0,1\}^n$ and $H_2^f\colon \{0,1\}^n \times \{0,1\}^s \to \{0,1\}^r$ such that $H^f(M_1 \,\|\, M_2) = H_2^f(H_1^f(M_1), M_2))$ for any $(M_1, M_2) \in \{0,1\}^p \times \{0,1\}^s$. Moreover, we require that the time to compute H_1^f and H_2^f is within a small, absolute constant of the time to compute H^f. In words, the hash function H^f can be computed in two stages, processing M_1 and then M_2 sequentially.

We note that most iterative hash function constructions are online computable for a variety of values p, n, s. For example, the so-called NMAC construction from [31]. It uses two underlying ideal objects $f\colon \{0,1\}^{2n} \to \{0,1\}^n$ and $g\colon \{0,1\}^n \to \{0,1\}^n$. Let $f^+\colon (\{0,1\}^n)^+ \to \{0,1\}^n$ be the mapping defined as follows: on input $M = M_1 \,\|\, \ldots \,\|\, M_b$, for each $i \in \{1,\ldots,b\}$ compute $V_i = f(V_{i-1} \,\|\, M_i)$, where V_0 is some fixed n-bit string, and return V_b. Now, let $H^{f,g}(M) = g(f^+(M))$, where the domain is $(\{0,1\}^n)^+$. This construction is (p,n,s)-online computable for any p and s that are multiples of n. Say $p = in$ for any i and $s = n$. Then let $H_1^f(M_1) = f^+(M_1)$ and $H_2^f(V, M_2) = g(f(V, M_2))$. Similarly, many other iterative constructions are online computable for such parameters, for example EMD [18], MDP [41], the Chop and so-called HMAC constructions [31], and numerous SHA-3 candidates.

It is clear to see that any (p,n,s)-online computable hash function *cannot* be CRP for those same parameters. For the NMAC example above, let $\mathcal{A}_1$ output $st = H_1^f(M) = f^+(M)$. Let $\mathcal{A}_2$ output $H_2(st, C) = g(f(st, C))$. The adversary wins with probability 1.

SAFESTORE AND STORAGE AUDITING IN PRACTICE. The SafeStore protocol used exactly the opposite ordering of N and M, specifying that audit responses be computed by $H^f(N \,\|\, M)$. This construction does indeed have CRP (though one cannot use composition to establish it). The point is that indifferentiability appears to imply that $N \,\|\, M$ and $M \,\|\, N$ are equivalently secure. Given the widespread use of hash functions as random oracles in practice (implicitly or explicitly), this shows that one must be careful to analyze security starting (at least) with the setting of the ideal primitive and applying composition only when it applies.

5 Indifferentiability Fails for Many Multi-stage Games

In the last section we saw how indifferentiability-based composition fails for a particular game, this being the CRP game. Here we extend that negative result to show how indifferentiability-based composition fails for many multi-stage games, including ones covering security of password-based key exchange, deterministic public-key encryption, non-malleability of hash functions, and more. To do so, we give a general method to show that indifferentiability does not imply security for games $G \in \mathcal{MG}$.

Our approach will be to show that one can augment any ideal primitive to include a *storage interface*. This will simply take (key,value) pairs from the adversary and allow retrieving values by looking up a key.

This augmentation does not affect any existing indifferentiability results involving the primitive — as we show below, a simulator for the original ideal primitive is easily converted to a simulator for the augmented primitive. Finally, we will show how cryptosystems cannot meet some multi-stage notions of security in the augmented primitive model.

Formally, let F_1 be a functionality. Let St be the procedure that exposes a hash table $\mathtt{T}$. That is, on input a pair of strings (X, Y), it sets $\mathtt{T}[X] \leftarrow Y$ and returns nothing. On input a string $(X, \perp)$ it outputs $\mathtt{T}[X]$, which is $\perp$ if $\mathtt{T}[X]$ has yet to be set to another value. Then the storage-augmented functionality $F_1^* = (F_1.hon, F_1^*.adv)$ has the same honest interface as F_1 but $F_1^*.adv$ exposes both $F_1.adv$ and St. That is, $F_1^*.adv = (F_1.adv, St)$.

The following theorem states that if F_1 is indifferentiable from some functionality F_2, then F_1^* is also indifferentiable from F_2. Its proof is straightforward and omitted.

Theorem 5.1 Let F_1, F_2 be functionalities and F_1^* be the storage-augmented version of F_1. Let $\mathcal{S}_{\mathcal{B}}$ be a simulator. Then there exists a simulator $\mathcal{S}_{\mathcal{A}}$ such that for all distinguishers $\mathcal{A}$ there exists a distinguisher $\mathcal{B}$ such that

$$\mathbf{Adv}^{\text{indiff}}_{F_1^*, F_2, \mathcal{S}_{\mathcal{A}}}(\mathcal{A}) = \mathbf{Adv}^{\text{indiff}}_{F_1, F_2, \mathcal{S}_{\mathcal{B}}}(\mathcal{B})$$

$\mathcal{B}$ runs in time that of $\mathcal{A}$ and uses the same number of queries; $\mathcal{S}_{\mathcal{A}}$ runs in time that of $\mathcal{S}_{\mathcal{B}}$ plus a small constant and uses the same number of queries. □

What Theorem 5.1 shows is that, as far as indifferentiability is concerned, it does not matter if some portion of the distinguisher's state is exported to an oracle. The intuition behind this result is straightforward: distinguishers in indifferentiability maintain state throughout the experiment and so it hardly matters whether one stores its state in an oracle or locally. But the ability to store data in an oracle obviates security for many multi-stage games. Here are some examples of cryptographic security goals that are not achievable in a storage-augmented primitive model.

Example 1: CDA security for public-key encryption. Public-key encryption (PKE) and the chosen-distribution attack (CDA) security goal are defined in Section 9. CDA generalizes deterministic PKE security notions [10, 14, 24], and CDA-secure PKE is also used in the context of efficient search over encrypted data [10] and defense-in-depth against randomness failures [11, 55]. It is easy to see that if one is working in the F_1^* model, this being a storage-augmented primitive model, then the security notion is unachievable. To attack any scheme, a first-stage adversary $\mathcal{A}_1$ picks (m_0, m_1, r) uniformly, and queries $St(0, (m_0, m_1, r))$. The second-stage adversary $\mathcal{A}_2$ queries $St(0, \perp)$ to retrieve (m_0, m_1, r), encrypts both messages under r, compares the results with the challenge ciphertext, and outputs the appropriate bit. This adversary wins with probability one.

We point out that the adaptive version of CDA security as defined in [11] is our first example of a multi-stage game for m any polynomial. This is because it allows an attacker to adaptively query its oracle with a message sampling algorithm q times. Each invocation uses fresh, independent coins and has access to the RO and so we end up in a setting where $m = q$.

Example 2: Nonmalleable hashing. Boldyreva et al. [23] give a notion of nonmalleability for hash functions. At a high level it requires that no efficient adversary can find a distribution $\mathcal{X}$ such that for a message x sampled from $\mathcal{X}$, the adversary, given $H(x)$, can find another message x^* and range point y^* such that $y^* = H(x^*)$ and the tuple $x, x^*, H(x), H(x^*)$ satisfies some non-trivial relationship. For example, the relationship could be that x is a proper prefix of x^*, this relationship corresponding to a length-extension attack. It is crucial that the adversary does not know the coins used to sample x when it attempts to to find a winning x^*, y^*. In a primitive model we should give $\mathcal{X}$ access to the adversarial interface of the primitive. This models that messages can depend on the hash function description. But in the storage-augmented primitive model, the adversary could give a $\mathcal{X}$ that queries St to save the message x and later retrieve it to trivially win. This means that, for example, indifferentiability is not a sufficient property to prove resistance to a natural

formulation of length extension attack resistance. This is especially troubling because provable resistance to length extension attacks was a primary motivation for building indifferentiable hash constructions [31].

Example 3: Password-based cryptography. To model arbitrary password selection, one uses an adversarial procedure together with some requirement on the unpredictability of its outputs. Moreover, for full generality, one should provide this password sampler with access to the ideal functionality used. This allows consideration of passwords that are computed as a function of a hash function, a situation occurring when, for example, users hash a large file to derive a password or generate a random password using a system random number generator (which itself uses the hash function). For example, the password-based authenticated key exchange security notion of Bellare et al. [17] explicitly uses such a functionality-dependent password sampler. As in the nonmalleable hashing example, it is clear that security cannot be achieved in a storage-augmented primitive model for such password-based cryptography security notions.

Example 4: Key-dependent message security. In a key-dependent message (KDM) attack [22], the adversary obtains encryptions of messages that are chosen via some adversarially chosen function whose input is the secret key used by encryption. The first constructions, due to [22], are given in the ROM — the message-derivation function is assumed to have access to the RO. Of course it is easy to see that no scheme is secure in the storage-augmented primitive model because the message function could just store the key (to which it has access) and the distinguishing stage could then retrieve the secret key. While recent results on KDM secure primitives in the standard model [1, 6, 9, 25, 26, 42] are not directly affected by this problem, the most efficient constructions to date enjoy analyses only in an idealized model [8, 15, 22, 40]. Here indifferentiability doesn't apply, and whether there exist structure-abusing attacks remains an open question.

Example 5: Related-key attacks. Bellare and Kohno first formalized related-key attacks (RKAs) against block ciphers [16]. In general, an RKA occurs when an adversary can obtain outputs of a cryptographic construction used with multiple, related secret keys. The adversary chooses the relation, and this relation should have access to any ideal primitive used by the construction [2]. For primitives like block ciphers, one does not typically prove RKA resistance but rather establishes it via cryptanalytic analysis. However, for other primitives like encryption the most practical RKA constructions only have analyses in the ROM [12]. As with KDM security, it is easy to see that in the storage-augmented primitive model security is obviated. Again, it is unclear if structure abusing attacks exist against typical constructions.

DISCUSSION. The negative results presented in this section rely on augmenting primitives to incorporate a storage procedure. Of course in the context of hash function design, no one would consider using such a primitive (nor would there necessarily be any way to instantiate one!). Rather these results are used to show that indifferentiability cannot imply security in the context of the multi-stage games considered. For these examples, then, we do not know whether security holds (or not) when using hash constructions built from more typical underlying primitives (i.e., ones that are not storage augmented). It may be that clever structure abusing attacks exist. We will rule out such attacks in one setting in Section 9. This leaves the other settings as important areas for future analysis.

The notions above are only multi-stage because all adversarial algorithms are allowed access to the random oracle or underlying ideal primitive. We can circumvent the limitations of indifferentiability by targeting weaker security notions for which only one stage is given access to the RO. With CDA, for example, one could choose a weaker security setting in which the message samplers are not hash dependent. However, it is unlikely that such weakenings are sufficient in many applications.

Another point is that avoiding ideal primitive models would obviate these concerns, since one wouldn't be relying on a random oracle or ideal cipher in the first place. Of course, for many constructions whose security is meaured by multi-stage games the ROM or ICM remains the only ones in which proofs are known or (in some cases) appear possible [32, 45].

The examples of multi-stage security games given above are not meant to be exhaustive. We expect there are many other settings in which multi-stage games are the requisite security target.

6 Indifferentiability with Simulator Resets

We propose a strengthening of indifferentiability that supports composition for both single-stage and multi-stage games. The counter-example of Section 4 indicates that typical indifferentiable hash constructions cannot enjoy such a notion. Indeed, no online computable hash function can meet a strengthening whose associated composition theorem covers the CRP game. Nevertheless, we may hope to design new hash functions that do meet our stronger notion. As we'll see, though, there doesn't seem to be much hope for (efficient) hash domain extensions that support composition for multi-stage games.

RESET INDIFFERENTIABILITY. We define a version of indifferentiability that requires simulators to function even under resets. For any simulator $\mathcal{S}$ we define the procedure pair $\widehat{\mathcal{S}} = (\widehat{\mathcal{S}}.\mathcal{S}, \widehat{\mathcal{S}}.Rst)$. The former procedure is simply a renaming of $\mathcal{S}$. The latter procedure takes no input and when run reinitializes all of $\widehat{\mathcal{S}}.\mathcal{S}$'s internal variables to their initial values. Likewise, let $F = (F.hon, F.adv)$ be any functionality. Let functionality $\overrightarrow{F} = (\overrightarrow{F}.hon, \overrightarrow{F}.adv) = (F.hon, (F.adv, nop))$ where the procedure pair $\overrightarrow{F}.adv = (F.adv, nop)$ includes a procedure nop that takes no input and does nothing. Let F_1 and F_2 be functionalities. Let $\mathcal{D}$ be an adversary that outputs a bit (the distinguisher). Let $\mathcal{S}$ be a simulator. Then we define the reset indifferentiability advantage of $\mathcal{D}$ as

$$\mathbf{Adv}^{\text{reset-indiff}}_{F_1,F_2,\mathcal{S}}(\mathcal{D}) = \Pr\left[\,\text{Real}^{\overrightarrow{F}_1,\mathcal{D}} \Rightarrow y\,\right] - \Pr\left[\,\text{Ideal}^{F_2,\mathcal{D}}_{\widehat{\mathcal{S}}} \Rightarrow y\,\right].$$

For consistency with our definition of the games Real and Ideal (Figure 2), we implicitly assume there is some distinguished symbol that, when received as input by the procedure Prim, causes the execution of nop or $\widehat{\mathcal{S}}.Rst$, respectively.

We have the following composition theorem.

Theorem 6.1 Let $G \in \mathcal{L}G$. Let F_1 and F_2 be functionalities. Let $\mathcal{A}_1, \ldots, \mathcal{A}_m$ be an adversary and let $\mathcal{S}^{F_2.adv}$ be a simulator that exports the same interface as $F_1.adv$. Then there exist an adversary $\mathcal{B}_1, \ldots, \mathcal{B}_m$ and distinguisher $\mathcal{D}$ such that for all values y

$$\Pr\left[\,G^{F_1,\mathcal{A}_1,\ldots,\mathcal{A}_m} \Rightarrow y\,\right] \le \Pr\left[\,G^{F_2,\mathcal{B}_1,\ldots,\mathcal{B}_m} \Rightarrow y\,\right] + \mathbf{Adv}^{\text{reset-indiff}}_{F_1,F_2,\mathcal{S}}(\mathcal{D})\,.$$

Moreover

$$t_{\mathcal{B}_i} \le t_{\mathcal{A}_i} + q_{\mathcal{A}_i} t_{\mathcal{S}} \qquad q_{\mathcal{B}_i} \le q_{\mathcal{A}_i} \cdot q_{\mathcal{S}} \qquad t_{\mathcal{D}} \le m + t_G + \sum_{i=1}^{m} q_{G,i} \cdot t_{\mathcal{A}_i} \qquad q_{\mathcal{D}} \le q_{G,0} + \sum_{i=1}^{m} q_{G,i} \cdot q_{\mathcal{A}_i}$$

where $t_{\mathcal{A}}, t_{\mathcal{B}}, t_{\mathcal{D}}$ are the maximum running times of $\mathcal{A}, \mathcal{B}, \mathcal{D}$; $q_{\mathcal{A}}, q_{\mathcal{B}}$ are the maximum number of queries made by $\mathcal{A}$ and $\mathcal{B}$ in a single execution; and $q_{G,0}, q_{G,i}$ are the maximum number of queries made by G to the honest interface and the i^{th} adversarial procedure (respectively). □

The proof of the above is readily established by adapting the proof of Theorem 3.1. For $1 \le i \le m$, let $\mathcal{B}_i^{F_2.adv} = \mathcal{A}_i^{\mathcal{S}^{F_2.adv}}$. This means in particular that a separate instance of $\mathcal{S}$ is used in each procedure $\mathcal{B}_i$. Then define the distinguisher $\mathcal{D}$, for any compatible functionality $F = (F.hon, F.adv)$, by modifying $\mathcal{D}^{F.hon,F.adv} = G^{F,\mathcal{A}_1,\ldots,\mathcal{A}_m}$ so that a reset call immediately precedes each $\mathcal{A}_i$ call.

We point out that reset indifferentiability is not only sufficient, but necessary, for secure general composition for games in $\mathcal{G}$. Following the reasoning given in [51] for indifferentiability, we note that if a construction is not reset indifferentiable then there exists a multi-stage game for which the construction does not yield security.

Reset indifferentiability can be achieved when one establishes (conventional) indifferentiability using a stateless simulator. This is because it is clear resetting such a simulator does not affect its subsequent behavior. However most existing indifferentiability results use stateful simulators.

PRACTICAL HASH CONSTRUCTIONS ARE NOT RESET INDIFFERENTIABLE. As mentioned above, online computable hash functions cannot be reset indifferentiable. This is because the composition theorem would then imply such a hash function met the CRP property and the results of Section 4 rule this out. But

some efficient hash constructions do meet the CRP property, and so the question remains if any efficient construction meets reset indifferentiability. We rule out the existence of single-pass hash domain extenders that are reset indifferentiable.

Fix some $p, n, s, r > 0$ such that $p > n$ and let $N = p + s$. Let P be an arbitrary ideal primitive. We restrict our attention to domain-extension constructions $H^f\colon \{0,1\}^N \to \{0,1\}^r$ that can be written as $H^P(\langle M_1, M_2\rangle) = H_2^P(H_1^P(M_1) \,\|\, M_2)$ for any $(M_1, M_2) \in \{0,1\}^p \times \{0,1\}^s$. Here $\langle M_1, M_2\rangle$ represents a unique encoding of M_1, M_2 into an N-bit string; $H_1\colon \{0,1\}^p \to \{0,1\}^n$; and $H_2\colon \{0,1\}^n \times \{0,1\}^s \to \{0,1\}^r$. Importantly, that $p > n$ means that H_1 is compressing. We require that the time to compute one each of the encoding, H_1, and H_2 is within a small, absolute constant of the time to compute H^P. As concrete examples, all online computable functions are trivially included by setting $\langle M_1, M_2\rangle = M_1 \,\|\, M_2$. But the flexibility endowed by the arbitrary encoding also means we encompass a wider range of H that do not allow online computing. For example, any single pass hash function that can be written in the form above. On the other hand, constructions such as the zipper hash [48] (which makes two passes over a message) are not considered.

The following theorem below shows that no construction fitting the form above is reset indifferentiable, no matter what underlying primitive P is used.

Theorem 6.2 Let integers p, n, s, r, N, functionality P, and construction H^P be as just described. Let functionality RO implement a random oracle with range $\{0,1\}^r$. There exists a reset-indifferentiability adversary $\mathcal{D}$ such that for all simulators $\mathcal{S}$ asking at most q queries,

$$\mathbf{Adv}^{\text{reset-indiff}}_{\mathsf{IP},\mathsf{RO},\mathcal{S}}(\mathcal{D}) \geq 1 - \left(\frac{q}{2^s} + \frac{q}{2^{p-n}} + \frac{1}{2^r}\right). \ \square$$

The general nature of Theorem 6.2 may make it somewhat hard to appreciate, so we give the following illustrative corollary. Consider the case of a $2n$-bit to n-bit ideal compression function P, and a "first" section H_1 that outputs n-bit strings. (Think, for example, of iterative constructions such as EMD, MDP, and gf^+.) Then when H^P extends the domain of P to at least $3n$ bits, we have the following.

Corollary 6.3 Let $s = r = n$, $\ell = s + r = 2n$, $N \geq 3n$, and let H^P be as in Theorem 6.2. Then for any $\mathcal{S}$ making at most q calls, there exists an $\mathcal{D}$ such that

$$\mathbf{Adv}^{\text{reset-indiff}}_{\mathsf{IP},\mathsf{RO},\mathcal{S}}(\mathcal{D}) \geq 1 - \frac{2q+1}{2^n}. \ \square$$

Proof of Theorem 6.2: Adversary $\mathcal{D}$ begins by selecting $M_1 \xleftarrow{\$} \{0,1\}^p$ and then computes $v \leftarrow H_1(M_1)$ by making the necessary calls to its right procedure (either the $P.adv$ procedure of $\overrightarrow{P}.adv$, or $\widehat{\mathcal{S}}.\mathcal{S}$ procedure of $\widehat{\mathcal{S}}$). It then issues a reset call to its right procedure (either nop or $\widehat{\mathcal{S}}.Rst$). After the reset, it samples $M_2 \xleftarrow{\$} \{0,1\}^s$. Using v and M_2, adversary $\mathcal{D}$ proceeds to compute $Z' \leftarrow H_2(v \,\|\, M_2)$ by making the necessary calls to its right procedure. Finally it calls its left procedure on $\langle M_1, M_2\rangle$, receiving Z in return. If $Z' = Z$, then $\mathcal{D}$ returns 1, otherwise it returns 0. We note immediately that in the case that $\mathcal{D}$'s procedure calls are serviced by $\overrightarrow{\mathsf{IP}}.hon = H^{P.hon}$ and $\overrightarrow{\mathsf{IP}}.adv = (P.adv, nop)$, it will always be the case that $\mathcal{D}$ returns 1. Thus $\Pr\left[\,\text{Real}^{\overrightarrow{\mathsf{IP}},\mathcal{D}} \Rightarrow 1\,\right] = 1$.

Now consider the case that $\mathsf{RO}.hon$ and $\widehat{\mathcal{S}}^{\mathsf{RO}.adv}$ service $\mathcal{D}$'s calls. Unless $\widehat{\mathcal{S}}$ calls $\mathsf{RO}.adv(\langle M_1, M_2\rangle)$ during its execution (before or after the reset), the value $Z = \mathsf{RO}.adv(\langle M_1, M_2\rangle)$ is independent of all quantities determined prior to $\mathcal{D}$'s final call, and so the chance that $Z = Z'$ is at most $1/2^r$. Letting bad indicate the event that $\widehat{\mathcal{S}}$ queries $\mathsf{RO}.adv(\langle M_1, M_2\rangle)$, we have

$$\Pr\left[\,\text{Ideal}^{\mathsf{RO},\mathcal{D}}_{\widehat{\mathcal{S}}} \Rightarrow 1\,\right] \leq \Pr\left[\,\mathsf{bad}\,\right] + \Pr\left[\,\text{Ideal}^{\mathsf{RO},\mathcal{D}}_{\widehat{\mathcal{S}}} \Rightarrow 1 \mid \neg\mathsf{bad}\,\right] \leq \Pr\left[\,\mathsf{bad}\,\right] + \frac{1}{2^r}$$

and so it remains for us to bound $\Pr[\,\mathsf{bad}\,]$. Let bad_1 indicate the event that $\widehat{\mathcal{S}}$ calls $\mathsf{RO}.adv(\langle M_1, M_2\rangle)$ before the reset. Since M_2 is not sampled until after the reset, is easy to see that $\Pr[\,\mathsf{bad}_1\,] \leq q/2^s$. We

note that one implication of bad_1 not occurring is that the simulator cannot somehow construct its responses to $\mathcal{D}$ pre-reset so that v could reveal Z to the simulator post-reset. (Intuitively, passing Z across the reset might allow $\widehat{\mathcal{S}}$ to respond properly post-reset without querying $\mathsf{RO}.adv(\langle M_1 \parallel M_2 \rangle$.) We will now argue that the simulator likewise cannot use v to leak M_1 across the reset.

Recall that $H_1(M_1)$ is compressing p bits to n bits, and consider the following experiment. Let $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ be a pair of adversarial procedures. A string $M_1 \leftarrow\!\!\$\ \{0,1\}^p$ is sampled and given to $\mathcal{A}_1$ as input. Procedure $\mathcal{A}_1$ ends its execution by outputting a string $v \in \{0,1\}^n$. Procedure $\mathcal{A}_2$ is then run on input v, outputting a string $M \in \{0,1\}^p$, and $\mathcal{A}$ is said to win if $M = M_1$. Without loss, we assume that $\mathcal{A}_1$ has hardcoded into its description the coins that maximize the probability of A winning this game. Thus $\mathcal{A}_1$ is deterministic, and

$$\Pr\left[\, M_1 \leftarrow\!\!\$\ \{0,1\}^p \,;\, v \leftarrow \mathcal{A}_1(M_1) \,;\, M \leftarrow\!\!\$\ \mathcal{A}_2(v) : M = M_1 \,\right]$$

is the probability we want to bound. Since $\mathcal{A}_1(M_1)$ is a random variable, the chance that $\mathcal{A}_2$ correctly guesses M_1 is at most $2^{-\tilde{H}_\infty(M_1 \mid \mathcal{A}_1(M_1))}$; Lemma 2.1 gives us that $\tilde{H}_\infty(M_1 \mid \mathcal{A}_1(M_1)) \geq H_\infty(M_1) - n = p - n$. Thus the chance that $\mathcal{A}$ wins the game we've described is at most $2^{-(p-n)}$. Translating this back to our setting, we can think of $\mathcal{A}_1$ and $\mathcal{A}_2$ as $\widehat{\mathcal{S}}$ before and after the reset (respectively). In fact $\widehat{\mathcal{S}}$ is more restricted, since we assumed that $\mathcal{A}_1$ used the best coins and could output an arbitrary string (without regard for observing any particular distribution). As M_2 is independent of M_1, and $\widehat{\mathcal{S}}$ makes at most q calls over its entire execution, we can conclude that that the chance that $\widehat{\mathcal{S}}$ calls $\mathsf{RO}.adv(\langle M_1, M_2 \rangle)$ after the reset is at most $q/2^{p-n}$. Collecting results, we have

$$\Pr\left[\ \mathrm{Ideal}^{\mathsf{RO},\mathcal{D}}_{\widehat{\mathcal{S}}} \Rightarrow 1\ \right] \leq \Pr\left[\, \mathsf{bad} \,\right] + \frac{1}{2^r} \leq \frac{q}{2^s} + \frac{q}{2^{p-n}} + \frac{1}{2^r}$$

proving our claim. ∎

7 Universal Composability and Multi-stage Games

The paper thus far has focused on the indifferentiability framework. What about other composition frameworks? Indifferentiability was inspired by universal composability [27] and reactive simulatability [53]. All these frameworks share similar mechanics, including the use of simulators, ideal functionalities, and general composition theorems. We here give a high-level discussion of how one translates our multi-stage counter example to the setting of universal composability (UC). The limitations seem to extend to other settings as well, such as the JUC [29] framework. We will then discuss the implications.

The stated goal of UC is to prove that a cryptographic protocol is secure regardless of the environment it operates within [27]. The approach builds upon the simulation-based paradigm of secure multiparty computation [37]. One defines an ideal functionality that captures a perfectly secure realization of the protocol. Then one shows that a protocol UC-emulates this ideal functionality. That is, one shows, for any environment, that for any attacker against the real protocol there exists an equally successful attacker (the simulator) against the ideal functionality. Most often the analysis stops here, in the hope that the ideal functionality by its very nature provides the properties required by applications.

UNIVERSAL COMPOSABILITY FOR SINGLE-STAGE ENVIRONMENTS. UC is typically formalized using interactive Turing Machines (ITMs). We give a high level discussion of these, define UC execution via code-based games, and refer the reader to [27] for more details on the ITM-based formulation. The experiment defining UC is parameterized by three ITMs. The challenge protocol ρ is an ITM that may use as a subroutine some other ITM P. The adversary $\mathcal{A}$ is an arbitrary ITM. The environment $\mathcal{Z}$ is likewise an arbitrary ITM. The execution protocol defines how the experiment works, for full details see [27]. Briefly, the environment $\mathcal{Z}$ is first executed, and it may then run just a single adversarial ITM $\mathcal{A}$, and multiple copies of ρ. For our purposes we will focus on when there is just a single instance of ρ, the discussion lifts to

main $\text{EXEC}^{\mathcal{A}_1,\ldots,\mathcal{A}_m}_{\mathcal{Z},\rho,P}$	**procedure Prot**(X):	**procedure Adv**$_i(X)$:
$b' \leftarrow\!\!\$\ \mathcal{Z}^{\mathbf{Prot},\mathbf{Adv}_1,\ldots,\mathbf{Adv}_m}$	Ret $\rho^P(X)$	Ret $\mathcal{A}_i^{\mathbf{Prot},P}(X)$
Ret b'		
main $\text{IDEAL}^{\mathcal{S}_1,\ldots,\mathcal{S}_m}_{\mathcal{Z},\mathcal{F}}$	**procedure Prot**(X):	**procedure Adv**$_i(X)$:
$b' \leftarrow\!\!\$\ \mathcal{Z}^{\mathbf{Prot},\mathbf{Adv}_1,\ldots,\mathbf{Adv}_m}$	Ret $\mathcal{F}(X)$	Ret $\mathcal{S}_i^{\mathbf{Prot}}(X)$
Ret b'		

Figure 4: Games defining multistage UC (mUC) security for a single protocol instance. The top box is the execution of the real protocol in the P-hybrid model. The bottom box is the execution for the ideal functionality.

the more general setting in a natural way. We are in the P-hybrid model if ρ uses as subroutine an ideal functionality P. The adversary $\mathcal{A}$ is also assumed to have access to P. We write ρ^P to denote that ρ uses P as a subroutine. We can define this (special case) UC execution using game $\text{EXEC}^{\mathcal{A}}_{\mathcal{Z},\rho,P}$ with $m = 1$. See Figure 4.

We also consider execution with an idealized version of a protocol. Here the same environment $\mathcal{Z}$ interacts instead with an ideal functionality $\mathcal{F}$ and a simulator adversary $\mathcal{S}$. The adversary here also has access to $\mathcal{F}$. Note that we do not need P ($\mathcal{F}$ does not rely on an underlying primitive). This ideal execution is defined as the game $\text{IDEAL}^{\mathcal{S}}_{\mathcal{Z},\mathcal{F}}$ with $m = 1$ as shown in Figure 4.

We say that ρ UC-emulates $\mathcal{F}$ in the P-hybrid model if for any environment $\mathcal{Z}$ and adversary $\mathcal{A}$ there exists a simulator $\mathcal{S}$ such that

$$\Pr\left[\,\text{EXEC}^{\mathcal{A}}_{\mathcal{Z},\rho,P} \Rightarrow 1\,\right] - \Pr\left[\,\text{IDEAL}^{\mathcal{S}}_{\mathcal{Z},\mathcal{F}} \Rightarrow 1\,\right]$$

is bounded by a negligible function in the (implicit) security parameter.

Consider a protocol ITM Π that uses as a subroutine $\mathcal{F}$. Then the protocol ITM $\Pi^{\rho/\mathcal{F}}$ is the same as Π except that all calls to $\mathcal{F}$ are replaced by calls to ρ. (It is implicit in the notation that ρ has access to an underlying primitive P if it needs one.) The following is reproduced from [27] and is a special case of the main UC composition theorem.

Corollary 7.1 ([27, Cor. 15]) Let $\rho, \Pi, \mathcal{F}$ be ITMs such that ρ UC-emulates $\mathcal{F}$ in the P-model. Then $\Pi^{\rho/\mathcal{F}}$ UC-emulates Π.

UNIVERSAL COMPOSABILITY FOR MULTI-STAGE ENVIRONMENTS. The restriction in both the real and ideal executions to a *single* adversarial ITM (or, in our games parlance, a single stateful adversarial procedure) gives rise to the implicit restriction on composition that it only applies when one is working relative to a single-stage environment. Thus we expand the UC executions to allow multi-stage adversaries. We refer to this as multi-stage UC (mUC).

Now an adversary is a tuple of m ITMs $\mathcal{A}_1, \ldots, \mathcal{A}_m$. An environment is multi-stage if it expects access to $m > 1$ adversarial ITMs. Real execution proceeds by first having the multi-stage environment $\mathcal{Z}$ run. It can invoke the different ITMs $\mathcal{A}_1, \ldots, \mathcal{A}_m$ as it sees fit. Ideal execution is similarly generalized to work with m simulator adversaries $\mathcal{S}_1, \ldots, \mathcal{S}_m$. The adversarial ITMs in either world do not share state, and in particular all communication between the adversaries must be through $\mathcal{Z}$. This is similar to the UC variant defined in [3] for collusion-free multi-party computation. We say that a protocol ρ mUC-emulates an ideal functionality $\mathcal{F}$ in the P-hybrid model if

$$\Pr\left[\,\text{EXEC}^{\mathcal{A}_1,\ldots,\mathcal{A}_m}_{\mathcal{Z},\rho,P} \Rightarrow 1\,\right] - \Pr\left[\,\text{IDEAL}^{\mathcal{S}_1,\ldots,\mathcal{S}_m}_{\mathcal{Z},\mathcal{F}} \Rightarrow 1\,\right]$$

is negligible in some (implicit) security parameter. The execution protocol implied by our game-based formulation is just one choice; we leave to future work a fuller treatment of the various ways in which multi-stage games can be considered in UC-like settings

We are now in a position to transport the negative result from indifferentiability to mUC. That is, we show that even if one UC-emulates an ideal functionality, this does not mean one mUC-emulates the ideal functionality and, in particular, it may not be secure using the construction at all. Take $\rho = H$, $\mathcal{F} = \mathrm{RO}$, and P to be a (say) ideal compression function. Then we have the following easy proposition which follows.

Proposition 7.2 (informal) If H is indifferentiable from a random oracle when using primitive P, then protocol $\rho = H$ UC-emulates $\mathcal{F} = \mathrm{RO}$ in the P-hybrid model.

To see why, let $\mathcal{S}'$ be the simulator given by H being indifferentiable from a RO. Then the UC adversary $\mathcal{S}$ works by running the real world adversary $\mathcal{A}$ and answering its P queries by executing $\mathcal{S}'$. This can then be used to show that H UC-emulates a RO.

But then it is easy to see that, in fact, H does not mUC-emulate a RO. Why? Assume that H is (p, n, s)-online computable, fix the mUC environment $\mathcal{Z} = \mathrm{CRP}_{p,n,s}$. Then the adversary $\mathcal{A}_1, \mathcal{A}_2$ that works as described in Section 4 causes $\mathcal{Z}$ to output 1 (representing a win) with probability 1. However, we showed that no adversary can cause $\mathcal{Z}$ to output 1 with more than negligible probability when using a RO. Thus, H cannot mUC-emulate a RO.

DISCUSSION. The above is at a high level, and pinning down formal details would require an appropriately detailed formalization of UC. Such details are beyond the scope of this work. The take away, however, is that UC does not work for multi-stage games. More precisely: showing an ideal functionality UC-emulates a protocol does *not* imply that the protocol is safe to use when security is measured by a multi-stage game. This stands in contrast to the usual claims made about the generality of UC security.

Others have pointed out limitations of the UC framework. Canetti and Rabin [29] introduce a joint-state UC (JUC) model to expand UC composition to the setting in which different protocols use common components. They introduce a joint-state UC (JUC) model which also does not cover multi-stage games. Canetti et al. [28] point out that for applications like deniability, the environment must have access to setup functionalities. In normal (J)UC the environment does not, and so they introduce the generalized UC (GUC) framework. As neither JUC nor GUC allow environments that treat multi-stage games, the limitations we point out seem to apply. One caveat is that GUC does give the environment direct access to the underlying primitive P, which in our treatment means they consider games outside of $\mathcal{L}G$. An interesting open problem is characterizing the extent to which GUC composition covers multi-stage games.

Closer to our work is collusion-free protocols [3, 4, 44]. In this setting separate adversaries are assumed to only be able to communicate via some (semi-trusted) mediator and the task is to prove collusion-freeness of a protocol running on top of that channel. In [4] it is pointed out that UC cannot be used to treat collusion-freeness because it does not include disjoint adversaries. They give their own framework which includes multiple disjoint adversaries and where showing security requires simulators for each adversary. This framework is conceptually close to reset indifferentiability and may be a good starting point for future work on results that use composition to analyze multi-stage games. That said, we expect that our negative results about hash constructions from Section 6 would extend to this kind of setting.

8 Other Limitations of Composition

We discuss other ways that composition might not be applicable. While our discussion below focuses on the indifferentiability framework, many of these issues apply to reset indifferentiability and the UC framework.

CONCRETE SECURITY. Theorem 3.1 gives a concrete security version of the original MRH theorem. We felt that concrete security is important, because it is clear that composition increases the looseness of an ideal model reduction in several ways. One way loss occurs is due to the additive advantage overhead, which in the setting of hash constructions is often a birthday bound in the security parameter (the output size of the hash function). If, say, one achieves beyond-the-birthday bound security in the ROM for a cryptosystem, then using a typical indifferentiable hash construction will reduce the guarantee. (One could use the Maurer

and Tessaro [49] construction, which has less additive loss, to potentitally avoid this.) Another loss is in the running time of the adversary due to the simulator's overhead. Theorem 3.1 makes clear that if one uses an inefficient simulator, then after composition security may no longer hold in the case that computational assumptions are needed to establish security of a cryptosystem in the ROM.

GAMES THAT USE THE $F.adv$ INTERFACE. We only consider games $G \in \mathcal{L}G$, meaning that the game cannot access the adversarial interface of a functionality. In Section 9 we will give the PrA security notion, which does not abide by this convention. Both Theorem 3.1 and Theorem 6.1 therefore do not apply to the PrA game. Trying to generalize composition to cover such games proves unsuccessful: the simulator would have to also be run by the game, and this changes the game behavior.

As another example, consider a formalization of PrA that only has the adversary given direct access to $F.adv$. Instead, the security notion restricts attention only to adversaries that output a transcript of all queries and their responses to $F.adv$. This notion is equivalent to PrA, and may (at first glance) seem to avoid violating the convention. But still this should be considered a violation, because the correctness of the transcript relies on the interface. Another point is that the adversary $\mathcal{B}$ built by composition may fail to abide by the restriction.

GAMES MAKING NON-BLACK-BOX USE OF A FUNCTIONALITY. The composition theorem requires that the game makes essentially black-box use of the functionality. So if the game instead relies in some non-black-box way on the functionality, then it is unclear how composition can apply — the game will change between one functionality and the next. This means that Theorem 3.1 will apply, but that the right-hand side game $G^{F_2,\mathcal{B}}$ will not be the one appropriately measuring security for the F_2 model.

RESTRICTIONS ON ADVERSARIES. The composition theorem may not explicitly maintain properties of an adversary desired. For example, one might desire information theoretic security in the ROM, but this would fail should composition be used with a hash construction enjoying indifferentiability only with respect to complexity-theoretic adversaries [47, 54]. (This issue was first pointed out in [54].) Precise bounds on running time may also not be preserved, as mentioned above in the discussion of concrete security.

9 Deterministic, Hedged, and Efficiently-Searchable Encryption

The results thus far reveal that schemes proven secure in the ROM may not be secure when using practical hash function constructions, when security is measured by a multi-stage game. As seen in Section 5 this includes numerous important cryptographic tasks. As a first step, we here take one example, that of deterministic, hedged, or efficiently-searchable public-key encryption, and provide a proof of security when using any one of a number of indifferentiable hash constructions. We choose this example due to the extensive use of the ROM in prior results and the practical importance of the schemes [10, 11, 55]. Of course we cannot rely on Theorem 3.1, so our proof is done directly in the ideal primitive model. Nevertheless, our main result covers a relatively large mix of PKE schemes and hash functions.

We focus on the hash construction from [35], which composes a preimage-aware function (see below) with a fixed-input-length RO. While we can do analysis without relying on preimage-awareness, doing so simplifies and modularizes our result. Let $h^f\colon \{0,1\}^* \to \{0,1\}^n$ be a function using some underlying primitive f. Let $g\colon \{0,1\}^n \to \{0,1\}^n$ be a function. Let $H^{f,g}\colon \{0,1\}^* \to \{0,1\}^n$ be defined by $H^{f,g}(M) = g(h^f(M))$. We point out that many hash functions fall into this form, including the so-called NMAC construction [31], MCM [54], NIRP [47], and various SHA-3 competitors.

PUBLIC-KEY ENCRYPTION. Recall that a public-key encryption (PKE) scheme $\mathcal{AE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ consists of three algorithms. Key generation $\mathcal{K}$ outputs a public key, secret key pair. Encryption $\mathcal{E}$ takes a public key, a message m, and randomness r and outputs a ciphertext. Decryption $\mathcal{D}$ takes a secret key, a ciphertext, and outputs a plaintext or a distinguished symbol $\perp$. Following [10], we define for any scheme $\mathcal{AE}$ the

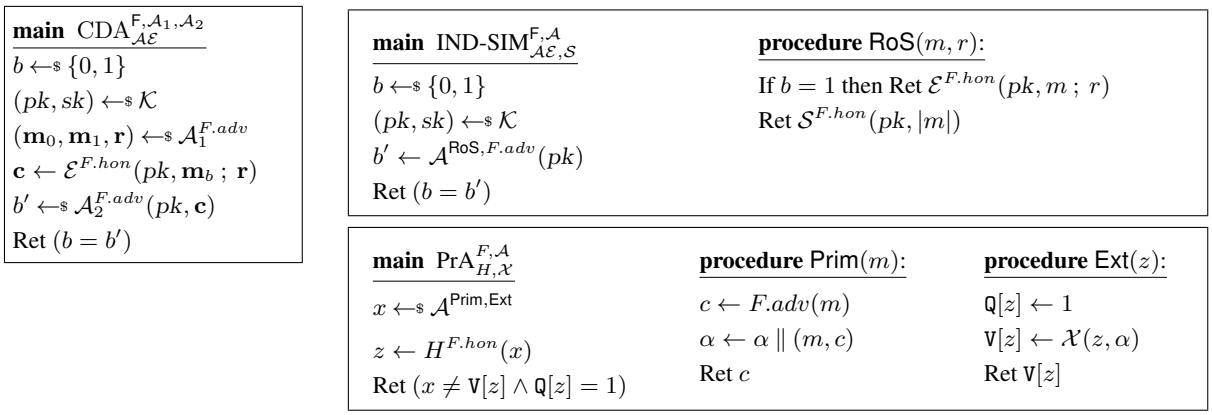

Figure 5: **(Left)** The non-adaptive CDA game. **(Right)** The IND-SIM and PrA games.

maximum public-key collision probability by

$$\mathsf{maxpk}_{\mathcal{AE}} = \max_{w \in \{0,1\}^*} \Pr\left[\, pk = w \ :\ (pk, sk) \leftarrow\$ \mathcal{K} \,\right] .$$

CDA SECURITY. In Figure 5 we detail the security game for (non-adaptive) chosen-distribution attacks [11]. (We conjecture that our main result below, Theorem 9.1, can be extended to cover adaptive adversaries as well, but do not provide proof.) This notion, orthogonal to the traditional notion of IND-CPA, captures the security of a PKE scheme when the randomness r used may not be a (sufficiently long) string of uniform bits. For the remainder of this section, fix a randomness length $\rho \geq 0$ and a message length $\omega > 0$. An (μ, ν)-mmr-source $\mathcal{M}$ is a randomized algorithm that outputs a triple of vectors $(\mathbf{m}_0, \mathbf{m}_1, \mathbf{r})$ such that $|\mathbf{m}_0| = |\mathbf{m}_0| = |\mathbf{r}| = \nu$, all components of $\mathbf{m}_0$ and $\mathbf{m}_1$ are bit strings of length ω, all components of $\mathbf{r}$ are bit strings of length ρ, and $(\mathbf{m}_b[i], \mathbf{r}[i]) \neq (\mathbf{m}_b[j], \mathbf{r}[j])$ for all $1 \leq i < j \leq \nu$ and all $b \in \{0,1\}$. Moreover, the source has min-entropy μ, meaning

$$\Pr\left[\, (\mathbf{m}_b[i], \mathbf{r}[i]) = (m', r') \ \mid\ (\mathbf{m}_0, \mathbf{m}_1, \mathbf{r}) \leftarrow\$ \mathcal{M} \,\right] \leq 2^{-\mu}$$

for all $b \in \{0,1\}$, all $1 \leq i \leq \nu$, and all (m', r').

A CDA adversary $\mathcal{A}_1, \mathcal{A}_2$ is a pair of procedures, the first of which is a (μ, ν)-mmr-source. The CDA advantage for a CDA adversary $\mathcal{A}_1, \mathcal{A}_2$ against scheme $\mathcal{AE}$ is defined by

$$\mathbf{Adv}^{\mathrm{cda}}_{\mathcal{AE},F}(\mathcal{A}_1, \mathcal{A}_2) = 2 \cdot \Pr\left[\, \mathrm{CDA}^{F,\mathcal{A}_1,\mathcal{A}_2}_{\mathcal{AE}} \Rightarrow \mathsf{true} \,\right] - 1 .$$

PREIMAGE AWARENESS. Dodis, Ristenpart, and Shrimpton's preimage awareness notion [35] generalizes collision resistance to include extractability. Game PrA is defined in Figure 5. We associate to any functionality F, hash construction H, extractor $\mathcal{X}$, and adversary $\mathcal{A}$ the PrA advantage defined by

$$\mathbf{Adv}^{\mathrm{pra}}_{H,F,\mathcal{X}}(\mathcal{A}) = \Pr\left[\, \mathrm{PrA}^{F,\mathcal{A}}_{H,\mathcal{X}} \Rightarrow \mathsf{true} \,\right] .$$

We point out that the game PrA does not abide by our convention that only the adversary queries $F.adv$, meaning PrA $\notin \mathcal{L}G$. Thus Theorems 3.1 and 6.1 do not apply to PrA. This is not a problem for past results or for our results below, both of which do not attempt to conclude PrA via indifferentiability-based composition.

IND-SIM SECURITY. We define a new notion of encryption scheme security that is of technical interest because it is as an intermediate step in proving Theorem 9.1, shown below. An encryption simulator for a scheme $\mathcal{AE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ is a procedure $\mathcal{S}$ that takes as input a public key and a message length and outputs a ciphertext. Game $\mathrm{IND\text{-}SIM}_{\mathcal{AE},\mathcal{S}}$ is shown in Figure 5. A IND-SIM adversary $\mathcal{A}$ can make multiple queries, but cannot repeat any queries. It measures the ability of an adversary to distinguish between encryptions of a chosen message under chosen randomness and the output of a simulator $\mathcal{S}$. We define the IND-SIM

advantage of an adversary $\mathcal{A}$ by

$$\mathbf{Adv}^{\text{ind-sim}}_{\mathcal{AE},\mathcal{S}}(\mathcal{A}) = 2 \cdot \Pr\left[\,\text{IND-SIM}^{\mathcal{A}}_{\mathcal{AE},\mathcal{S}} \Rightarrow \mathsf{true}\,\right] - 1\ .$$

Note that the adversary can choose the message and also the randomness used to encrypt it. In the standard model this security goal is unachievable if $\mathcal{E}$ uses no further randomness beyond that input. However, we will use IND-SIM security in the ROM when the adversary does not make any RO queries. In Appendix 10 we show that for a variety of encryption schemes, IND-SIM security in the ROM against adversaries who do not query the RO is implied by IND-CPA security of an underlying (randomized) scheme.

CDA SECURITY FOR PKE. Theorem 9.1 below establishes CDA security of PKE schemes that, during encryption, apply $g(h^f(M))$ once to hash an M including an encoding of the public key, as long as the scheme meets the IND-SIM notion above (in the ROM). The ROM schemes for deterministic, hedged, or efficiently-searchable encryption from [10, 11, 55] are of this form and have IND-SIM implied by the IND-CPA security of an underlying randomized encryption scheme. We make no assumptions about f, so the result applies both to hash functions based on an ideal cipher and ideal compression function.

We provide some brief intuition regarding the proof. The PrA security of f^+ means that, to learn anything about the value $g(f^+(M))$, the adversary must query f in order to compute $f^+(M)$. But the inclusion of the public key in the message hashed by $\mathcal{E}$ means that the source $\mathcal{A}_1$ is unlikely to be able to query any of the messages used in computing the challenge ciphertexts. Essentially this means that $\mathcal{E}$ gets randomness via queries to $g(f^+(M))$ that is hidden from the adversary, and this allows one to use the IND-SIM property of $\mathcal{AE}$ to show that ciphertexts leak no information about the challenge message, randomness pairs. This means that $\mathcal{A}_2$ learns nothing about the coins used by $\mathcal{A}_1$, and so the min-entropy of $\mathcal{A}_1$ implies that $\mathcal{A}_2$ has little chance of learning $g(f^+(M))$ outputs for M's used in computing the challenges.

Theorem 9.1 Let f be a functionality and g be a FIL RO. Let $H^{f,g}(M) = g(h^f(M))$ for some procedure h. Let $\mathcal{AE}$ be a PKE scheme that queries $H^{f,g}$ on a single message per $\mathcal{E}$ invocation, that message including (an encoding of) the public key. Let $\mathcal{A}_1, \mathcal{A}_2$ be a CDA adversary making at most q_f queries to f and q_g queries to g and where $\mathcal{A}_1$ is a (μ,ν)-mmr-source. Then for any encryption simulator $\mathcal{S}$ and PrA extractor $\mathcal{X}$ there exists an IND-SIM adversary $\mathcal{B}$ and a PrA adversary $\mathcal{C}$ such that.

$$\mathbf{Adv}^{\text{cda}}_{\mathcal{AE},(f,g)}(\mathcal{A}_1,\mathcal{A}_2) \le 4\cdot\mathbf{Adv}^{\text{ind-sim}}_{\mathcal{AE},\mathsf{RO},\mathcal{S}}(\mathcal{B}) + 4\cdot\mathbf{Adv}^{\text{pra}}_{h,f,\mathcal{X}}(\mathcal{C}) + \frac{2\nu q_g}{2^\mu} + 2q_g\cdot\mathsf{maxpk}_{\mathcal{AE}}$$

$\mathcal{B}$ makes no random oracle queries, makes ν RoS-queries, and runs in time that of $(\mathcal{A}_1,\mathcal{A}_2)$. $\mathcal{C}$ makes at most q_f primitive queries and runs in time at most that of $(\mathcal{A}_1,\mathcal{A}_2)$. $\square$

Proof: We assume without loss of generality that $\mathcal{A}_1, \mathcal{A}_2$ make no pointless queries (they do not repeat a query to f or to g). Note that this does not mean that $\mathcal{A}_2$ never repeats a query made by $\mathcal{A}_1$. We use a sequence of games and adversaries to show that

$$\begin{aligned}
&\Pr\left[\,\text{CDA}^{\mathcal{A}_1,\mathcal{A}_2}_{\mathcal{AE}} \Rightarrow \mathsf{true}\,\right] && (3)\\
&= \Pr\left[\,\mathrm{G}_0 \Rightarrow \mathsf{true}\,\right] && (4)\\
&\le \Pr\left[\,\mathrm{G}_1 \Rightarrow \mathsf{true}\,\right] + \Pr\left[\,\mathrm{G}_1 \text{ sets } \mathsf{bad}_3\,\right] + \Pr\left[\,\mathrm{G}_1 \text{ sets } \mathsf{bad}_1 \vee \mathrm{G}_1 \text{ sets } \mathsf{bad}_2\,\right] && (5)\\
&\le \Pr\left[\,\mathrm{G}_1 \Rightarrow \mathsf{true}\,\right] + \Pr\left[\,\mathrm{G}_1 \text{ sets } \mathsf{bad}_3\,\right] + q_g\cdot\mathsf{maxpk}_{\mathcal{AE}} + \mathbf{Adv}^{\text{pra}}_{h,f,\mathcal{X}}(\mathcal{C}_{1,2}) && (6)\\
&\le \Pr\left[\,\mathrm{G}_2 \Rightarrow \mathsf{true}\,\right] + \Pr\left[\,\mathrm{G}_2 \text{ sets } \mathsf{bad}_3\,\right] + q_g\cdot\mathsf{maxpk}_{\mathcal{AE}} + \mathbf{Adv}^{\text{pra}}_{h,f,\mathcal{X}}(\mathcal{C}_{1,2}) && (7)\\
&\le \Pr\left[\,\mathrm{G}_3 \Rightarrow \mathsf{true}\,\right] + \Pr\left[\,\mathrm{G}_3 \text{ sets } \mathsf{bad}_3\,\right] + 2\cdot\mathbf{Adv}^{\text{ind-sim}}_{\mathcal{AE},\mathsf{RO},\mathcal{S}}(\mathcal{B}) + q_g\mathsf{maxpk}_{\mathcal{AE}} + \mathbf{Adv}^{\text{pra}}_{h,f,\mathcal{X}}(\mathcal{C}_{1,2}) && (8)\\
&\le \frac{1}{2} + \Pr\left[\,\mathrm{G}_4 \text{ sets } \mathsf{bad}_3\,\right] + 2\cdot\mathbf{Adv}^{\text{ind-sim}}_{\mathcal{AE},\mathsf{RO},\mathcal{S}}(\mathcal{B}) + q_g\cdot\mathsf{maxpk}_{\mathcal{AE}} + \mathbf{Adv}^{\text{pra}}_{h,f,\mathcal{X}}(\mathcal{C}_{1,2}) && (9)\\
&\le \frac{1}{2} + \frac{q_g\nu}{2^\mu} + 2\cdot\mathbf{Adv}^{\text{ind-sim}}_{\mathcal{AE},\mathsf{RO},\mathcal{S}}(\mathcal{B}) + q_g\cdot\mathsf{maxpk}_{\mathcal{AE}} + 2\cdot\mathbf{Adv}^{\text{pra}}_{h,f,\mathcal{X}}(\mathcal{C}) && (10)
\end{aligned}$$

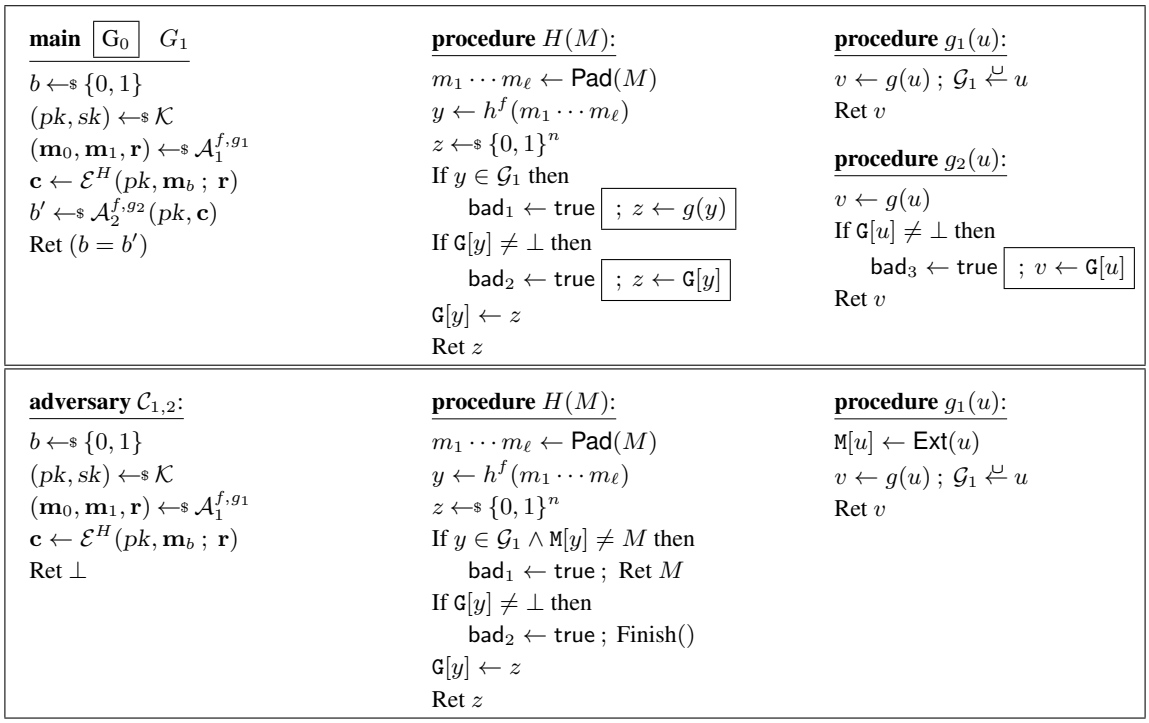

Figure 6: Games $\mathrm{G}_0, \mathrm{G}_1$ and adversary $\mathcal{C}_{1,2}$ used in the proof of Theorem 9.1.

Applying the definition of CDA advantage yields the advantage relation of the theorem statement. We now justify the inequalities above. Game G_0 (Figure 6, boxed statements included) implements exactly the game $\mathrm{CDA}_{\mathcal{AE}}$, justifying (3). To see this, note that the conditionals in procedures H and g_2 ensure that throughout the game a single random function g is defined. The three bad flags reflect three situations:

- bad_1 is set if $\mathcal{A}_1$ queried a point to g that matches the output of $h^f(M)$ for one of the messages M queried to H by $\mathcal{E}$.
- bad_2 is set if $\mathcal{E}$ caused a collision in the output of h^f.
- bad_3 is set if $\mathcal{A}_2$ queries g on a point that matches the output of $h^f(M)$ for one of the messages M queried to H by $\mathcal{E}$.

Game G_1 omits the boxed statements of game G_0. The two games are therefore identical until one of $\mathsf{bad}_1, \mathsf{bad}_2, \mathsf{bad}_3$ is set. Let "G_i sets bad_j" be the event that game G_i sets bad_j. The fundamental lemma of game-playing [20] and a union bound justifies (5).

We now show that the probability of bad_1 or bad_2 being set is bounded by the PrA advantage of an adversary against h^f plus the probability of $\mathcal{A}_1$ guessing the public key pk. The PrA adversary $\mathcal{C}_{1,2}$ is shown in Figure 6. It runs game G_1 up through the computation of the challenge ciphertexts. It queries its Ext oracle on every one of $\mathcal{A}_1$'s queries to g_1, recording the response in a table $\mathtt{M}$. It forwards $\mathcal{A}_1$'s queries to f to its own f oracle. If bad_1 is set, this corresponds to one of the messages M queried to H by $\mathcal{E}$ as having $h^f(M)$ equal to a value u queried to g_1. The Finish() procedure (not shown) finds the colliding pair of messages M, M' queried by $\mathcal{E}$ to H, queries Ext on $h^f(M)$ and $h^f(M')$ and finally halts $\mathcal{C}_{1,2}$, outputting whichever of M, M' was not returned by Ext. It is clear that $\mathcal{C}_{1,2}$ wins in the PrA game should bad_2 be set. For the value returned after bad_1 is set to be a win for $\mathcal{C}_{1,2}$ in the PrA game, it must be that $\mathtt{M}[y] \neq M$. But this only happens with probability $q_1 \cdot \mathsf{maxpk}_{\mathcal{AE}}$ because the pk is included in each query by $\mathcal{E}$ to H and the choice of pk is independent of the queries to f by $\mathcal{A}_1$. Let "E" be the event that $\mathtt{M}[y] = M$ for some $y \in \mathcal{G}_1$ during

the execution of $\mathrm{PrA}_h^{f,\mathcal{C}_{1,2}}$. We have that

$$\begin{aligned}\Pr[\,\mathrm{G}_1 \text{ sets } \mathsf{bad}_1 \vee \mathrm{G}_1 \text{ sets } \mathsf{bad}_2\,] &= \Pr[\,\mathrm{G}_1 \text{ sets } \mathsf{bad}_1 \vee \mathrm{G}_1 \text{ sets } \mathsf{bad}_2 \mid \mathsf{E}\,]\cdot\Pr[\,\mathsf{E}\,] \\ &\quad + \Pr\left[\,\mathrm{G}_1 \text{ sets } \mathsf{bad}_1 \vee \mathrm{G}_1 \text{ sets } \mathsf{bad}_2 \mid \overline{\mathsf{E}}\,\right]\cdot\Pr\left[\,\overline{\mathsf{E}}\,\right] \\ &\le \Pr[\,\mathsf{E}\,] + \Pr\left[\,\mathrm{G}_1 \text{ sets } \mathsf{bad}_1 \vee \mathrm{G}_1 \text{ sets } \mathsf{bad}_2 \mid \overline{\mathsf{E}}\,\right] \\ &\le q_g\cdot\mathsf{maxpk}_{\mathcal{AE}} + \mathbf{Adv}^{\mathrm{pra}}_{h,f,\mathcal{X}}(\mathcal{C}_{1,2})\ .\end{aligned}$$

We have justified (6).

Game G_2 (Figure 6, boxed statements omitted) is the same as G_1 except that the setting of bad_1 and bad_2 has been dropped. This doesn't change the behavior of the game, and so

$$\Pr[\,\mathrm{G}_2 \Rightarrow \mathsf{true}\,] = \Pr[\,\mathrm{G}_1 \Rightarrow \mathsf{true}\,] \quad \text{and} \quad \Pr[\,\mathrm{G}_2 \text{ sets } \mathsf{bad}_3\,] = \Pr[\,\mathrm{G}_1 \text{ sets } \mathsf{bad}_3\,]\ ,$$

which justifies (7).

Game G_3 replaces ciphertext vector $\mathbf{c}$ given to $\mathcal{A}_2$ with the output of an algorithm $\mathcal{S}^{\nu}(pk,\omega)$. This runs the encryption simulator $\mathcal{S}(pk,\omega)$ a total of ν times and returns the resulting vector of ciphertexts. We now define two IND-SIM adversaries $\mathcal{B}'$ and $\mathcal{B}''$ that will be used to bound the first two terms of the right hand side of equation (3). The adversaries are defined in Figure 7. They simulate the primitives f, g for the underlying adversary $\mathcal{A}_1, \mathcal{A}_2$ and in computing H. The adversaries are identical except that $\mathcal{B}'$ returns $b = 1$ if $\mathcal{A}_2$ guessed the correct bit while $\mathcal{B}''$ returns $b = 1$ if $\mathcal{A}_2$ set bad_3. Note that in both adversaries, $\mathcal{E}^H(pk, \mathbf{m}_b\,;\,\mathbf{r})$ is executed (with return value ignored). This is done to merely cause $H(M)$ to be called on each M value generated by $\mathcal{E}$, as is done in both game G_2 and game G_3. By construction we have that

$$\Pr\left[\,\text{IND-SIM1}^{\mathcal{B}'}_{\mathcal{AE}} \Rightarrow 1\,\right] = \Pr[\,\mathrm{G}_2 \Rightarrow \mathsf{true}\,] \quad \text{and} \quad \Pr\left[\,\text{IND-SIM0}^{\mathcal{B}'}_{\mathcal{AE}} \Rightarrow 1\,\right] = \Pr[\,\mathrm{G}_3 \Rightarrow \mathsf{true}\,]$$

and that

$$\Pr\left[\,\text{IND-SIM1}^{\mathcal{B}''}_{\mathcal{AE}} \Rightarrow 1\,\right] = \Pr[\,\mathrm{G}_2 \text{ sets } \mathsf{bad}_3\,] \ \text{ and } \ \Pr\left[\,\text{IND-SIM0}^{\mathcal{B}''}_{\mathcal{AE}} \Rightarrow 1\,\right] = \Pr[\,\mathrm{G}_3 \text{ sets } \mathsf{bad}_3\,]\ .$$

Let $\mathcal{B}$ be whichever of $\mathcal{B}'$ and $\mathcal{B}''$ achieves larger advantage. Then the above equations, together with the definition of IND advantage (Section 2), justify (8). In game G_3 the execution of $\mathcal{A}_2$ (including its oracles) does not rely on the challenge bit b. Thus $\Pr[\mathrm{G}_3 \Rightarrow \mathsf{true}] = 1/2$, justifying (9).

We now bound the probability that bad_3 is set in G_3. We first move to a final game G_4 (Figure 7). It defers running of $\mathcal{A}_1$ until after $\mathcal{A}_2$ and moves the setting of bad_3 from g_2 to H. We have that $\Pr[\mathrm{G}_3 \text{ sets } \mathsf{bad}_3\,] = \Pr[\mathrm{G}_4 \text{ sets } \mathsf{bad}_3\,]$ because the executions of $\mathcal{A}_1$ and $\mathcal{A}_2$ are independent and the check for bad_3 in G_4 triggers on the same event as the condition for setting bad_3 in G_3.

We now build a PrA adversary $\mathcal{C}_3$ against h^f. It is shown in Figure 7, and runs G_4 with the following changes. For each g_2 query by $\mathcal{A}_2$, adversary $\mathcal{C}_3$ queries u to its Ext. It forwards all f queries to its own f oracle. It returns M if a query $H(M)$ by $\mathcal{E}$ is such that $h^f(M) = u$ for some u previously queried to g_2 by $\mathcal{A}_2$ and M does not match the value output by Ext on that point. Similarly to the analysis of $\mathcal{C}_{1,2}$ above, let "E" be the event that $\mathtt{M}[u] = M$ for some u queried to g_2 by $\mathcal{A}_2$ and M queried to H by $\mathcal{E}$ during the execution of $\mathrm{PrA}_h^{f,\mathcal{C}_3}$. We have that

$$\begin{aligned}\Pr[\,\mathrm{G}_4 \text{ sets } \mathsf{bad}_3\,] &= \Pr[\,\mathrm{G}_4 \text{ sets } \mathsf{bad}_3 \mid \mathsf{E}\,]\cdot\Pr[\,\mathsf{E}\,] + \Pr\left[\,\mathrm{G}_4 \text{ sets } \mathsf{bad}_3 \mid \overline{\mathsf{E}}\,\right]\cdot\Pr\left[\,\overline{\mathsf{E}}\,\right] \\ &\le \Pr[\,\mathsf{E}\,] + \Pr\left[\,\mathrm{G}_4 \text{ sets } \mathsf{bad}_3 \mid \overline{\mathsf{E}}\,\right] \\ &\le \frac{q_g\nu}{2^{\mu}} + \mathbf{Adv}^{\mathrm{pra}}_{h,f,\mathcal{X}}(\mathcal{C}_3)\ .\end{aligned}$$

To bound $\Pr[\mathsf{E}]$ as we did above, we used the fact that $\mathcal{A}_1$ is an mmr-source with min-entropy μ. That means that any message it outputs will equal some fixed message with probability at most $2^{-\mu}$. Let $\mathcal{C}$ be whichever of $\mathcal{C}_{1,2}$ and $\mathcal{C}_3$ achieves higher advantage. We have justified (10). ∎

main G_2 $\boxed{\mathrm{G}_3}$	**procedure** $H(M)$:	**procedure** $g_2(u)$:
$b \leftarrow\!\!\$\ \{0,1\}$	$m_1 \cdots m_\ell \leftarrow \mathsf{Pad}(M)$	$v \leftarrow g(u)$
$(pk, sk) \leftarrow\!\!\$\ \mathcal{K}$	$y \leftarrow h^f(m_1 \cdots m_\ell)$	If $\mathtt{G}[u] \neq \bot$ then $\mathsf{bad}_3 \leftarrow \mathsf{true}$
$(\mathbf{m}_0, \mathbf{m}_1, \mathbf{r}) \leftarrow\!\!\$\ \mathcal{A}_1^{f,g}$	$z \leftarrow\!\!\$\ \{0,1\}^n$	Ret v
$\mathbf{c} \leftarrow \mathcal{E}^H(pk, \mathbf{m}_b\ ;\ \mathbf{r})$	$\mathtt{G}[y] \leftarrow z$	
$\boxed{\mathbf{c} \leftarrow\!\!\$\ \mathcal{S}^\nu(pk, \omega)}$	Ret z	
$b' \leftarrow\!\!\$\ \mathcal{A}_2^{f,g_2}(pk, \mathbf{c})$		
Ret $(b = b')$		

adversary $\mathcal{B}'(pk)$:	**procedure** $H(M)$:	**procedure** $g_2(u)$:
$b \leftarrow\!\!\$\ \{0,1\}$	$m_1 \cdots m_\ell \leftarrow \mathsf{Pad}(M)$	$v \leftarrow g(u)$
$(\mathbf{m}_0, \mathbf{m}_1, \mathbf{r}) \leftarrow\!\!\$\ \mathcal{A}_1^{f,g}$	$y \leftarrow h^f(m_1 \cdots m_\ell)$	If $\mathtt{G}[u] \neq \bot$ then $\mathsf{bad}_3 \leftarrow \mathsf{true}$
$\mathcal{E}^H(pk, \mathbf{m}_b\ ;\ \mathbf{r})$	$z \leftarrow\!\!\$\ \{0,1\}^n$	Ret v
$\mathbf{c} \leftarrow \mathsf{RoS}(\mathbf{m}_b, \mathbf{r})$	$\mathtt{G}[y] \leftarrow z$	
$b' \leftarrow\!\!\$\ \mathcal{A}_2^{f,g_2}(pk, \mathbf{c})$	Ret z	
If $(b = b')$ then Ret 1		
Ret 0		

adversary $\mathcal{B}''(pk)$:	**procedure** $H(M)$:	**procedure** $g_2(u)$:
$b \leftarrow\!\!\$\ \{0,1\}$	$m_1 \cdots m_\ell \leftarrow \mathsf{Pad}(M)$	$v \leftarrow g(u)$
$(\mathbf{m}_0, \mathbf{m}_1, \mathbf{r}) \leftarrow\!\!\$\ \mathcal{A}_1^{f,g}$	$y \leftarrow h^f(m_1 \cdots m_\ell)$	If $\mathtt{G}[u] \neq \bot$ then $\mathsf{bad}_3 \leftarrow \mathsf{true}$
$\mathcal{E}^H(pk, \mathbf{m}_b\ ;\ \mathbf{r})$	$z \leftarrow\!\!\$\ \{0,1\}^n$	Ret v
$\mathbf{c} \leftarrow \mathsf{RoS}(\mathbf{m}_b, \mathbf{r})$	$\mathtt{G}[y] \leftarrow z$	
$b' \leftarrow\!\!\$\ \mathcal{A}_2^{f,g_2}(pk, \mathbf{c})$	Ret z	
If $(\mathsf{bad}_3 = \mathsf{true})$ then Ret 1		
Ret 0		

main G_4	**procedure** $H(M)$:	**procedure** $g_2(u)$:
$b \leftarrow\!\!\$\ \{0,1\}$	$m_1 \cdots m_\ell \leftarrow \mathsf{Pad}(M)$	$v \leftarrow g(u)$
$(pk, sk) \leftarrow\!\!\$\ \mathcal{K}$	$y \leftarrow h^f(m_1 \cdots m_\ell)$	$\mathtt{G}[u] \leftarrow v$
$\mathbf{c} \leftarrow\!\!\$\ \mathcal{S}^\nu(pk, \omega)$	$z \leftarrow\!\!\$\ \{0,1\}^n$	Ret v
$b' \leftarrow\!\!\$\ \mathcal{A}_2^{f,g_2}(pk, \mathbf{c})$	If $\mathtt{G}[y] \neq \bot$ then $\mathsf{bad}_3 \leftarrow \mathsf{true}$	
$(\mathbf{m}_0, \mathbf{m}_1, \mathbf{r}) \leftarrow\!\!\$\ \mathcal{A}_1^{f,g}$	Ret z	
$\mathbf{c} \leftarrow \mathcal{E}^H(pk, \mathbf{m}_b\ ;\ \mathbf{r})$		
Ret $(b = b')$		

adversary $\mathcal{C}_3$:	**procedure** $H(M)$:	**procedure** $g_2(u)$:
$b \leftarrow\!\!\$\ \{0,1\}$	$m_1 \cdots m_\ell \leftarrow \mathsf{Pad}(M)$	$\mathtt{M}[u] \leftarrow \mathsf{Ext}(u)$
$(pk, sk) \leftarrow\!\!\$\ \mathcal{K}$	$y \leftarrow h^f(m_1 \cdots m_\ell)$	$v \leftarrow g(u)$
$\mathbf{c} \leftarrow\!\!\$\ \mathcal{S}^\nu(pk, \omega)$	$z \leftarrow\!\!\$\ \{0,1\}^n$	$\mathtt{G}[u] \leftarrow v$
$b' \leftarrow\!\!\$\ \mathcal{A}_2^{f,g_2}(pk, \mathbf{c})$	If $\mathtt{G}[y] \neq \bot \wedge \mathtt{M}[u] \neq M$ then	Ret v
$(\mathbf{m}_0, \mathbf{m}_1, \mathbf{r}) \leftarrow\!\!\$\ \mathcal{A}_1^{f,g}$	$\quad \mathsf{bad}_3 \leftarrow \mathsf{true}$; Ret M	
$\mathbf{c} \leftarrow \mathcal{E}^H(pk, \mathbf{m}_b\ ;\ \mathbf{r})$	Ret z	
Ret $\bot$		

Figure 7: Games and adversaries used in the proof of Theorem 9.1.

main $\mathrm{IND1}^{\mathcal{A}}_{\mathcal{AE}}$	**procedure** $\mathsf{LoR}(m_0, m_1)$:
$b' \leftarrow\!\!\$\ \mathcal{A}^{\mathsf{LoR}}$	$r \leftarrow\!\!\$\ \{0,1\}^{\rho}$
Ret b'	Ret $\mathcal{E}_{\mathrm{r}}(m_1\ ;\ r)$

main $\mathrm{IND0}^{\mathcal{A}}_{\mathcal{AE}}$	**procedure** $\mathsf{LoR}(m_0, m_1)$:
$b' \leftarrow\!\!\$\ \mathcal{A}^{\mathsf{LoR}}$	$r \leftarrow\!\!\$\ \{0,1\}^{\rho}$
Ret b'	Ret $\mathcal{E}_{\mathrm{r}}(m_0\ ;\ r)$

Figure 8: The games that define IND-CPA security.

10 IND-SIM Security of Some Encryption Schemes

For any PKE scheme $\mathcal{AE}$ define $\text{IND-SIM1}_{\mathcal{AE},\mathcal{S}}$ be defined just as $\text{IND-SIM}_{\mathcal{AE},\mathcal{S}}$ except that b is set to one at the beginning of **main**. Likewise define $\text{IND-SIM1}_{\mathcal{AE},\mathcal{S}}$ to be $\text{IND-SIM}_{\mathcal{AE},\mathcal{S}}$ except that b is set to zero at the beginning of **main**.

SEMANTIC SECURITY. We define the typical notion of IND-CPA for public key encryption. For a PKE scheme $\mathcal{AE} = (\mathcal{K}, \mathcal{E}_{\mathrm{r}}, \mathcal{D}_{\mathrm{r}})$ with randomness length ρ we define the $\mathrm{IND1}_{\mathcal{AE}}$ and $\mathrm{IND0}_{\mathcal{AE}}$ games in Figure 8. IND-CPA advantage of an adversary $\mathcal{A}$ is defined by

$$\mathbf{Adv}^{\mathrm{ind}}_{\mathcal{AE}}(\mathcal{A}) = \Pr\left[\,\mathrm{IND1}^{\mathcal{A}}_{\mathcal{AE}} \Rightarrow 1\,\right] - \Pr\left[\,\mathrm{IND0}^{\mathcal{A}}_{\mathcal{AE}} \Rightarrow 1\,\right] .$$

THE REwH1 AND EwH SCHEMES. We prove the first Randomized-Encrypt-With-Hash (REwH1) scheme from [11] and, as a corollary, the Encrypt-With-Hash (EwH) scheme from [10] to be IND-SIM secure in the case that no random oracle queries are allowed. This is the required security for Theorem 9.1. We expect other ROM schemes can similarly be analyzed. Let $\mathcal{AE} = (\mathcal{K}, \mathcal{E}_{\mathrm{r}}, \mathcal{D}_{\mathrm{r}})$ be a PKE encryption scheme with randomness length ρ. Let RO have range size ρ bits. The encryption schemes $\mathsf{REwH1} = (\mathcal{K}, \mathcal{E}_1, \mathcal{D}_{\mathrm{r}})$ from [11] inherits $\mathcal{K}$ and $\mathcal{D}_{\mathrm{r}}$ from $\mathcal{AE}$ and has encryption defined as

$$\mathcal{E}_1^{\mathsf{RO}}(pk, m\ ;\ r) = \mathcal{E}_{\mathrm{r}}(pk, m\ ;\ \mathsf{RO}(pk \parallel m \parallel r)) .$$

This scheme generalizes the Encrypt-with-Hash scheme from [10], the latter derived by setting $|\rho| = 0$.

Theorem 10.1 Let $\mathcal{A}$ be an IND-SIM adversary making no RO queries. Then there exists an encryption simulator $\mathcal{S}$ and IND-CPA adversary $\mathcal{B}$ such that

$$\mathbf{Adv}^{\text{ind-sim}}_{\mathsf{REwH1},\mathsf{RO},\mathcal{S}}(\mathcal{A}) \le \mathbf{Adv}^{\mathrm{ind}}_{\mathcal{AE}}(\mathcal{B}) .$$

$\mathcal{B}$ runs in time that of $\mathcal{A}$. □

Proof: The encryption simulator $\mathcal{S}$ works as follows. On input pk, x it chooses $r \leftarrow\!\!\$\ \{0,1\}^{\rho}$, then runs $\mathcal{E}_{\mathrm{r}}(pk, 0^x\ ;\ r)$, and finally outputs the resulting ciphertext. Adversary $\mathcal{B}$ works as follows. It implements $\text{IND-SIM}^{\mathsf{RO},\mathcal{A}}_{\mathsf{REwH1},\mathcal{S}}$ except for the following two changes. First, it answers an RoS query on m, r by returning the result of querying $LR(0^{|m|}, m)$. Second, it outputs the bit output by $\mathcal{A}$. Because $\mathcal{A}$ never queries RO and never repeats an RoS query, the output of RO is the same as fresh randomness as used in the LR oracle. Thus, we have that

$$\Pr\left[\,\text{IND-SIM1}^{\mathsf{RO},\mathcal{A}}_{\mathsf{REwH1},\mathcal{S}} \Rightarrow \mathsf{true}\,\right] = \Pr\left[\,\mathrm{IND1}^{\mathcal{B}}_{\mathcal{AE}} \Rightarrow \mathsf{true}\,\right]$$

and that

$$\Pr\left[\,\text{IND-SIM0}^{\mathsf{RO},\mathcal{A}}_{\mathsf{REwH1},\mathcal{S}} \Rightarrow \mathsf{false}\,\right] = \Pr\left[\,\mathrm{IND0}^{\mathcal{B}}_{\mathcal{AE}} \Rightarrow \mathsf{false}\,\right] ,$$

which together imply the advantage statement of the theorem. ▌

THE EaH SCHEME. We now treat the Encrypt-And-Hash efficiently searchable encryption scheme from [10]. Let $\mathcal{AE} = (\mathcal{K}, \mathcal{E}_{\mathrm{r}}, \mathcal{D}_{\mathrm{r}})$ be a PKE encryption scheme with randomness length ρ. Let RO have range size ω bits. The efficiently-searchable encryption scheme $\mathsf{EaH} = (\mathcal{K}, \mathcal{E}_2, \mathcal{D}_2)$ from [10] inherits $\mathcal{K}$ from $\mathcal{AE}$, has

randomness length 0 and has encryption defined as

$$\mathcal{E}_2^{\mathsf{RO}}(pk, m) = \mathsf{RO}(pk \,\|\, m) \,\|\, \mathcal{E}_{\mathrm{r}}(pk, m \,;\, r')$$

where $r' \leftarrow\!\!\$\ \{0,1\}^\rho$ is chosen fresh. Note that $\mathcal{E}_2$ is randomized by choice of r', unlike the schemes above which take all randomness as external inputs. By setting the "randomness length" of EaH to zero, we achieve syntactic match with the IND-SIM game — the r value queried is ignored. (Here we just consider randomness length to cover the portion of randomness controlled by the adversary.) Decryption $\mathcal{D}_2$ is defined in [10].

Theorem 10.2 Let $\mathcal{A}$ be an IND-SIM adversary making no RO queries. Then there exists an encryption simulator $\mathcal{S}$ and IND-CPA adversary $\mathcal{B}$ such that

$$\mathbf{Adv}^{\text{ind-sim}}_{\mathsf{EaH},\mathsf{RO},\mathcal{S}}(\mathcal{A}) \le \mathbf{Adv}^{\text{ind}}_{\mathcal{AE}}(\mathcal{B}) \ .$$

$\mathcal{B}$ runs in time that of $\mathcal{A}$. $\square$

Proof: The simulator $\mathcal{S}$ works as follows. On input pk, x it chooses $h \leftarrow\!\!\$\ \{0,1\}^\omega$ and $r' \leftarrow\!\!\$\ \{0,1\}^\rho$, runs $c \leftarrow \mathcal{E}_{\mathrm{r}}(pk, 0^x \,;\, r')$, and outputs $h \,\|\, c$. Adversary $\mathcal{B}$ works as follows. It implements $\text{IND-SIM}^{\mathsf{RO},\mathcal{A}}_{\mathsf{EaH},\mathcal{S}}$ except for the following two changes. First, it answers an RoS query on m by returning the result of querying $LR(0^{|m|}, m)$. Second, it outputs the bit output by $\mathcal{A}$. Because $\mathcal{A}$ never queries RO and never repeats an RoS query, the output of RO is the same as fresh randomness (as used in the simulator). Thus, we have that

$$\Pr\left[\ \text{IND-SIM1}^{\mathsf{RO},\mathcal{A}}_{\mathsf{REwH1},\mathcal{S}} \Rightarrow \mathsf{true}\ \right] = \Pr\left[\ \text{IND1}^{\mathcal{B}}_{\mathcal{AE}} \Rightarrow \mathsf{true}\ \right]$$

and that

$$\Pr\left[\ \text{IND-SIM0}^{\mathsf{RO},\mathcal{A}}_{\mathsf{REwH1},\mathcal{S}} \Rightarrow \mathsf{false}\ \right] = \Pr\left[\ \text{IND0}^{\mathcal{B}}_{\mathcal{AE}} \Rightarrow \mathsf{false}\ \right] \ ,$$

which together imply the advantage statement of the theorem. ▌

Acknowledgments

Thomas Ristenpart was supported in part by Mihir Bellare's NSF grant CCF-0915675 and by a UCSD Center for Networked Systems grant. Hovav Shacham was supported by the MURI program under AFOSR Grant No. FA9550-08-1-0352 and (while at the Weizmann Institute) by a Koshland Scholars Program postdoctoral fellowship. Thomas Shrimpton was supported by NSF CAREER grant CNS-0845610.

We thank Mihir Bellare, Mike Dahlin, Yevgeniy Dodis, Daniele Micciancio, and Moni Naor for helpful discussions about this work.

References

[1] T. Acar, M. Belenkiy, M. Bellare, and D. Cash. Cryptographic agility and its relation to circular encryption. In H. Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 403–422, French Riviera, May 30 – June 3, 2010. Springer, Berlin, Germany.

[2] M. Albrecht, P. Farshim, K. Paterson, and G. Watson. On cipher-dependent related-key attacks in the ideal cipher model. In *Fast Software Encryption*, 2011.

[3] J. Alwen, J. Katz, Y. Lindell, G. Persiano, A. Shelat, and I. Visconti. Collusion-free multiparty computation in the mediated model. In S. Halevi, editor, *CRYPTO 2009*, volume 5677 of *LNCS*, pages 524–540, Santa Barbara, CA, USA, Aug. 16–20, 2009. Springer, Berlin, Germany.

[4] J. Alwen, A. Shelat, and I. Visconti. Collusion-free protocols in the mediated model. In D. Wagner, editor, *CRYPTO 2008*, volume 5157 of *LNCS*, pages 497–514, Santa Barbara, CA, USA, Aug. 17–21, 2008. Springer, Berlin, Germany.

[5] E. Andreeva, B. Mennink, and B. Preneel. On the indifferentiability of the Grøstl hash function. In J. A. Garay and R. D. Prisco, editors, *SCN 10*, volume 6280 of *LNCS*, pages 88–105, Amalfi, Italy, Sept. 13–15, 2010. Springer, Berlin, Germany.

[6] B. Applebaum, D. Cash, C. Peikert, and A. Sahai. Fast cryptographic primitives and circular-secure encryption based on hard learning problems. In S. Halevi, editor, *CRYPTO 2009*, volume 5677 of *LNCS*, pages 595–618, Santa Barbara, CA, USA, Aug. 16–20, 2009. Springer, Berlin, Germany.

[7] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, and D. Song. Provable data possession at untrusted stores. In S. De Capitani di Vimercati and P. Syverson, editors, *Proceedings of CCS 2007*, pages 598–609. ACM Press, Oct. 2007.

[8] M. Backes, M. Dürmuth, and D. Unruh. OAEP is secure under key-dependent messages. In J. Pieprzyk, editor, *ASIACRYPT 2008*, volume 5350 of *LNCS*, pages 506–523, Melbourne, Australia, Dec. 7–11, 2008. Springer, Berlin, Germany.

[9] B. Barak, I. Haitner, D. Hofheinz, and Y. Ishai. Bounded key-dependent message security. In H. Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 423–444, French Riviera, May 30 – June 3, 2010. Springer, Berlin, Germany.

[10] M. Bellare, A. Boldyreva, and A. O'Neill. Deterministic and efficiently searchable encryption. In A. Menezes, editor, *CRYPTO 2007*, volume 4622 of *LNCS*, pages 535–552, Santa Barbara, CA, USA, Aug. 19–23, 2007. Springer, Berlin, Germany.

[11] M. Bellare, Z. Brakerski, M. Naor, T. Ristenpart, G. Segev, H. Shacham, and S. Yilek. Hedged public-key encryption: How to protect against bad randomness. In M. Matsui, editor, *ASIACRYPT 2009*, volume 5912 of *LNCS*, pages 232–249, Tokyo, Japan, Dec. 6–10, 2009. Springer, Berlin, Germany.

[12] M. Bellare, D. Cash, and R. Miller. A comparitive study of achievability of security against related-key attack. ePrint Archive, `http://eprint.iacr.org/2011/252`.

[13] M. Bellare, A. Desai, E. Jokipii, and P. Rogaway. A concrete security treatment of symmetric encryption. In *38th FOCS*, pages 394–403, Miami Beach, Florida, Oct. 19–22, 1997. IEEE Computer Society Press.

[14] M. Bellare, M. Fischlin, A. O'Neill, and T. Ristenpart. Deterministic encryption: Definitional equivalences and constructions without random oracles. In D. Wagner, editor, *CRYPTO 2008*, volume 5157 of *LNCS*, pages 360–378, Santa Barbara, CA, USA, Aug. 17–21, 2008. Springer, Berlin, Germany.

[15] M. Bellare and S. Keelveedhi. Authenticated and misuse-resistant encryption of key-dependent data. In P. Rogaway, editor, *Advances in Cryptology — CRYPTO 2011*. Springer, 2011.

[16] M. Bellare and T. Kohno. A theoretical treatment of related-key attacks: RKA-PRPs, RKA-PRFs, and applications. In E. Biham, editor, *EUROCRYPT 2003*, volume 2656 of *LNCS*, pages 491–506, Warsaw, Poland, May 4–8, 2003. Springer, Berlin, Germany.

[17] M. Bellare, D. Pointcheval, and P. Rogaway. Authenticated key exchange secure against dictionary attacks. In B. Preneel, editor, *EUROCRYPT 2000*, volume 1807 of *LNCS*, pages 139–155, Bruges, Belgium, May 14–18, 2000. Springer, Berlin, Germany.

[18] M. Bellare and T. Ristenpart. Multi-property-preserving hash domain extension and the EMD transform. In X. Lai and K. Chen, editors, *ASIACRYPT 2006*, volume 4284 of *LNCS*, pages 299–314, Shanghai, China, Dec. 3–7, 2006. Springer, Berlin, Germany.

[19] M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In V. Ashby, editor, *ACM CCS 93*, pages 62–73, Fairfax, Virginia, USA, Nov. 3–5, 1993. ACM Press.

[20] M. Bellare and P. Rogaway. The security of triple encryption and a framework for code-based game-playing proofs. In S. Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 409–426, St. Petersburg, Russia, May 28 – June 1, 2006. Springer, Berlin, Germany.

[21] G. Bertoni, J. Daemen, M. Peeters, and G. Van Assche. On the indifferentiability of the sponge construction. In N. P. Smart, editor, *EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 181–197, Istanbul, Turkey, Apr. 13–17, 2008. Springer, Berlin, Germany.

[22] J. Black, P. Rogaway, and T. Shrimpton. Encryption-scheme security in the presence of key-dependent messages. In K. Nyberg and H. M. Heys, editors, *SAC 2002*, volume 2595 of *LNCS*, pages 62–75, St. John's, Newfoundland, Canada, Aug. 15–16, 2003. Springer, Berlin, Germany.

[23] A. Boldyreva, D. Cash, M. Fischlin, and B. Warinschi. Foundations of non-malleable hash and one-way functions. In M. Matsui, editor, *ASIACRYPT 2009*, volume 5912 of *LNCS*, pages 524–541, Tokyo, Japan, Dec. 6–10, 2009. Springer, Berlin, Germany.

[24] A. Boldyreva, S. Fehr, and A. O'Neill. On notions of security for deterministic encryption, and efficient constructions without random oracles. In D. Wagner, editor, *CRYPTO 2008*, volume 5157 of *LNCS*, pages 335–359, Santa Barbara, CA, USA, Aug. 17–21, 2008. Springer, Berlin, Germany.

[25] D. Boneh, S. Halevi, M. Hamburg, and R. Ostrovsky. Circular-secure encryption from decision diffie-hellman. In D. Wagner, editor, *CRYPTO 2008*, volume 5157 of *LNCS*, pages 108–125, Santa Barbara, CA, USA, Aug. 17–21, 2008. Springer, Berlin, Germany.

[26] Z. Brakerski and S. Goldwasser. Circular and leakage resilient public-key encryption under subgroup indistinguishability - (or: Quadratic residuosity strikes back). In *CRYPTO 2010*, LNCS, pages 1–20, Santa Barbara, CA, USA, Aug. 2010. Springer, Berlin, Germany.

[27] R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd FOCS*, pages 136–145, Las Vegas, Nevada, USA, Oct. 14–17, 2001. IEEE Computer Society Press.

[28] R. Canetti, Y. Dodis, R. Pass, and S. Walfish. Universally composable security with global setup. In S. P. Vadhan, editor, *TCC 2007*, volume 4392 of *LNCS*, pages 61–85, Amsterdam, The Netherlands, Feb. 21–24, 2007. Springer, Berlin, Germany.

[29] R. Canetti and T. Rabin. Universal composition with joint state. In D. Boneh, editor, *CRYPTO 2003*, volume 2729 of *LNCS*, pages 265–281, Santa Barbara, CA, USA, Aug. 17–21, 2003. Springer, Berlin, Germany.

[30] D. Chang and M. Nandi. Improved indifferentiability security analysis of chopMD hash function. In K. Nyberg, editor, *FSE 2008*, volume 5086 of *LNCS*, pages 429–443, Lausanne, Switzerland, Feb. 10–13, 2008. Springer, Berlin, Germany.

[31] J.-S. Coron, Y. Dodis, C. Malinaud, and P. Puniya. Merkle-Damgård revisited: How to construct a hash function. In V. Shoup, editor, *CRYPTO 2005*, volume 3621 of *LNCS*, pages 430–448, Santa Barbara, CA, USA, Aug. 14–18, 2005. Springer, Berlin, Germany.

[32] Y. Dodis, R. Oliveira, and K. Pietrzak. On the generic insecurity of the full domain hash. In V. Shoup, editor, *CRYPTO 2005*, volume 3621 of *LNCS*, pages 449–466, Santa Barbara, CA, USA, Aug. 14–18, 2005. Springer, Berlin, Germany.

[33] Y. Dodis, R. Ostrovsky, L. Reyzin, and A. Smith. Fuzzy Extractors: How to Generate Strong Keys from Biometrics and Other Noisy Data. *SIAM Journal of Computing*, 38(1):97–139, 2008.

[34] Y. Dodis, L. Reyzin, R. L. Rivest, and E. Shen. Indifferentiability of permutation-based compression functions and tree-based modes of operation, with applications to MD6. In O. Dunkelman, editor, *FSE 2009*, volume 5665 of *LNCS*, pages 104–121, Leuven, Belgium, Feb. 22–25, 2009. Springer, Berlin, Germany.

[35] Y. Dodis, T. Ristenpart, and T. Shrimpton. Salvaging Merkle-Damgård for practical applications. In A. Joux, editor, *EUROCRYPT 2009*, volume 5479 of *LNCS*, pages 371–388, Cologne, Germany, Apr. 26–30, 2009. Springer, Berlin, Germany.

[36] J. Franks, P. Hallam-Baker, J. Hostetler, P. Leach, A. Luotonen, E. Sink, and L. Stewart. An Extension to HTTP: Digest Access Authentication. RFC 2069 (Proposed Standard), Jan. 1997. Obsoleted by RFC 2617.

[37] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game, or a completeness theorem for protocols with honest majority. In A. Aho, editor, *19th ACM STOC*, pages 218–229, New York City,, New York, USA, May 25–27, 1987. ACM Press.

[38] S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270–299, 1984.

[39] S. Goldwasser, S. Micali, and R. L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing*, 17(2):281–308, Apr. 1988.

[40] S. Halevi and H. Krawczyk. Security under key-dependent inputs. In P. Ning, S. D. C. di Vimercati, and P. F. Syverson, editors, *ACM CCS 07*, pages 466–475, Alexandria, Virginia, USA, Oct. 28–31, 2007. ACM Press.

[41] S. Hirose, J. H. Park, and A. Yun. A simple variant of the Merkle-Damgård scheme with a permutation. In K. Kurosawa, editor, *ASIACRYPT 2007*, volume 4833 of *LNCS*, pages 113–129, Kuching, Malaysia, Dec. 2–6, 2007. Springer, Berlin, Germany.

[42] D. Hofheinz and D. Unruh. Towards key-dependent message security in the standard model. In N. P. Smart, editor, *EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 108–126, Istanbul, Turkey, Apr. 13–17, 2008. Springer, Berlin, Germany.

[43] A. Juels and B. Kaliski. PORs: Proofs of retrievability for large files. In S. De Capitani di Vimercati and P. Syverson, editors, *Proceedings of CCS 2007*, pages 584–97. ACM Press, Oct. 2007.

[44] S. Kamara, P. Mohassel, and M. Raykova. Outsourcing multi-party computation. Cryptology ePrint Archive, Report 2011/272, 2011. `http://eprint.iacr.org/`.

[45] E. Kiltz and K. Pietrzak. On the security of padding-based encryption schemes - or - why we cannot prove OAEP secure in the standard model. In A. Joux, editor, *EUROCRYPT 2009*, volume 5479 of *LNCS*, pages 389–406, Cologne, Germany, Apr. 26–30, 2009. Springer, Berlin, Germany.

[46] R. Kotla, L. Alvisi, and M. Dahlin. SafeStore: A durable and practical storage system. In J. Chase and S. Seshan, editors, *Proceedings of USENIX Technical 2007*, pages 129–42. USENIX, June 2007.

[47] A. Lehmann and S. Tessaro. A modular design for hash functions: Towards making the mix-compress-mix approach practical. In M. Matsui, editor, *Advances in Cryptology — ASIACRYPT 2009*, volume 5912 of *Lecture Notes in Computer Science*, pages 364–381. Springer-Verlag, Dec. 2009.

[48] M. Liskov. Constructing an ideal hash function from weak ideal compression functions. In E. Biham and A. M. Youssef, editors, *SAC 2006*, volume 4356 of *LNCS*, pages 358–375, Montreal, Canada, Aug. 17–18, 2006. Springer, Berlin, Germany.

[49] U. Maurer and S. Tessaro. Basing PRFs on constant-query weak PRFs: Minimizing assumptions for efficient symmetric cryptography. In J. Pieprzyk, editor, *Advances in Cryptology — ASIACRYPT 2008*, volume 5350 of *Lecture Notes in Computer Science*, pages 161–178. Springer-Verlag, Dec. 2008.

[50] U. M. Maurer. Indistinguishability of random systems. In L. R. Knudsen, editor, *EUROCRYPT 2002*, volume 2332 of *LNCS*, pages 110–132, Amsterdam, The Netherlands, Apr. 28 – May 2, 2002. Springer, Berlin, Germany.

[51] U. M. Maurer, R. Renner, and C. Holenstein. Indifferentiability, impossibility results on reductions, and applications to the random oracle methodology. In M. Naor, editor, *TCC 2004*, volume 2951 of *LNCS*, pages 21–39, Cambridge, MA, USA, Feb. 19–21, 2004. Springer, Berlin, Germany.

[52] M. Naor and M. Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *22nd ACM STOC*, Baltimore, Maryland, USA, May 14–16, 1990. ACM Press.

[53] B. Pfitzmann and M. Waidner. Composition and integrity preservation of secure reactive systems. In S. Jajodia and P. Samarati, editors, *ACM CCS 00*, pages 245–254, Athens, Greece, Nov. 1–4, 2000. ACM Press.

[54] T. Ristenpart and T. Shrimpton. How to build a hash function from any collision-resistant function. In K. Kurosawa, editor, *ASIACRYPT 2007*, volume 4833 of *LNCS*, pages 147–163, Kuching, Malaysia, Dec. 2–6, 2007. Springer, Berlin, Germany.

[55] T. Ristenpart and S. Yilek. When good randomness goes bad: Virtual machine reset vulnerabilities and hedging deployed cryptography. In *Network and Distributed Systems Security – NDSS '10*. ISOC, 2010.

[56] G. Tsudik. Message authentication with one-way hash functions. In *Proceedings IEEE INFOCOM'92*, volume 3, pages 2055–2059. IEEE, 1992.

[57] F. F. Yao and Y. L. Yin. Design and analysis of password-based key derivation functions. In A. Menezes, editor, *CT-RSA*, volume 3376 of *LNCS*, pages 245–261. Springer, 2005.

A Inequivalence of CRP and Standard Hash Function Security Notions

In this section, we show that the CRP property of hash functions defined in Section 4 is inequivalent to the standard notions of hash function security. Note that both lemmas apply as well when the challenge hash is computed as $H(C \,\|\, M)$, meaning that they are relevant also to the challenge hash specified for the SafeStore system [46].

Lemma A.1 If there exists a hash function that is collision-resistant (resp., first preimage resistant, second preimage resistant) then there exists a function that is collision-resistant (resp., first preimage resistant) and not CRP.

Proof: Let $h\colon\ \{0,1\}^p \to \{0,1\}^r$ be the collision-resistant (or preimage reistant) function. Define $h\colon\ \{0,1\}^p \to \{0,1\}^s \to \{0,1\}r + s$ as

$$H(M \parallel C) = h(M) \parallel C.$$

Then H is collision resistant, first preimage resistant, or second preimage resistant on its input $(M\|C)$ whenever h is on its input M, since any collision or preimage attack on H would also give a collision or preimage attack on h, the portion of its output that is not one-to-one. But H is clearly (p, n, s)-online computable for $n = r$, the output length of h: set $st = H_1(M) = h(M)$ and $H_2(st, C) = st \parallel C$. A function that is (p, n, s)-online computable cannot be (p, n, s)-CRP.

Lemma A.2 If there exists a CRP hash function then there exists a CRP hash function that is not collision-resistant, first preimage resistant, or second preimage resistant.

Proof: Let h be (p, n, s)-CRP hash function with r-bit output, $h\colon\ \{0,1\}^p \times \{0,1\}^s \to \{0,1\}^r$. Define Define $H\colon\ \{0,1\}^p \times \{0,1\}^{s+r} \to \{0,1\}^r$ as

$$H\big(M \parallel (C \parallel C')\big) = h(M \parallel C) \oplus C'.$$

We can easily see that H is not collision-resistant, first preimage resistant, or second preimage resistant on its input (M, C, C'). For any desired output value x, pick M and C at random and set $C' \leftarrow h(M \parallel C) \oplus x$; now $H\big(M \parallel (C \parallel C')\big) = x$. This procedure computes (random) first preimages and can trivially be used to compute second preimages and collisions.

But H is $(p, n, s + r)$-CRP. Suppose it were not. Then by definition there exists an adversary $(\mathcal{A}_1, \mathcal{A}_2)$ such that, for $M \leftarrow\!\!\$\ \{0,1\}^p$, $C \leftarrow\!\!\$\ \{0,1\}^s$, and $C' \leftarrow\!\!\$\ \{0,1\}^r$, $\mathcal{A}_1(M)$ outputs state st (with $|st| \le n$) and $\mathcal{A}_2\big(st, (C \parallel C')\big)$ outputs Z where $Z = H\big(M \parallel (C \parallel C')\big)$ with nonnegligible probability. Using $(\mathcal{A}_1, \mathcal{A}_2)$, we build an adversary $(\mathcal{A}'_1, \mathcal{A}'_2)$ that shows that h is not (p, n, s)-CRP, a contradiction. Algorithm $\mathcal{A}'_1$ is simply $\mathcal{A}_1$. Algorithm $\mathcal{A}'_2$, on input st and C, chooses C' at random from $\{0,1\}^r$ and outputs $\mathcal{A}_2(st, C) \oplus C'$. Then clearly The state output by $(\mathcal{A}'_1, \mathcal{A}'_2)$ is of the same length as the state output by $(\mathcal{A}_1, \mathcal{A}_2)$, and $\mathcal{A}'_2(\mathcal{A}'_1(M), C) = h(M \parallel C)$ whenever $\mathcal{A}_2\big(\mathcal{A}_1(M), (C \parallel C')\big) = H\big(M \parallel (C \parallel C')\big)$.