Careful with Composition: Limitations of the Indifferentiability Framework

Thomas Ristenpart[1], Hovav Shacham[2], and Thomas Shrimpton[3]

[1] Dept. of Computer Sciences, University of Wisconsin–Madison, USA
rist@cs.wisc.edu
[2] Dept. of Computer Science & Engineering, UC San Diego, USA
hovav@cs.ucsd.edu
[3] Dept. of Computer Science, Portland State University, USA
teshrim@cs.pdx.edu

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 from [27] applies to *any* cryptosystem. We characterize the uncovered limitations of indifferentiability 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, and more. We formalize a stronger notion, reset indifferentiability, that enables a composition theorem covering such multi-stage security notions, but our results show that practical hash constructions cannot be reset indifferentiable. We finish by giving direct security proofs for several important PKE schemes.

1 Introduction

The indifferentiability framework of Maurer, Renner, and Holenstein (MRH) [27] supports modular proofs of security for cryptosystems. A crucial application of the framework has been to allow proofs in the random oracle model (ROM) [8] 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.

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.

K.G. Paterson (Ed.): Eurocrypt 2011, LNCS 6632, pp. 487–506, 2011.

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 [8]; 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 [32]. 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 [18]. 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. [15] 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., [1, 7, 10, 14, 16, 17, 21]), 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.

Hash-based storage auditing. We now describe an application that shows this consensus opinion to be wrong. In the design of secure distributed systems, the following important 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 [2, 22]. 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 [23]. Consider the following protocol. The client sends a random challenge C to the server; the server proves possession of the file M by computing $Z \leftarrow \mathsf{Hash}(M \,\|\, C)$ 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 [15]. 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., [1, 7, 10, 14, 16, 21]). 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. [15] 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 [19] for encryption schemes and unforgeability under chosen message attacks [20] 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).

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 [23]. But the flaw this example uncovers is that the common interpretation of composition actually *encourages* 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 multistage 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 danger is widespread. The recent security notions for deterministic [3, 5, 12], hedged [4], and efficiently

searchable [3] public-key encryption (PKE) are all multi-stage. When formalizing password-based cryptography (e.g. [6, 33]) to allow arbitrary, hash-dependent password sampling algorithms, one uses multi-stage games. A recently proposed hash function nonmalleability security notion [11] 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.

So, we cannot generically use indifferentiability-based 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.

Yet we prove that such proofs do not exist. Namely, we show in Section 5 that indifferentiability does not imply security in the multi-stage settings mentioned above.

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. However, it is easy to see that reset indifferentiability is a strong property. While constructions that only require stateless, deterministic simulators can be easily shown to achieve reset indifferentiability, it is unclear if any non-trivial constructions requiring randomized, stateful simulators can meet it. Moreover, there is clear intuition that typical hash constructions are unlikely to be reset indifferentiable — they have the property of online computability. Still, that leaves open if other efficient constructions perform better. We answer this question in the negative, proving that a wide class of single-pass hash function domain extension constructions cannot be shown reset indifferentiable. 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 the MRH 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 [17]. The CDA security notion [4] 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 [3, 5, 12], hedged PKE (which provides message privacy even in the face of poor randomness) [4, 31], and efficiently searchable encryption (an extension of deterministic PKE) [3]. 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.

Other limitations. In the course of understanding the hash-based auditing counter-example, we uncovered other subtle ways in which composition may fail to help one establish security; a discussion of these appears in the full version [29].

Universal composability. Our results have analogous repercussions for composition frameworks similar to indifferentiability, such as universal composability [13]. We discuss other frameworks in the full version [29].

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 includes many very important ones, and even after our results indifferentiability-based composition remains an elegant way to analyze security for these cases. Instead, we stress that one must be careful when using composition to perform a security analysis, ensuring that it does in fact apply as expected.

2 Preliminaries

A code-based games framework. We formalize a version of the code-based games framework of Bellare and Rogaway [9] 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). Here we give only the most important details, deferring others to the full version of this paper. 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 unspecified 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 $Q_1, Q_2, \ldots, Q_k$. Procedures P_1 and P_2 are said to *export the*

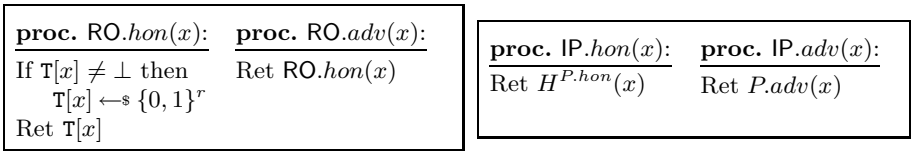

Fig. 1. Procedures implementing the functionality of the random oracle model (ROM) **(left)** and the ideal primitive model (IPM) **(right)**. The number r is set as appropriate for a given context.

same interface if their inputs and outputs agree in number and type. This will typically be clear from context.

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.)

Variables are implicitly set initially to default values, i.e. integer variables are set to 0, arrays are everywhere $\perp$, etc. 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.

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 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

$G^{F,\mathcal{A}_1,\ldots,\mathcal{A}_m}$. We fix the convention that the main and specified procedures of G can call $F.hon$ and $\mathcal{A}_1,\ldots,\mathcal{A}_m$ (but may not call $F.adv$) while the adversarial procedures may call $F.adv$ (but may not call $F.hon$). Thus $F.adv$ is the adversarial interface of F, and $F.hon$ is the honest interface. 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 4.

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.

3 Indifferentiability Framework for Single-Stage Games

We describe the indifferentiablity framework [27] using games, unlike prior treatments that used random systems [26, 27] or interactive Turing machines [15]. We feel that using explicit code-based games makes understanding the limitations of indifferentiability easier, because it will enable expressing these limitations 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 [27] 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

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

Fig. 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.

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 [15]: 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 [27] 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 [27] notion as weak indifferentiability and to the [15] notion as strong indifferentiability. We will focus on strong indifferentiability here since it implies weak.

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 [27].

Theorem 1. *Let G be a game expecting access to a functionality and a single adversarial procedure. 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: $t_{\mathcal{B}} \leq t_{\mathcal{A}} + q_{\mathcal{A}} \cdot t_{\mathcal{S}}$, $q_{\mathcal{B}} \leq q_{\mathcal{A}} \cdot q_{\mathcal{S}}$, $t_{\mathcal{D}} \leq t_G + q_{G,1} \cdot t_{\mathcal{A}}$, and $q_{\mathcal{D}} \leq q_{G,0} + q_{G,1} \cdot q_{\mathcal{A}}$, 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 1 is readily established by adapting the proof of [27, 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

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

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

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

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. ∎

Single-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 expects access to adversarial procedures $\mathcal{A}_1, \ldots, \mathcal{A}_m$ and now consider generalizing Theorem 1 to account for G. Recall that these adversarial procedures do not 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:

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

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.

All this means that indifferentiability-based composition can only apply to security notions defined via single-stage games, which we now define. Consider a game that has m procedures. We say that a game is *stage minimal* if all games G' that are equivalent to G use the same number of adversarial procedures. We now restrict attention to stage minimal games. Then, an m-stage game is one that has m stages. A single-stage game is one for which $m = 1$ and a multi-stage game is one for which $m > 1$. Let $\mathcal{SG}$ be the set of all single-stage games. Note that $\mathcal{SG}$ includes the games defining indifferentiability above, the classic notions of encryption security such as IND-CPA [19] or IND-CCA [28], unforgeability under chosen message attack UF-CMA [20], and many others.

If $\mathcal{G}$ is the set of all games, then we let $\mathcal{MG} = \mathcal{G} \backslash \mathcal{SG}$ be the set of games that are not single stage. We call any game in $\mathcal{MG}$ a *multi-stage game*. Examples of multi-stage games include chosen distribution attack security for public-key encryption [4] (see Figure 4), non-malleability of hash functions [11], password-based key exchange [6], and others.

Discussion. A game that uses multiple adversarial procedures, but is equivalent to a game with a single adversarial procedure, is not considered multi-stage by our definition above. Many experiments are formalized with multiple adversarial procedures, but the game forwards arbitrary adversarial state from one procedure to the next. It is clear such games are actually equivalent to one with a single adversarial procedure. Some games allow a small amount of state to be passed directly from one adversarial procedure to the next. 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 cannot be written with a single adversarial procedure.

We do note, however, that we may extend Theorem 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.

Note that there are other subtleties of composition that might lead to erroneous beliefs and claims. We provide a detailed discussion of these in the full version.

4 A Practically Motivated Counterexample

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 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 [23]. 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.

The following theorem shows that, as expected, a random oracle possesses property CRP. The proof appears in the full version.

$$\begin{array}{|l}
\hline
\textbf{main } \mathrm{CRP}_{p,n,s}^{F,\mathcal{A}_1,\mathcal{A}_2} \\
\hline
M \leftarrow\!\!\$\ \{0,1\}^p \\
st \leftarrow\!\!\$\ \mathcal{A}_1^{F.adv}(M) \\
\text{If } |st| > n \text{ then Ret } \mathsf{false} \\
C \leftarrow\!\!\$\ \{0,1\}^s \\
Z \leftarrow\!\!\$\ \mathcal{A}_2^{F.adv}(st, C) \\
\text{Ret } \big(Z = F.hon(M \parallel C)\big) \\
\hline
\end{array}$$

Fig. 3. Game capturing our challenge-response hash function property

Theorem 2. *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*

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

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

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 $H^f(M_1 \parallel M_2) = H_2^f(H_1^f(M_1), M_2))$ 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 [15]. 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 $M = M_1 \parallel \ldots \parallel M_b$, for each $i \in \{1, \ldots, b\}$ compute $V_i = f(V_{i-1} \parallel M_i)$, 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 [7], MDP [21], the Chop and so-called HMAC constructions [15], 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 $H^f(N \parallel M)$. This construction does indeed have CRP (though one cannot use composition to establish it). The point is that indifferentiability appears to imply that $N \parallel M$ and $M \parallel N$ are equivalently secure. Given the widespread use of hash functions as random oracles in practice (implicitly or explicitly), we must be careful to assess each application's security starting from the ideal primitive underneath the hash function and only use indifferentiability-based composition when it is truly applicable.

5 Indifferentiability Fails for 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 appears in the full version.

Theorem 3. *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 3 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. Note that all these are feasible in the ROM.

Example: CDA security for public-key encryption. Public-key encryption (PKE) and the chosen-distribution attack (CDA) security goal are defined in Section 7. CDA generalizes deterministic PKE security notions [3, 5, 12], and CDA-secure PKE is useful for efficiently search over encrypted data [3] and defense-in-depth against randomness failures [31]. 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.

In the full version, we give analogous results for nonmalleability of hash functions [11], password-based authenticated key exchange [6], and others. Interestingly, the hash function nonmalleability notion is the only formal notion that captures resistance to length-extension attacks. This is especially troubling because provable resistance to length extension attacks was a primary motivation for building indifferentiable hash constructions [15].

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.

6 Indifferentiability with Simulator Resets

We initiate the exploration of strengthenings of indifferentiability that support composition for 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 stronger notions.

We propose a strengthening of indifferentiability, called reset indifferentiability, that immediately admits a composition theorem covering multi-stage games.

Reset indifferentiability. We define a version of indifferentiability that requires simulators 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 that 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 4. *Let $G \in \mathcal{G}$. 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*

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

Moreover: $t_{\mathcal{B}_i} \leq t_{\mathcal{A}_i} + q_{\mathcal{A}_i} t_{\mathcal{S}}$, $q_{\mathcal{B}_i} \leq q_{\mathcal{A}_i} \cdot q_{\mathcal{S}}$, $t_{\mathcal{D}} \leq m + t_G + \sum_{i=1}^{m} q_{G,i} \cdot t_{\mathcal{A}_i}$, and $q_{\mathcal{D}} \leq q_{G,0} + \sum_{i=1}^{m} q_{G,i} \cdot q_{\mathcal{A}_i}$, 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 1. For $1 \leq i \leq m$, 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.

Reset indifferentiability can be achieved when one establishes (conventional) indifferentiability using a stateless and deterministic simulator. This is because it is clear resetting such a simulator does not affect its subsequent behavior. Unfortunately it seems challenging to achieve reset indifferentiability for all but trivial constructions, and we will show negative results for efficient constructions below. We leave finding non-trivial constructions, even inefficient ones, as an open question.

On practical domain extension under resets. 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 answer this question in the negative, ruling out a large class of "efficient" constructions from being reset indifferentiable from a RO.

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 $H^P(\langle M_1, M_2\rangle) = H_2^P(H_1^P(M_1) \parallel M_2)$ 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 $\langle M_1, M_2\rangle = M_1 \parallel M_2$. 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 [25] (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. Its proof appears in the full version.

Theorem 5. *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,*

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

7 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 [3, 4, 31]. Of course we cannot rely on Theorem 1, so our proof is done directly in the ideal primitive model. Nevertheless, our main result covers a broad mix of PKE and hash functions.

We focus on the hash construction from [17], 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 [15], MCM [30], NIRP [24], and various SHA-3 competitors.

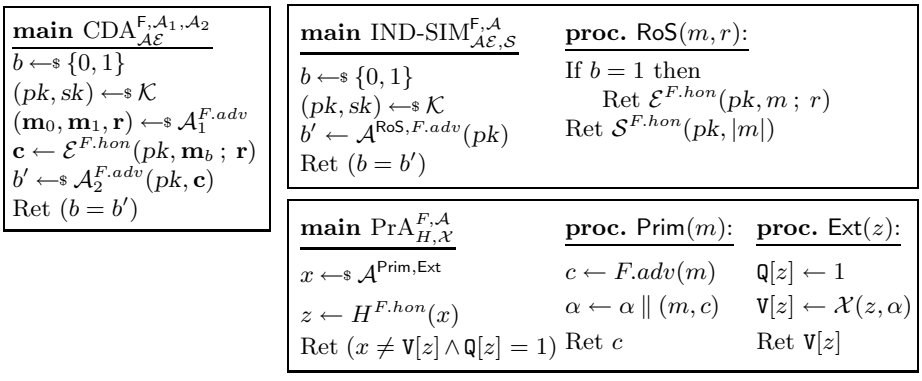

Fig. 4. (Left) The non-adaptive CDA game. **(Right)** The IND-SIM and PrA games

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 [3], we define for any scheme $\mathcal{AE}$ the maximum public-key collision probability by $\mathsf{maxpk}_{\mathcal{AE}} = \max_{w \in \{0,1\}^*} \Pr[\, pk = w \ :\ (pk, sk) \leftarrow\$ \mathcal{K}\,]$.

CDA security. In Figure 4 we detail the security game for (non-adaptive) chosen-distribution attacks [4]. 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

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

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 an 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 [17] generalizes collision resistance to include extractability. Game PrA is defined in Figure 4. 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 $F.adv$. Thus Theorems 1 and 4 do not apply when $G = \text{PrA}$. 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 6, 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 $\text{IND-SIM}_{\mathcal{AE},\mathcal{S}}$ is shown in Figure 4. 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 the full version 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 6 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 [3, 4, 31] 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. The full proof appears in the full version.

Theorem 6. *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.*

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

$\mathcal{B}$ 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 $(\mathcal{A}_1, \mathcal{A}_2)$. □

Acknowledgements

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, Daniele Micciancio, and Moni Naor for helpful discussions about this work.

References

1. Andreeva, E., Mennink, B., Preneel, B.: On the indifferentiability of the grøstl hash function. In: Garay, J.A., Prisco, R.D. (eds.) SCN 2010. LNCS, vol. 6280, pp. 88–105. Springer, Heidelberg (2010)
2. Ateniese, G., Burns, R., Curtmola, R., Herring, J., Kissner, L., Peterson, Z., Song, D.: Provable data possession at untrusted stores. In: De Capitani di Vimercati, S., Syverson, P. (eds.) Proceedings of CCS 2007, pp. 598–609. ACM Press, New York (October 2007)
3. Bellare, M., Boldyreva, A., O'Neill, A.: Deterministic and efficiently searchable encryption. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 535–552. Springer, Heidelberg (2007)
4. Bellare, M., Brakerski, Z., Naor, M., Ristenpart, T., Segev, G., Shacham, H., Yilek, S.: Hedged public-key encryption: How to protect against bad randomness. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 232–249. Springer, Heidelberg (2009)
5. Bellare, M., Fischlin, M., O'Neill, A., Ristenpart, T.: Deterministic encryption: Definitional equivalences and constructions without random oracles. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 360–378. Springer, Heidelberg (2008)
6. Bellare, M., Pointcheval, D., Rogaway, P.: Authenticated key exchange secure against dictionary attacks. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 139–155. Springer, Heidelberg (2000)
7. Bellare, M., Ristenpart, T.: Multi-property-preserving hash domain extension and the EMD transform. In: Lai, X., Chen, K. (eds.) ASIACRYPT 2006. LNCS, vol. 4284, pp. 299–314. Springer, Heidelberg (2006)

8. Bellare, M., Rogaway, P.: Random oracles are practical: A paradigm for designing efficient protocols. In: Ashby, V. (ed.) ACM CCS 1993, Fairfax, Virginia, USA, November 3-5, pp. 62–73. ACM Press, New York (1993)
9. Bellare, M., Rogaway, P.: The security of triple encryption and a framework for code-based game-playing proofs. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 409–426. Springer, Heidelberg (2006)
10. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: On the indifferentiability of the sponge construction. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 181–197. Springer, Heidelberg (2008)
11. Boldyreva, A., Cash, D., Fischlin, M., Warinschi, B.: Foundations of non-malleable hash and one-way functions. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 524–541. Springer, Heidelberg (2009)
12. Boldyreva, A., Fehr, S., O'Neill, A.: On notions of security for deterministic encryption, and efficient constructions without random oracles. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 335–359. Springer, Heidelberg (2008)
13. Canetti, R.: Universally composable security: A new paradigm for cryptographic protocols. In: 42nd FOCS, Las Vegas, Nevada, USA, October 14-17, pp. 136–145. IEEE Computer Society Press, Los Alamitos (2001)
14. Chang, D., Nandi, M.: Improved indifferentiability security analysis of chopMD hash function. In: Nyberg, K. (ed.) FSE 2008. LNCS, vol. 5086, pp. 429–443. Springer, Heidelberg (2008)
15. Coron, J.-S., Dodis, Y., Malinaud, C., Puniya, P.: Merkle-damgård revisited: How to construct a hash function. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 430–448. Springer, Heidelberg (2005)
16. Dodis, Y., Reyzin, L., Rivest, R.L., Shen, E.: Indifferentiability of permutation-based compression functions and tree-based modes of operation, with applications to MD6. In: Dunkelman, O. (ed.) FSE 2009. LNCS, vol. 5665, pp. 104–121. Springer, Heidelberg (2009)
17. Dodis, Y., Ristenpart, T., Shrimpton, T.: Salvaging merkle-damgård for practical applications. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 371–388. Springer, Heidelberg (2009)
18. Franks, J., Hallam-Baker, P., Hostetler, J., Leach, P., Luotonen, A., Sink, E., Stewart, L.: An Extension to HTTP: Digest Access Authentication. RFC 2069 (Proposed Standard) (January 1997), Obsoleted by RFC 2617
19. Goldwasser, S., Micali, S.: Probabilistic encryption. Journal of Computer and System Sciences 28(2), 270–299 (1984)
20. Goldwasser, S., Micali, S., Rivest, R.L.: A digital signature scheme secure against adaptive chosen-message attacks. SIAM Journal on Computing 17(2), 281–308 (1988)
21. Hirose, S., Park, J.H., Yun, A.: A simple variant of the merkle-damgård scheme with a permutation. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 113–129. Springer, Heidelberg (2007)
22. Juels, A., Kaliski, B.: PORs: Proofs of retrievability for large files. In: De Capitani di Vimercati, S., Syverson, P. (eds.) Proceedings of CCS 2007, pp. 584–597. ACM Press, New York (October 2007)
23. Kotla, R., Alvisi, L., Dahlin, M.: SafeStore: A durable and practical storage system. In: Chase, J., Seshan, S. (eds.) Proceedings of USENIX Technical 2007, pp. 129–142. USENIX (June 2007)

24. Lehmann, A., Tessaro, S.: A Modular Design for Hash Functions: Towards Making the Mix-Compress-Mix Approach Practical. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 364–381. Springer, Heidelberg (2009)
25. Liskov, M.: Constructing an ideal hash function from weak ideal compression functions. In: Biham, E., Youssef, A.M. (eds.) SAC 2006. LNCS, vol. 4356, pp. 358–375. Springer, Heidelberg (2007)
26. Maurer, U.M.: Indistinguishability of random systems. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 110–132. Springer, Heidelberg (2002)
27. Maurer, U.M., Renner, R., Holenstein, C.: Indifferentiability, impossibility results on reductions, and applications to the random oracle methodology. In: Naor, M. (ed.) TCC 2004. LNCS, vol. 2951, pp. 21–39. Springer, Heidelberg (2004)
28. Naor, M., Yung, M.: Public-key cryptosystems provably secure against chosen ciphertext attacks. In: 22nd ACM STOC, Baltimore, Maryland, USA, May 14-16. ACM Press, New York (1990)
29. Ristenpart, T., Shacham, H., Shrimpton, T.: Careful with composition: Limitations of the indifferentiability framework. Full version of this paper
30. Ristenpart, T., Shrimpton, T.: How to build a hash function from any collision-resistant function. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 147–163. Springer, Heidelberg (2007)
31. Ristenpart, T., Yilek, S.: When good randomness goes bad: Virtual machine reset vulnerabilities and hedging deployed cryptography. In: Network and Distributed Systems Security – NDSS 2010. ISOC (2010)
32. Tsudik, G.: Message authentication with one-way hash functions. In: Proceedings IEEE INFOCOM 1992, vol. 3, pp. 2055–2059. IEEE, Los Alamitos (1992)
33. Yao, F.F., Yin, Y.L.: Design and analysis of password-based key derivation functions. In: Menezes, A. (ed.) CT-RSA 2005. LNCS, vol. 3376, pp. 245–261. Springer, Heidelberg (2005)