# Careful with Composition: Limitations of the Indifferentiability Framework

Summary for Research Seminar in Cryptography

Riivo Talviste

May 26, 2014

## 1 Introduction

This report summarizes the results of [12] by Ristenpart, Shacham and Shrimpton. They show that the indifferentiability composition theorem by Maurer, Renner and Holenstein [10] is often misinterpreted and used incorrectly resulting in insecure cryptographic constructions.

## 2 Preliminaries

### 2.1 Ideal primitive

Ideal primitive is defined as an algorithmic entity which receives inputs from on of the parties and immediately delivers its output to the querying party [6]. Examples of ideal primitives include the random oracle and the ideal cipher.

### 2.2 Random oracle model

Random oracle model (ROM) is an idealized model, where all participating parties have access to a public random oracle. This oracle answers all queries, specified by a key, by generating and outputting a truly random value from its output domain. Whenever a query with the same key is repeated, it answers with the same value. Pseudocode for such random oracle is shown in Figure 1(above). The random oracle model was introduced by Bellare and Rogaway [4] to connect cryptographic theory and practice. As a result, a lot of practical, in-use cryptographic schemes are proven secure in ROM and for many of them, ROM proofs are the only ones known.

### 2.3 Notation

When a value is uniformly randomly sampled from a finite non-empty set $\mathcal{X}$, we denote it as $x \xleftarrow{\$} \mathcal{X}$, where $x$ is the variable that is assigned the random

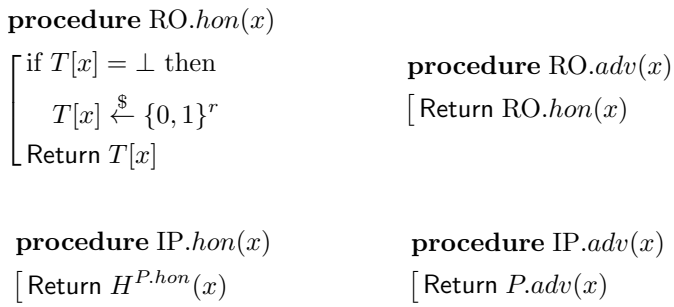


Figure 1: Procedures implementing the functionality of the random oracle model (above) and the ideal primitive model (below). The number $r$ is set by the context.

value. However, if $\mathcal{A}$ is an algorithm, $x \xleftarrow{\$} \mathcal{A}$ denotes that $x$ is assigned a value according to the distribution induced by $\mathcal{A}$.

For describing cryptographic games, we use a version of the code-based games framework introduced by Bellare and Rogaway [5]. A *procedure* is a sequence of statements with zero or more inputs and zero or more outputs. An *unspecified procedure* is a procedure whose statements, inputs and outputs are understood from the context. This is generally used for adversarial procedures. A procedure can itself also call other procedures. If a procedure $P$ expects to call $k$ other procedures, we denote it as $P^{Q_1,\ldots,Q_k}$ to show that the called procedures are handled by $Q_1$ through $Q_k$. In such case, we assume that there are no syntactic mismatches between the calls and all the necessary inputs and outputs correspond to each other. Also, all procedures are assumed to halt and for simplicity, we say that each statement runs in unit time.

Variables are by default local and maintain their state between the calls of the procedure. Collections of procedures, denoted $P = (P.x, P.y)$, share their variables and the suffixes $x$ and $y$ (in this case) are called *interfaces* of $P$. Procedures are said to *export the same interface* if they agree on number and types for inputs and outputs.

Procedure **main** is a special procedure that has no inputs and has some output. It cannot be called by other procedures and it can access all variables of other specified procedures.

A functionality $F$ is a collection of procedures, usually with *honest* and *adversarial* interfaces: $F = (F.hon, F.adv)$. For example, we give two functionalities showing two computation models on Figure 1. $\mathrm{RO} = (\mathrm{RO}.hon, \mathrm{RO}.adv)$ implements random oracle and $\mathrm{IP} = (\mathrm{IP}.hon, \mathrm{IP}.adv)$ shows an (ideal) primitive model. In the latter case let $P = (P.hon, P.adv)$ implement some ideal primitive that underlies some construction $H$.

A game $G$ is denoted by "**main** $G$". It can make a use of a functionality $F$ and any number of adversarial procedures $\mathcal{A}_1, \ldots, \mathcal{A}_m$ (together referred as

*adversary*), denoted by $G^{F,\mathcal{A}_1,\ldots,\mathcal{A}_m}$. Running a game means sequentially executing all statements in its **main** procedure and the output of game $G$ is the value returned by its **main** procedure. We denote by $G^{F,\mathcal{A}_1,\ldots,\mathcal{A}_m} \Rightarrow y$ a situation when game $G$ outputs value $y$. Finally, for any fixed functionality $F$ and adversary $\mathcal{A}_1, \ldots, \mathcal{A}_m$ we say that 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 $y$.

### 2.4 Single and multi-stage games

In the following we will consider games where only adversarial procedures call *F.adv*, i.e. it is not called by the main or any other specified procedure of the game. Then we say that a game is *single-stage* game if it uses only a single adversarial procedure. Games that use more then one adversarial procedures are called *multi-stage*.

## 3 The Indifferentiability Framework

### 3.1 Indifferentiability

Let us fix two functionalities $F_1$ and $F_2$, for example, standing for (some) ideal primitive and random oracle, respectively. As usual we have to environments – Real and Ideal (see Figure 2) – and an adversary that tries to guess in which environment we currently are. In this case we call the adversary a distinguisher $\mathcal{D}$. The indifferentiability advantage of the distinguisher $\mathcal{D}$ is defined as

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

where $\mathcal{S}$ is a simulator and $y$ is an arbitrary fixed value.

**Definition 1.** *$F_1$ is indifferentiable from $F_2$ if there exists a polynomial-time simulator $\mathcal{S}$ such that for any polynomial-time $\mathcal{D}$ the advantage $\mathsf{Adv}^{\text{indiff}}$ is negligible in the security parameter.*

In the context of this paper the goal of indifferentiability is to allow translating security analysis of a cryptographic scheme from one model of computation to another to be easily accomplished. The following theorem from [10] (and reiterated in [12]) enables this.

**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 exists an adversary $\mathcal{B}$ and a distinguisher $\mathcal{D}$ such that for all values $y$*

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

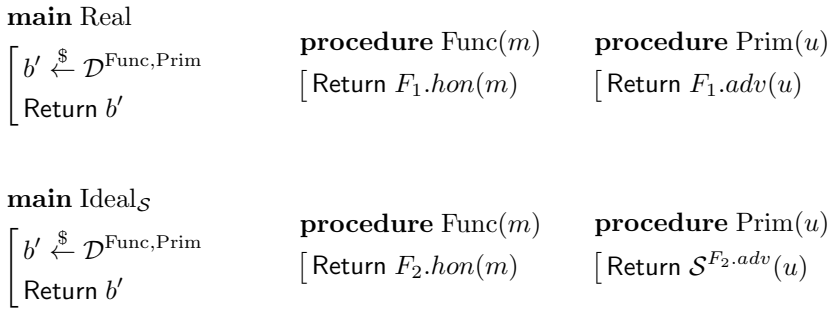


Figure 2: Games defining indifferentiability. Adversary $\mathcal{D}$ and functionalities $F_1$ and $F_2$ are unspecified. $\mathcal{S}$ is a simulator of the game.

*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 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 and to the adversarial procedures, respectively.*

*Proof.* Fix any value $y$. Let $F = (F.hon, F.adv)$ be some unspecified functionality that exports the same interface as $(F_1.hon, F_2.adv)$. Let the indifferentiability adversary $\mathcal{D}$ be defined as follows. Adversary $\mathcal{D}$ runs game $G$ and 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 $\mathcal{A}$. At the end $\mathcal{D}$ outputs whatever $G$ outputs. Then in the case that $F = F_1$ we get

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

$q_{\mathcal{D}} \leq q_{G,0} + q_{G,1} \cdot q_{\mathcal{A}}$ and $t_{\mathcal{D}} \leq t_G + q_{G,1} \cdot t_{\mathcal{A}}$. Now let's define adversary $\mathcal{B}$ for $F = F_2$. Adversary $\mathcal{B}$ runs $\mathcal{A}$ and whenever $\mathcal{A}$ queries its oracle, adversary $\mathcal{B}$ runs $\mathcal{S}$ using its $F_2.adv$ oracle to answer any queries that $\mathcal{S}$ makes. Adversary $\mathcal{B}$ outputs whatever $\mathcal{A}$ outputs. By construction we get

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

with $q_{\mathcal{B}} \leq q_{\mathcal{A}} \cdot q_{\mathcal{S}}$ and $t_{\mathcal{B}} \leq t_{\mathcal{A}} + q_{\mathcal{A}} \cdot t_{\mathcal{S}}$. By substituting equations 1 and 2 into the definition of indifferentiability advantage we derive the advantage relation of the theorem. □

Theorem 1 explicitly considers only single-stage games. However, it seems that this is taken as an artificial restriction and recently, the theorem has been used to "prove" that many cryptographic constructions are indifferentiable from ROM. It is taken for granted that a game $G$ with access to a single adversarial

$$
\begin{array}{l}
\textbf{main } CRP_{p,n,s}^{F,\mathcal{A}_1,\mathcal{A}_2} \\
\left[\begin{array}{l}
M \xleftarrow{\$} \{0,1\}^p \\
st \xleftarrow{\$} \mathcal{A}_1^{F.adv}(M) \\
\text{if } |st| > n \text{ then } \mathsf{Return\ false} \\
C \xleftarrow{\$} \{0,1\}^s \\
Z \xleftarrow{\$} \mathcal{A}_2^{F.adv}(st, C) \\
\mathsf{Return}\ [Z = F.hon(M \parallel C)]
\end{array}\right.
\end{array}
$$

Figure 3: Game showing the challenge-response hash function property.

procedure $\mathcal{A}$ used in the theorem could be easily substituted with another game that requires access to multiple adversarial procedures $\mathcal{A}_1, \ldots, \mathcal{A}_m$ by simply replacing the adversary $\mathcal{B}$ with a set of adversaries $\mathcal{B}_1, \ldots, \mathcal{B}_m$ such that $\mathcal{B}_i = \mathcal{A}_i^S$ for $i = 1..m$. However, on a closer look it comes out that the proof of Theorem 1 would actually fail in this case as the simulator $\mathcal{S}$ cannot maintain its state between the $m$ invocations and thus the analogue of equation 2 does not hold:

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

# 4 Indifferentiability Fails for Multi-Stage Games

## 4.1 Practically motivated counterexample

To demonstrate how Theorem 1 fails for multi-stage games, the authors of [12] give a rather simple counterexample. Later, they go on by generalizing this to all multi-stage games.

The counterexample is motivated by the growing popularity of cloud-based storage. The question is how a client can be sure that his or her file is really stored by the cloud (or any other server for that matter) [1, 8]? A malicious server could just delete the file received from the user to save its disk space. The example presented in this paper is inspired by the proof-of-storage challenge-response protocol proposed by a system called SafeStore [9]. The protocol, called CRP, is explained in the following and also given as a game in Figure 3.

Let $M$ be a message (file) that the client wants to store this on a server. The client can check if the server has stored the message by sending it a random challenge $C$ to which the server is supposed to answer with $H(M \parallel C)$, where $H$ is a hash function. The client can then compare server's response with the hash value computed from the local copy of the message. If $H$ is a random oracle then there is no way the server can cheat other that guessing the challenge $C$ in advance.

Next, we will show how such construction fails if $H$ is built from an ideal primitive. For that, we will first define a new property of hash functions, called *online computability*:

**Definition 2.** *Consider a hash function $H^f : \{0,1\}^* \to \{0,1\}^r$ using some underlying primitive $f$. Then we say that $H^f$ is $(p, n, s)$-online computable if for $p, n, s > 0$ there exists functions $H_1^f : \{0,1\}^p \to \{0,1\}^n$ and $H_2^f : \{0,1\}^n \times \{0,1\}^s \to \{0,1\}^r$ 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, it is required 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 other words, what online computability means is that the hash function can be computed in two stages, processing $M_1$ and $M_2$ sequentially. Let us construct the counterexample by using a hash function construction that is proven to be indifferentiable from RO in [6]. Assume that both the message $M$ and challenge $C$ in CRP game are $d$ bits long $(p, s = d)$ and hash function is $H^f$, where $f$ is an ideal compression function that outputs $n$-bit strings, such that $n < d$. By construction, $H$ returns the first $r = n/2$ bits of $f(f(IV, M), C)$, where $IV$ is an initialization vector (constant string). It is easy to see that the adversary always wins the CRP game shown on Figure 3 if $\mathcal{A}_1$ outputs $st = H_1^f(M) = f(IV, M)$ and $\mathcal{A}_2$ outputs $H_2^f(st, C) = f(st, C)$. Simply put, the storage server can just store the (smaller) hash of the message and discard the message $M$ itself, while still being able to answer future challenge queries successfully.

Most iterative hash function constructions are in fact online computable, including HMAC and NMAC constructions [6], EMD [3], MDP [7] and many SHA-3 competition candidates. None of these or any other $(p, n, s)$-online computable hash function constructions can be used in the CRP game.

## 4.2 Indifferentiability fails for all multi-stage games

Following the example of the CRP game, we will now show how indifferentiability does not imply security for any multi-stage game. We will show how any ideal primitive can be augmented to include a storage interface, such that all of the indifferentiability related results stay unaffected. The interface itself allows an adversary to save key-value pairs and retrieve values by key. Formally, let $F_1$ be a functionality and $St$ be a procedure that exposes hash table $T$. That is, on input $(X, Y)$ it sets $T[X] \leftarrow Y$ and returns nothing; and on input $(X, \perp)$ it returns $T[X]$ if it is set and $\perp$ otherwise. Now, the storage-augmented functionality $F_1^* = (F_1.hon, F_1^*.adv)$ has the same honest interface as $F_1$ but the adversarial interface additionally exposes $St$: $F_1^*.adv = (F_1.adv, St)$.

The following theorem states that if functionality $F_1$ is indifferentiable from some functionality $F_2$ then $F_1^*$ is also indifferentiable from $F_2$.

**Theorem 2.** *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*

$$
\begin{array}{l}
\textbf{main } CDA^{F,\mathcal{A}_1,\mathcal{A}_2}_{\mathrm{PKE}} \\
\left\lceil
\begin{array}{l}
b \xleftarrow{\$} \{0,1\} \\
(pk, sk) \xleftarrow{\$} \mathcal{K} \\
(m_0, m_1, r) \xleftarrow{\$} \mathcal{A}_1^{F.adv} \\
c \leftarrow \mathcal{E}^{F.hon}(pk, m_b; r) \\
b' \xleftarrow{\$} \mathcal{A}_2^{F.adv}(pk, c) \\
\mathsf{Return}\ [b = b']
\end{array}
\right.
\end{array}
$$

Figure 4: The non-adaptive CDA game.

*that for all distinguishers $\mathcal{A}$ there exists a distinguisher $\mathcal{B}$ such that*

$$\mathsf{Adv}^{\mathrm{indiff}}_{F_1^*,F_2,\mathcal{S}_\mathcal{A}}(\mathcal{A}) = \mathsf{Adv}^{\mathrm{indiff}}_{F_1,F_2,\mathcal{S}_\mathcal{B}}(\mathcal{B}).$$

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

The proof of this theorem is omitted here and available in the full version of the original paper [11]. The rationale behind the theorem is that distinguishers in indifferentiability maintain their state throughout the game and it does not matter if they store it locally or export it to an oracle as is the case with storage-augmented functionalities. However, the ability to use storage oracle breaks security for many multi-stage games, for example chosen-distribution attack for public-key cryptography, non-malleable hashing, password-based cryptography, key-dependent message security and related-key attack. Full version of the original paper [11] gives adversary constructions for all of the mentioned security goals. Here, we will briefly describe how resistance against chosen-distribution attack (CDA) of public-key encryption (PKE) is not achievable in the storage-augmented primitive model.

Chosen-distribution attack security [2] captures the security of a PKE scheme when the randomness $r$ may not be a (sufficiently long) string of uniform bits. Figure 4 shows the non-adaptive CDA game, where $\mathcal{K}$ is a key generation algorithm and $\mathcal{E}$ is a encryption algorithm of a PKE scheme. It is easy to see that if we replace the functionality $F$ with its storage-augmented counterpart $F^*$ then the adversary should play as follows: $\mathcal{A}_1$ picks $(m_0, m_1, r)$ randomly as in $F$ but also queries $St(0, (m_0, m_1, r))$. Later $\mathcal{A}_2$ queries $St(0, \perp)$ to retrieve the same triple, encrypts both messages $m_0$ and $m_1$ under $r$, compares both ciphertexts with the challenge and outputs the correct bit. Such adversary always wins.

# 5 Reset Indifferentiability

After showing that indifferentiability as stated in [10] fails for all multi-stage games, the authors of [12] continue by introducing a stronger notion called *reset indifferentiability* that also takes into account multi-stage games.

Reset indifferentiability uses simulator resets where the state of simulator is reset to its initial value in the beginning of each stage. In this case the simulator is expressed as a pair of procedures $\overset{\circlearrowleft}{\mathcal{S}} = (\overset{\circlearrowleft}{\mathcal{S}}.\mathcal{S}, \overset{\circlearrowleft}{\mathcal{S}}.Rst)$, where $\overset{\circlearrowleft}{\mathcal{S}}.\mathcal{S}$ is the same as original $\mathcal{S}$ and $\overset{\circlearrowleft}{\mathcal{S}}.Rst$ is a procedure that reinitializes all internal variables of $\overset{\circlearrowleft}{\mathcal{S}}.\mathcal{S}$ to their initial values. Similarly, we have a new notation for functionality $\vec{F} = (\vec{F}.hon, \vec{F}.adv) = (F.hon, (F.adv, nop))$, where *nop* is a special "no operation" functionality that takes no input and does nothing. Depending on the context, either $\overset{\circlearrowleft}{\mathcal{S}}.Rst$ and *nop* operation is called by Prim procedure shown on Figure 2. Let $F_1$ and $F_2$ be functionalities, $\mathcal{D}$ an adversary (the distinguisher) that outputs a bit and $\mathcal{S}$ a simulator. Then we define the reset indifferentiability advantage of $\mathcal{D}$ as

$$\mathsf{Adv}^{\mathrm{reset-indiff}}_{F_1,F_2,\mathcal{S}} = \Pr\left[\mathrm{Real}^{\vec{F}_1,\mathcal{D}} \Rightarrow y\right] - \Pr\left[\mathrm{Ideal}^{\vec{F}_2,\mathcal{D}}_{\overset{\circlearrowleft}{\mathcal{S}}} \Rightarrow y\right],$$

where $y$ is an arbitrary fixed value.

The following theorem is an analogue of Theorem 1 that uses reset indifferentiability.

**Theorem 3.** *Let $G$ be a game and let $F_1$, $F_2$ be two 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 exists an adversary $\mathcal{B}_1, \ldots, \mathcal{B}_m$ and a distinguisher $\mathcal{D}$ such that for all values $y$*

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

*Moreover: $t_{\mathcal{B}_i} \leq t_{\mathcal{A}_i} + q_{\mathcal{A}_i} \cdot t_{\mathcal{S}}$, $q_{\mathcal{B}_i} \leq q_{\mathcal{A}_i} \cdot q_{\mathcal{S}}$, $t_{\mathcal{D}} \leq m + t_G + \sum_{i=1}^{m} q_{G,i} \cdot t_{\mathcal{A}_i}$ 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 this theorem is the same as for Theorem 1 with small variations to count for adversary with multiple procedures. First, as separate instance of $\mathcal{S}$ has to be used for each procedure $\mathcal{B}_i$: $\mathcal{B}_i^{F_2.adv} = \mathcal{A}_i^{\mathcal{S}^{F_2.adv}}$ $(1 \leq i \leq m)$. Secondly, define distinguisher $\mathcal{D}$ by modifying $\mathcal{D}^{F.hon,F.adv} = G^{F,\mathcal{A}_1,\ldots,\mathcal{A}_m}$ so that each $\mathcal{A}_i$ call is immediately preceded by a reset call.

## 5.1 Practical implications

Using reset indifferentiability is in general the same as using (conventional) indifferentiability with stateless and deterministic simulator as in this case the

simulator's behavior is not affected by resets. Keeping that in mind, it is questionable if the stringer reset indifferentiability can be achieved for any non-trivial constructions. Moreover, in [12] a large class of efficient hash constructions is shown to not achieve the reset indifferentiability property.

# 6 Conclusion

The authors of [12] draw attention to a fact that the main theorem in [10] is in many cases loosely interpreted resulting in cryptographic protocol "proofs" where the theorem is in fact not applicable. The authors show that the theorem works only for single-stage games and fails for all multi-stage games. They go on by introducing a stronger notion of indifferentiability, called *reset indifferentiability*, and an analogue of the theorem in question that takes into account multi-stage games. Unfortunately, a large class of efficient hash functions turns out to be not reset indifferentiable, leaving the existence of any non-trivial reset indifferentiable constructions in question.

# References


[1] Giuseppe Ateniese, Randal Burns, Reza Curtmola, Joseph Herring, Lea Kissner, Zachary Peterson, and Dawn Song. Provable data possession at untrusted stores. IACR Cryptology ePrint Archive, Report 2007/202, 2007.

[2] Mihir Bellare, Zvika Brakerski, Moni Naor, Thomas Ristenpart, Gil Segev, Hovav Shacham, and Scott Yilek. Hedged public-key encryption: How to protect against bad randomness. In *Advances in Cryptology ASIACRYPT 2009*, volume 5912 of *Lecture Notes in Computer Science*, pages 232–249. Springer Berlin Heidelberg, 2009.

[3] Mihir Bellare and Thomas Ristenpart. Multi-property-preserving hash domain extension and the emd transform. In *Advances in Cryptology ASIACRYPT 2006*, volume 4284 of *Lecture Notes in Computer Science*, pages 299–314. Springer Berlin Heidelberg, 2006.

[4] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *Proceedings of the 1st ACM Conference on Computer and Communications Security*, CCS '93, pages 62–73, New York, NY, USA, 1993. ACM.

[5] Mihir Bellare and Phillip Rogaway. The security of triple encryption and a frameworkforcode-basedgame-playingproofs. In *Advances in Cryptology - EUROCRYPT 2006*, volume 4004 of *Lecture Notes in Computer Science*, pages 409–426. Springer Berlin Heidelberg, 2006.

[6] Jean-Sébastien Coron, Yevgeniy Dodis, Cécile Malinaud, and Prashant Puniya. Merkle-damgrd revisited: How to construct a hash function. In

*Advances in Cryptology CRYPTO 2005*, volume 3621 of *Lecture Notes in Computer Science*, pages 430–448. Springer Berlin Heidelberg, 2005.

[7] Shoichi Hirose, JeHong Park, and Aaram Yun. A simple variant of the merkle-damgrd scheme with a permutation. In *Advances in Cryptology ASIACRYPT 2007*, volume 4833 of *Lecture Notes in Computer Science*, pages 113–129. Springer Berlin Heidelberg, 2007.

[8] Ari Juels and Burton S. Kaliski, Jr. Pors: Proofs of retrievability for large files. In *Proceedings of the 14th ACM Conference on Computer and Communications Security*, CCS '07, pages 584–597, New York, NY, USA, 2007. ACM.

[9] Ramakrishna Kotla, Lorenzo Alvisi, and Michael Dahlin. Safestore: A durable and practical storage system. In *USENIX Annual Technical Conference*, pages 129–142. USENIX, 2007.

[10] Ueli Maurer, Renato Renner, and Clemens Holenstein. Indifferentiability, impossibility results on reductions, and applications to the random oracle methodology. In *Theory of Cryptography*, volume 2951 of *Lecture Notes in Computer Science*, pages 21–39. Springer Berlin Heidelberg, 2004.

[11] Thomas Ristenpart, Hovav Shacham, and Thomas Shrimpton. Careful with composition: Limitations of indifferentiability and universal composability. IACR Cryptology ePrint Archive, Report 2011/339, 2011.

[12] Thomas Ristenpart, Hovav Shacham, and Thomas Shrimpton. Careful with composition: Limitations of the indifferentiability framework. In *Advances in Cryptology EUROCRYPT 2011*, volume 6632 of *Lecture Notes in Computer Science*, pages 487–506. Springer Berlin Heidelberg, 2011.