# How to Build a Hash Function from Any Collision-Resistant Function

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

[1] Dept. of Computer Science & Engineering, University of California San Diego
9500 Gilman Drive, La Jolla, CA 92093-0404, USA
tristenp@cs.ucsd.edu
http://www-cse.ucsd.edu/users/tristenp
[2] Dept. of Computer Science, Portland State University
Room 120, Forth Avenue Building, 1900 SW 4th Avenue, Portland OR 97201 USA
teshrim@cs.pdx.edu
http://www.cs.pdx.edu/∼teshrim
[3] Faculty of Informatics, University of Lugano
Via Buffi 13, CH-6900 Lugano, Switzerland
thomas.shrimpton@unisi.ch
http://www.inf.unisi.ch/

**Abstract.** Recent collision-finding attacks against hash functions such as MD5 and SHA-1 motivate the use of *provably* collision-resistant (CR) functions in their place. Finding a collision in a provably CR function implies the ability to solve some hard problem (e.g., factoring). Unfortunately, existing provably CR functions make poor replacements for hash functions as they fail to deliver behaviors demanded by practical use. In particular, they are easily distinguished from a random oracle. We initiate an investigation into building hash functions from provably CR functions. As a method for achieving this, we present the Mix-Compress-Mix (MCM) construction; it envelopes any provably CR function $H$ (with suitable regularity properties) between two injective "mixing" stages. The MCM construction simultaneously enjoys (1) provable collision-resistance in the standard model, and (2) indifferentiability from a monolithic random oracle when the mixing stages themselves are indifferentiable from a random oracle that observes injectivity. We instantiate our new design approach by specifying a blockcipher-based construction that appropriately realizes the mixing stages.

## 1 Introduction

Background. SHA-1, a Merkle-Damgård style [24, 15] iterated function, is provably collision resistant under the *assumption* that its underlying compression function is collision resistant. But the recent collision-finding attacks against SHA-1 (and related hash functions) [37, 38] have made clear the point that assumptions of collision resistance are often unfounded in practice.

Rather than assuming collision resistance outright, several works [12, 22, 26, 33, 14] build functions for which the guarantee of collision resistance rests, in a

K. Kurosawa (Ed.): ASIACRYPT 2007, LNCS 4833, pp. 147–163, 2007.

provable way, on the hardness of some well-studied computational problem. As a simple example, consider the function $H(m) = x^m \mod n$ where $x$ is some fixed base and $n$ is a (supposedly) hard-to-factor composite [26, 33]. This function is (what we shall call) *provably* CR since there exists a formal reduction showing that the ability to find collisions in $H$ implies the ability to efficiently factor $n$.

But such a collision-resistant function is not a *hash function*, at least not when one attempts to define a hash function by its myriad uses in practice[1]. For example, hash functions are frequently used as a way to compress and 'mix-up' strings of bits in an 'unpredictable' way; here it seems clear that the intent is for the hash function to mimic a random oracle, a publicly available random function with a large domain. Unfortunately, the provably CR function $H$ is a poor real-world instantiation of a random oracle. Note, for example, that $H(2m) \equiv H(m)^2 \pmod n$, which would be true with exceedingly small probability if $H$ were instead a random oracle. The very structure that gives $H$ and other provably CR functions their collision-resistance thus renders them useless for many practical applications of hash functions [12, 36].

On the other hand, recent results [13, 2, 11] offer constructions that 'behave' as random oracles (and are called pseudorandom oracles, or PROs) when the underlying primitives are themselves idealized objects, like fixed-input length random oracles or ideal ciphers. In theory then, a PRO is a secure hash functions in a very broad sense. But the security guarantees offered by a PRO only hold in an idealized model. When one steps outside of the ideal model in which the security proofs take place, the actual security guarantees are much less clear. As an example, Bellare and Ristenpart [2] have pointed out that the PRO constructions from [13] fail to be collision resistant when the underlying compression function is only assumed to be CR (rather than being a fixed-input-length random oracle).

This paper. We begin an investigation into methods for building functions that are *both* provably CR in the standard model and provably pseudorandom oracles in idealized models. In particular, we offer a generic construction that we call Mix-Compress-Mix, or MCM; See Figure 1. Essentially MCM is a way to encapsulate a provably CR function in such a way that the resulting object is a PRO when the encapsulation steps behave ideally, and yet remains provably collision resistant in the standard model (i.e., when the encapsulation steps are only complexity theoretic objects).

The construction is simple: first apply an injective "mixing" step $\mathcal{E}_1$ to the input message, then compress the result using a provably CR function $H$, and finally apply a second injective "mixing" step $\mathcal{E}_2$ to produce the output. Here $H$ and $\mathcal{E}_1$ can accept variable-input-lengths. Note that since MCM is building a hash function, the mixing steps $\mathcal{E}_1$ and $\mathcal{E}_2$ are necessarily deterministic and publically computable functions. By demanding that they also be injective, we have immediately that collisions against MCM imply collisions against $H$. We stress that *no* cryptographic assumptions about the mixing steps are needed to prove collision resistance of MCM.

[1] This viewpoint is not ours alone. One of the designers of VSH [12], Arjen Lenstra, once publicly stated "VSH is not a hash function."

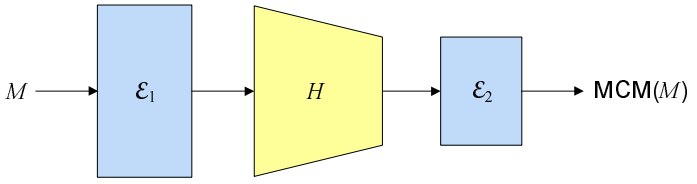


**Fig. 1.** The MCM construction: $H$ is a collision resistant hash function, and $\mathcal{E}_1, \mathcal{E}_2$ are mixing functions. All three components of MCM must be deterministic and publically computable.

At the same time, MCM behaves like a random oracle when $\mathcal{E}_1, \mathcal{E}_2$ are PROs, and the CR hash function is close to regular (i.e., the preimage set of any particular output isn't too large). In fact, we will actually construct $\mathcal{E}_1, \mathcal{E}_2$ to be *pseudorandom injective oracles*, or PRIOs; we'll say more about these in a moment. To make precise our use of the word "behaves" above, we use the indifferentiability framework of Maurer et al. [23]. We'll prove that MCM is indifferentiable from a monolithic random oracle when the mixing steps $\mathcal{E}_1$ and $\mathcal{E}_2$ are indifferentiable from random oracles (that observe injectivity). While the formal results are quite technical, the practical intuition behind the security of MCM is straightforward: the mixing steps obfuscate input-output relationships of the underlying compressing step. Recall our provably CR example $H(m) = x^m \bmod n$ and the associated attack that distinguished it from a random oracle. Adapting that attack for use against $\mathcal{H}(M) = \mathcal{E}_2(H(\mathcal{E}_1(M)))$ requires that the adversary determine non-trivial input-output relationships across both $\mathcal{E}_1$ and $\mathcal{E}_2$, too.

One might be tempted to think a construction even simpler than MCM meets our goals. In Section 4 we discuss natural simplifications of MCM (e.g., dropping $\mathcal{E}_1$ or lifting our stringent injectivity requirements), showing that these fall short in one way or another. Moreover, we review in more detail why existing approaches for building hash functions also fail.

Although we have just described MCM in the variable-input-length setting, we note that it also works for building a dual-property compression function (i.e., a fixed-input-length function) from any CR compression function. The result could be then be used inside a multi-property-preserving domain extension transform such as EMD [2].

A new approach to hash function design. By generically composing appropriate mixing and compressing stages, MCM allows the following separation of design tasks. First, design a function with strong guarantees of collision-resistance, inducing whatever structure is necessary. Second, design an injective function that destroys any structure present in its input. This approach is a significant departure from traditional hash function designs, in which one typically constructs a compression function that must necessarily (and simultaneously) be secure in various ways. With MCM, we instead build a hash function by designing components to achieve specific security goals. The benefits of such specialized components are immediate: MCM allows building a single hash function that has

very strong CR guarantees while simultaneously being suitable for instantiating a random oracle.

Secure mixing steps. Remaining is the question of how to build mixing steps sufficient for the goals of MCM. As we've said, we require the mixing steps to be both injective *and* indifferentiable from a random oracle that observes injectivity. At first glance these requirements might seem overly burdensome. Can't the requirement simply be for the mixing steps to realize pseudorandom oracles, which we already know (via [13, 11, 2]) how to build? No: while a PRO would satisfy the second constraint, albiet with some additive birthday-bound loss in concrete security, it would not at the same time suffice for MCM's crucial standard-model CR guarantee. This is because a PRO provides no guarantees of collision-resistance outside of an idealized model. In fact, simultaneously satisfying both requirements, injectivity and indifferentiability, is technically challenging.

To our knowledge, building a PRIO has never been considered before. Dodis and Puniya [17,16] consider a similar goal, that of building random permutations from random functions, but these are invertible by construction, whereas PRIOs are not. Moreover, their proofs of security only hold for honest-but-curious adversaries. We therefore present the Tag-and-Encipher (TE) construction for realizing a PRIO (see Section 5). It is a blockcipher mode of operation (which also employs a single trapdoor one-way permutation call) that is injective by construction. In the ideal cipher model and under the assumption of trusted setup of the trapdoor permutation, the TE construction is indifferentiable from an injective random oracle. While not particularly efficient, we view the TE construction as a proof-of-concept, and hope it fosters future efforts to build these novel primitives.

Notes on indifferentiability and composability. In order to accomplish our task of building a hash function with both strong standard model and ideal model guarantees, we exercise the indifferentiability framework in novel ways. First, both MCM and TE are a combination of complexity-theoretic objects (the CR function $H$ and the trapdoor permutation) and information-theoretic objects (the idealized components). Previous indifferentiability results have been solely information-theoretic. Second, our model allows the simulator to choose the trapdoor permutation utilized in TE. These two facts imply limitations on the generic composability of our schemes. Composability refers to the guarantee that *any* cryptographic scheme proven secure using an ideal object remains secure when this object is replaced by a construction that is indifferentiable from it. In practice the limited composability of our constructions means that they might not be suitable for all applications of random (injective) oracles. We discuss this matter in more detail, and pose some interesting open questions raised by it, in Section 6.

## 2 Preliminaries

Basics. Let $X, Y \in \{0,1\}^*$. We denote the concatenation of $X$ and $Y$ by $X \,||\, Y$ or simply $XY$. The $i^{th}$ bit of $X$ is $X[i]$ and so $X = X[1]X[2]\cdots X[|X|]$. We

write $X|_n$ (resp. $X|^n$) to represent the substring consisting of the last (resp. first) $n$ bits of $X$ for any $n \leq |X|$. For a set $S$ we often write $S \xleftarrow{\cup} x$, which means $S \leftarrow S \cup \{x\}$. We define $\mathsf{Time}_f(\mu)$ as the worst-case time to compute $f$ on a message of length at most $\mu$.

Following [8, 13] we utilize Interactive Turing Machines (ITM) for our computational model. Cryptographic primitives, schemes, and adversaries are all interactive Turing machines.

Random functions and injections. Let *Dom* and *Rng* be sets. Recall that a function $f$: *Dom* $\rightarrow$ *Rng* is injective if $f(X) = f(X')$ implies that $X = X'$. (Necessarily for an injection $|Dom| \leq |Rng|$.) For simplicity, we only consider injections with constant stretch $\tau$. Particularly if $X \in Dom$ then $|f(X)| = |X| + \tau$. The following algorithms implement a *random function* and a *random injection*.

| Algorithm $\mathsf{RF}_{Dom,Rng}(X)$: | Algorithm $\mathsf{RI}_{Dom,Rng}(X)$: |
|---|---|
| If $\mathtt{R}[X] \neq \perp$ then Ret $\mathtt{R}[X]$ | $\ell \leftarrow \lvert X\rvert + \tau$ |
| Ret $\mathtt{R}[X] \xleftarrow{\$} Rng$ | If $\mathtt{I}[X] \neq \perp$ then Ret $\mathtt{I}[X]$ |
| | $\mathtt{I}[X] \xleftarrow{\$} \{0,1\}^\ell \backslash \mathcal{R}_\ell$ |
| | $\mathcal{R}_\ell \xleftarrow{\cup} \mathtt{I}[X]$ |
| | Ret $\mathtt{I}[X]$ |

The tables $\mathtt{R}$ and $\mathtt{I}$ are initially everywhere set to $\perp$ and the set $\mathcal{R}_\ell$ is initially empty for every $\ell$. We write $f = \mathsf{RF}_{Dom,Rng}$ to signify that $f$ is an ITM mapping points from *Dom* to *Rng* according to the algorithm specified above. We write $\mathsf{RF}_{d,r}$ if $Dom = \{0,1\}^d$ and $Rng = \{0,1\}^r$ for some numbers $d, r$. We write $\mathcal{I} = \mathsf{RI}_{Dom,Rng}$ for an ITM mapping points from *Dom* to *Rng* as per the algorithm specified above. (The other notational conventions lift to $\mathsf{RI}$ in the obvious ways.) A random oracle is a random function that is publically accessible by all parties. Similarly an random (or ideal) injection is a publically-accessible random function that respects injectivity.

Ideal ciphers. For integers $k, n > 0$, a blockcipher $E$: $\{0,1\}^k \times \{0,1\}^n \rightarrow \{0,1\}^n$ is a function for which $E(K,\cdot) = E_K(\cdot)$ is a permutation for every $K \in \{0,1\}^k$. The inverse of $E$ is $D$ and is defined such that $D(K,Y) = M$ iff $E(K,M) = Y$. An ideal cipher is a blockcipher uniformly selected from $\mathrm{BC}(k,n)$, the space of all blockciphers with $k$-bit keys and $n$-bit blocksize. In the ideal cipher model, both an ideal cipher $E$ and its inverse are given to all parties as oracles.

Security notions. Let $f$: $\mathcal{K} \times Dom \rightarrow Rng$ be a function family indexed by a non-empty key space $\mathcal{K}$. Then we define the collision-finding advantage of an adversary $\mathcal{A}$ against $f$ as

$$\mathbf{Adv}^{\mathrm{cr}}_f(\mathcal{A}) = \Pr\left[\, f_K(X) = f_K(X') \;:\; K \xleftarrow{\$} \mathcal{K}; (X, X') \xleftarrow{\$} \mathcal{A}(K) \,\right]$$

where the probability is over the random choice of $K$ and the random coins utilized by $\mathcal{A}$.

A function $f$: $Dom \rightarrow \{0,1\}^\eta$ is *regular* if each image has an equal number of preimages. A function family $f$: $\mathcal{K} \times Dom \rightarrow \{0,1\}^\eta$ is regular if $f_K$ is regular for

each $K \in \mathcal{K}$. Associated to a function family $f$ is the set $\mathsf{PreIm}(K, \ell, Y)$ that, for each $K \in \mathcal{K}$, $\ell$ such that $\{0,1\}^\ell \subseteq \mathit{Dom}$, and $Y \in \{0,1\}^\eta$, is the set of preimages (under $K$) of $Y$ that are of length $\ell$. That is, $\mathsf{PreIm}(K, \ell, Y) = \{X : X \in \mathit{Dom} \wedge |X| = \ell \wedge f_K(X) = Y\}$. We also define the following function related to $f$

$$\delta(K, \ell, Y) = \frac{|\mathsf{PreIm}(K, \ell, Y)| - 2^{\ell-\eta}}{2^\ell}.$$

The $\delta$ function measures how much bigger (or smaller) a particular preimage set is than it would be if $f_K$ were regular. We define $\Delta_K = \max\{\delta(K, \ell, Y)\}$, where the maximum is taken over all choices of $\ell$ and $Y$, and we say a function family $f$ is $\Delta$-regular if

$$\frac{\sum_{K \in \mathcal{K}} \Delta_K}{|\mathcal{K}|} \leq \Delta.$$

Intuitively, this measures on average (over keys) how far $f$ is from regular.

Let $\mathbb{F}$ be a *trapdoor permutation generator*: on input $1^k$ it outputs a trapdoor permutation pair $(f, f^{-1})$ where $f\colon \{0,1\}^k \to \{0,1\}^k$ and $f^{-1}(f(X)) = X$. The one-way advantage of an adversary $\mathcal{A}$ against $\mathbb{F}$ for security parameter $k$ is defined by

$$\mathbf{Adv}^{\mathrm{owf}}_{\mathbb{F}}(\mathcal{A}) = \Pr\left[ f(X) = f(X') \;:\; \begin{array}{c} (f, f^{-1}) \xleftarrow{\$} \mathbb{F}(1^k); X \xleftarrow{\$} \{0,1\}^k; \\ Y \leftarrow f(X); X' \xleftarrow{\$} \mathcal{A}(f, Y) \end{array} \right].$$

The RSA and Rabin function families are conjectured to allow generation of secure trapdoor permutations [32, 28, 29].

PROs AND PRIOs. The notion of indifferentiability [23] is a generalization of conventional indistinguishability [18].] It facilitates reasoning about the ability of constructions to emulate some idealized functionality (e.g., a random oracle) in settings where the construction itself utilizes public, idealized components (e.g., an ideal cipher or fixed-input-length (FIL) random oracle). We follow the formalization of indifferentiability from [13, 2] to define security for pseudorandom oracles and pseudorandom injective oracles. First, a *simulator* $\mathcal{S} = (\mathcal{S}_1, \ldots, \mathcal{S}_l)$ is an interactive Turing machine with $l$ interfaces $\mathcal{S}_1, \ldots, \mathcal{S}_l$. The interfaces share common state, i.e. all variables defined in one interface are available to all other interfaces. Let $C$ be some cryptographic scheme utilizing primitives $f_1, \ldots, f_l$ and let *Dom* and *Rng* be non-empty sets. We define the pro and prio advantage of an adversary $\mathcal{A}$ against $C$ with respect to simulator $\mathcal{S}$ as

$$\mathbf{Adv}^{\mathrm{pro}}_{C,\mathcal{S}}(\mathcal{A}) = \Pr\left[\mathcal{A}^{C^{f_1,\ldots,f_l}, f_1, \ldots, f_l} \Rightarrow 1\right] - \Pr\left[\mathcal{A}^{\mathcal{F}, \mathcal{S}^{\mathcal{F}}} \Rightarrow 1\right]$$

$$\mathbf{Adv}^{\mathrm{prio}}_{C,\mathcal{S}}(\mathcal{A}) = \Pr\left[\mathcal{A}^{C^{f_1,\ldots,f_l}, f_1, \ldots, f_l} \Rightarrow 1\right] - \Pr\left[\mathcal{A}^{\mathcal{I}, \mathcal{S}^{\mathcal{I}}} \Rightarrow 1\right]$$

where $\mathcal{F} = \mathsf{RF}_{\mathit{Dom},\mathit{Rng}}$ and $\mathcal{I} = \mathsf{RI}_{\mathit{Dom},\mathit{Rng}}$ and the probabilities are over the random coins used by the appropriate objects. We emphasize that the simulator has oracle access to the idealized object ($\mathcal{F}$ or $\mathcal{I}$), but does not see the queries $\mathcal{A}$

makes to it. In the case that the construction uses publically-keyed components (e.g., the key for a CR function), all three entities ($C$, $\mathcal{S}$, $\mathcal{A}$) have access to the key. We disallow $\mathcal{A}$ from making *pointless* queries, which in this setting means querying an oracle twice.

Informally we call a cryptographic scheme $C$ a pseudorandom oracle (PRO), or say it is indifferentiable from a random oracle, if there exists an "efficient" simulator against which all adversaries have "small" pro advantage. Likewise we call a cryptographic scheme $C$ a pseudorandom injective oracle (PRIO) if there exists an "efficient" simulator against which all adversaries have "small" prio advantage. We do not formalize "efficient" or "small", giving concrete running times and bounds, instead.

We formalize trusted setup of a trapdoor permutation generator $\mathbb{F}$ via an interactive Turing machine $\mathsf{TGen}$ that behaves as follows. When called, it computes $(f, f^{-1}) \xleftarrow{\$} \mathbb{F}(1^k)$ and returns $f$. Subsequent calls return the same $f$. Constructions that utilize a trapdoor permutation are given oracle access to $\mathsf{TGen}$, for example in the pro and prio definitions $f_i = \mathsf{TGen}$ for some $i \in [1\,..\,l]$. We also allow the simulator to run the oracle corresponding to $\mathsf{TGen}$. This means, in particular, that the simulator knows the trapdoor $f^{-1}$, while the adversary does not. See Section 6 for a discussion of the repercussions of this modeling decision.

## 3 The MCM Construction

Fix numbers $\eta$ and $\tau$. Let $H\colon\ \mathcal{K} \times \mathcal{M}_H \to \{0,1\}^\eta$ be a function family with key space $\mathcal{K}$ and domain $\mathcal{M}_H = \{0,1\}^{\leq L}$ for some large number $L$ (e.g., $2^{64}$). Let $\mathcal{E}_1\colon\ \mathcal{M} \to \mathcal{M}_H$ be an injective function where $\mathcal{M} = \{0,1\}^{\leq L'}$ for $L' = L - \tau$. For any $X \in \mathcal{M}$ we have that $|\mathcal{E}_1(X)| = |X| + \tau$, hence $\tau$ is the *stretch* of $\mathcal{E}_1$. Finally let $\mathcal{E}_2\colon\ \{0,1\}^\eta \to \{0,1\}^{\eta+\tau}$ be an injective function. Then we define the hash function $\mathcal{H} = \mathsf{MCM}[\mathcal{E}_1, H, \mathcal{E}_2]$ with key space $\mathcal{K}$, domain $\mathcal{M}$, and range $\{0,1\}^{\eta+\tau}$ by $\mathcal{H}_K(M) = \mathcal{H}(K, M) = \mathcal{E}_2(H_K(\mathcal{E}_1(M)))$. Overloading our notation, if $\mathcal{I}_1 = \mathsf{RI}_{\mathcal{M},\mathcal{M}_H}$ and $\mathcal{I}_2 = \mathsf{RI}_{\eta,\eta+\tau}$ then we write $\mathcal{H} = \mathsf{MCM}[\mathcal{I}_1, H, \mathcal{I}_2]$ where now $\mathcal{H}$ is itself an ITM using oracle access to $\mathcal{I}_1$ and $\mathcal{I}_2$ to calculate $\mathcal{H}_K(M) = \mathcal{I}_2(H_K(\mathcal{I}_1(M)))$.

Here $\tau$ is also the stretch of $\mathcal{H}$ — it's the number of bits beyond $\eta$ needed to hold a hash value. Ideally $\tau = 0$, in which case $\mathcal{E}_1$ and $\mathcal{E}_2$ would be a permutations. We have the following theorem, which states that $\mathcal{H}$ inherits the collision-resistance of $H$.

**Theorem 1.** *Fix $\eta > 0$ and $\tau \geq 0$. Let $H\colon\ \mathcal{K} \times \mathcal{M}_H \to \{0,1\}^\eta$ be a function and $\mathcal{E}_1\colon\ \mathcal{M} \to \mathcal{M}_H$ and $\mathcal{E}_2\colon\ \{0,1\}^\eta \to \{0,1\}^{\eta+\tau}$ be injections. Let $\mathcal{H} = \mathsf{MCM}[\mathcal{E}_1, H, \mathcal{E}_2]$. Let $\mathcal{A}$ be an adversary that runs in time $t$ and outputs messages each of length at most $\mu$. Then there exists an adversary $\mathcal{B}$ such that*

$$\mathbf{Adv}^{\mathrm{cr}}_{\mathcal{H}}(\mathcal{A}) = \mathbf{Adv}^{\mathrm{cr}}_{H}(\mathcal{B})$$

*where $\mathcal{B}$ runs in time $t' \leq t + 2(c\mu + \mathsf{Time}_{\mathcal{E}_1}(\mu))$ for an absolute constant $c$.* □

*Proof.* Let $\mathcal{B}$ be the adversary that behaves as follows. On input key $K$ It runs $\mathcal{A}(K)$, which eventually outputs $(X, X')$. Then $\mathcal{B}$ outputs $(\mathcal{E}_1(X), \mathcal{E}_1(X'))$. We have that if $\mathcal{H}_K(X) = \mathcal{H}_K(X')$ then because $\mathcal{E}_1$ and $\mathcal{E}_2$ are injections, necessarily $H_K(\mathcal{E}_1(X)) = H_K(\mathcal{E}_1(X'))$. Adversary $\mathcal{B}$ runs in time $t' \leq t + 2(c\mu + \mathsf{Time}_{\mathcal{E}_1}(\mu))$ where $c$ is an absolute constant. ∎

We point out that similar theorems can be given for several other hash function properties, including target collision-resistance (TCR, or eSec), preimage resistance, and always preimage resistance (aPre) [34][2]. The next theorem captures that MCM is a PRO if both $\mathcal{E}_1$ and $\mathcal{E}_2$ are modeled as random injections.

**Theorem 2.** *Fix $\eta > 0$ and $\tau \geq 0$. Let $H\colon \mathcal{K} \times \mathcal{M}_H \to \{0,1\}^\eta$ be a $\Delta$-regular function, $\mathcal{I}_1 = \mathsf{RI}_{\mathcal{M},\mathcal{M}_H}$, and $\mathcal{I}_2 = \mathsf{RI}_{\eta,\eta+\tau}$. Let $\mathcal{H} = \mathsf{MCM}[\mathcal{I}_1, H, \mathcal{I}_2]$. Let $\nu$ be the minimal message length of $\mathcal{H}$. Let $\mathcal{A}$ be an adversary that runs in time $t$ and making at most $(q_1, q_2, q_3)$ queries with the combined length of all queries being at most $\mu$. Then there exists an adversary $\mathcal{B}$ such that*

$$\mathbf{Adv}^{\mathrm{pro}}_{\mathcal{H},\mathcal{S}}(\mathcal{A}) \leq \mathbf{Adv}^{\mathrm{cr}}_{H}(\mathcal{B}) + \frac{(q_1+q_3)^2}{2^{\eta+\tau}} + \frac{(q_1+q_2)^2}{2^{\nu+\tau}} + (q_1+q_2)q_3\left(\frac{1}{2^\eta} + \Delta\right)$$

*where the simulator $\mathcal{S}$, specified below, runs in time $t_{\mathcal{S}} \leq c\mu(q_1 + q_1 q_3)$ for some absolute constant $c$ and makes at most $\min\{q_2, q_3\}$ oracle queries. Adversary $\mathcal{B}$ runs in time at most $t_{\mathcal{B}} \leq t + t_{\mathcal{S}} + c'\mu$ for some absolute constant $c'$.* □

The proof of this theorem is given in the full version of the paper [31], though below we give a sketch highlighting the main aspects of the proof. First, we discuss the theorem statement. As long as $\mathcal{E}_1$ and $\mathcal{E}_2$ are PRIOs we can securely replace them by actual random injections (as per the composition theorem of [23]). Then, Theorem 2 states that no adversary can differentiate between a real random oracle and the construction unless it is given sufficient time to break the collision-resistance of $H$ or allowed to make approximately $2^{(\tau+\min\{\eta,\nu\})/2}$ queries. Here $\nu$ could in fact be small, since this is the minimal message length in the domain of our hash function (and we'd certainly want to include short messages). However, in practice, $H$ will have some minimal message length $\nu_H$ (e.g., the blocksize of an underlying compression function) to which short messages would necessarily be padded anyway. Thus, $\mathcal{H}$ can 'aggressively' pad short strings to a minimal length $\nu = \nu_H - \tau$, recovering our security guarantee.

*Proof (Sketch).* We first fix a simulator $\mathcal{S} = (\mathcal{S}_1, \mathcal{S}_2)$, which has access to the random oracle $\mathcal{R}$. The first interface $\mathcal{S}_1$ implements a random injection $\hat{\mathcal{I}}_1 = \mathsf{RI}_{\mathcal{M},\mathcal{M}_H}$ without ever using its access to $\mathcal{R}$. The second interface works as described below (recall that it has access to all of the values defined for $\hat{\mathcal{I}}_1$):

[2] Although it is unclear how one would prove that MCM preserves the other notions from [34], specifically everywhere preimage resistance (ePre), second-preimage resistance (Sec) and always second-preimage resistance (aSec).

**procedure** $\mathcal{S}_2(Y)$

If $\exists M$ s.t. $Y = H_K(\hat{\mathcal{I}}_1(M))$ then
  Ret $\mathcal{R}(M)$
Ret $C \xleftarrow{\$} \{0,1\}^\eta$

This interface checks if $\hat{\mathcal{I}}_1$ already maps a string $M$ to a preimage under $H_K$ of the queried value $Y$. In the case that multiple such $M$ exist, then the lexicographically first is used. If such an $M$ exists, then the simulator queries $\mathcal{R}$ on $M$ and the output of $\mathcal{S}_2(Y)$ is "programmed" to match this value.

Now we argue that no adversary $\mathcal{A}$, given $K$, can differentiate between oracles $(\mathcal{H}, \mathcal{I}_1, \mathcal{I}_2)$ and $(\mathcal{R}, \mathcal{S}_1, \mathcal{S}_2)$. Let $\mathcal{O}_1, \mathcal{O}_2, \mathcal{O}_3$ be the oracles given to $\mathcal{A}$. By the construction of the simulator, the adversary gains no advantage by querying messages in the order of the construction, i.e. a query $X \leftarrow \mathcal{O}_2(M)$ and then $\mathcal{O}_3(H_K(X))$. On the other hand, there do exist sequences of queries that can cause the simulator to fail (with high probability) to respond in a manner consistent with the responses of $(\mathcal{H}, \mathcal{I}_1, \mathcal{I}_2)$. We argue that these "bad" sequences are hard for any adversary to generate.

The first is if two messages $M$ and $M'$ are queried to $\mathcal{S}_1$ and the returned values are such that $H_K(\hat{\mathcal{I}}_1(M)) = H_K(\hat{\mathcal{I}}_1(M')) = Y$. In this case the adversary can query $Y$ to $\mathcal{S}_2$ and the simulator can at best guess whether to return $\mathcal{R}(M)$ or $\mathcal{R}(M')$ (which are distinct with high probability). But note that since $\hat{\mathcal{I}}$ is an injection, this actually implies that $\mathcal{A}$ has found a collision against $H_K$. This reflects the first term in the bound of the theorem statement.

The second "bad" sequence occurs if $\mathcal{A}$ queries $\mathcal{S}_2(Y)$, forcing the simulator to commit to a return value $Z$, and then later queries $\mathcal{S}_1(M)$ which returns a value $X$ such that $H_K(X) = Y$. Since the probability is low that $\mathcal{R}(M) = Z$, there exists little chance that $\mathcal{S}$ answered the original query consistently. But the probability that $\mathcal{S}_1(M)$ returns $Y$ is in fact the probability of choosing a random domain point $X$ such that $H_K(X) = Y$. Indeed the $\Delta$-regularity of $H$ gives that this can only happen with low probability. This accounts for the last term in the bound.

The remaining two unexplained terms correspond to birthday-bounds for moving between random injections and random functions. For a complete proof see the full version of the paper [31]. ∎

## 4 Insecurity of Other Approaches

Here we give just a brief investigation of several alternative approaches to MCM. In all cases, either the resulting object is not provably collision-resistant in the standard model or not provably a PRO in an ideal model.

Using existing Blockcipher-based hash functions. Let $E$: $\{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ be a blockcipher, modeled as ideal. Let $f$ be a $2n$-bit to $n$-bit compression function. Fix some suitable domain extension transform, for example Merkle-Damgård with a prefix-free encoding. That is $\mathcal{H}(M) = f^+(g(M))$, where $f^+(M_1 \cdots M_m)$ is equal to $Y_m$ defined recursively by $Y_0 = IV$ (some constant) and $Y_i = f(Y_{i-1}, M_i)$, and $g$: $\{0,1\}^* \to (\{0,1\}^n)^+$ is a prefix-free padding

function. For simplicity let $g(M)$ simply split $M$ into blocks of $n-1$ bits ($M$ having been appropriately padded), and then appending a zero to each block except the last and appending a one to the last block. If $f$ is one of the twenty group-1/2 schemes from [5], then $\mathcal{H}$ is collision-resistant in the ideal cipher model. Moreover, a recent paper by Chang et al. [11] shows that sixteen of these twenty yield a PRO $\mathcal{H}$.

However as soon as one leaves the ideal cipher model, $\mathcal{H}$ is *not* provably CR. For example let $E'$ be the blockcipher defined as follows:

$$E'(K,M) = \begin{cases} M & \text{if } K = 0^k \\ E(K,M) & \text{otherwise} \end{cases} .$$

where, now, $E$ is no longer ideal. Let $f(Y_{i-1}, M_i) = E'(M_i, Y_{i-1}) \oplus Y_{i-1}$. We can see that an adversary can trivially find collisions against $\mathcal{H}$ built using $E'$. This is true even though $E'$ is a good pseudorandom permutation (the usual standard model security property of blockciphers) whenever $E$ is also.[3]

Removing injectivity requirements. If either $\mathcal{E}_1$ or $\mathcal{E}_2$ are not injective, then the MCM construction looses its provable collision-resistance. Assuming they are built from using blockciphers (as we suggest), then one can, in spirit similar to the counter-example above, construct a collision resistant function $H'$ and a good PRP $E'$ that, when utilized in MCM, would lead to a trivial collisions.

Note that one might imagine replacing $\mathcal{E}_1$ and $\mathcal{E}_2$ with objects that are not injective, yet have some other standard model guarantees to ensure provable collision-resistance in MCM. Short of establishing their collision-resistance, its not clear what properties could achieve this goal. Additionally, this approach would seem to violate the separation of design tasks intrinsic to the MCM approach.

Omitting $\mathcal{E}_1$ from MCM. If one omits the first "mixing" step $\mathcal{E}_1$ of MCM, then the construction no longer results in a PRO. This result is essentially equivalent to the Coron et al. insecurity result regarding the composition of a CR and one-way function $H$ with a random oracle [13], but we state a version of it here for completeness. Let $\mathcal{H} = \mathsf{CM}[H, \mathcal{I}_2]$ be this modified construction for $\mathcal{I}_2 = \mathsf{RI}_{\eta,\eta+\tau}$, i.e. $\mathcal{H}(M) = \mathcal{I}_2(H(M))$. Now we show that $\mathcal{H}$ is easily differentiable from a true random oracle $\mathcal{R} = \mathsf{RF}_{\mathcal{M}_H,\eta+\tau}$. Let $\mathcal{A}$ be an adversary that queries it's first oracle on a uniformly selected message of length $M \in \mathcal{M}_H$ of some length $\ell$. Let the returned value be $C$. Now the adversary queries its second oracle (representing either $\mathcal{I}_2$ or a simulator) on $H_K(C)$. Let the returned value be $C'$. If $C = C'$ then $\mathcal{A}$ returns one, guessing that it's interacting with the construction. Otherwise it returns zero, guessing that it's interacting with the true random oracle. We have that $\Pr\left[\mathcal{A}^{\mathcal{H},\mathcal{I}_2} \Rightarrow 1\right] = 1$. On the other hand, $\Pr\left[\mathcal{A}^{\mathcal{R},\mathcal{S}} \Rightarrow 1\right]$ is bounded by the advantage of a related adversary in breaking the one-wayness of $H$.

Allowing $\mathcal{E}_1, \mathcal{E}_2$ to be invertible. Our formalization of PRIOs ensure that constructions meeting the goal are *not* invertible. Thus, objects that are invert-

[3] Hopwood and Wagner noted (in postings on sci.crypt) that one could exhibit good PRPs that would make finding collisions in the twenty [5] functions trivial.

ible do not meet the goal. It remains an open question whether MCM is, in fact, secure under easy-to-invert mixing steps.

## 5 Secure Mixing Steps: The TE Construction

Pseudorandom injective oracles. We now turn to showing the feasibility of instantiating the mixing steps $\mathcal{E}_1$ and $\mathcal{E}_2$ starting from blockciphers. We note that our eventual construction also works starting from a suitable fixed-input-length random oracle. This would have slight theoretical benefits because of a lack of implication between the ROM and ICM [17]. However, one might want to utilize blockciphers and the proofs are only rendered more complex when considering invertible components, thus we stick to the former.

We specify a construction that is a PRIO, i.e. indifferentiable from a random injection. Under the composability guarantees of the indifferentiability framework [23] (though see Section 6), the security of schemes (e.g., MCM) proven secure while modeling components as ideal injections remains when these objects are replaced by PRIOs.

At first glance the notion of a PRIO might appear to be essentially equivalent to that of a pseudorandom oracle. The distinction is analogous to the difference between PRPs and PRFs. Indeed, random injections and random functions behave similarly up to a birthday-bound, which implies that any PRIO is a good PRO and vice versa. But the more important (and subtle) concern is that the closeness of the definitions might lead one to the conclusion that there are trivial constructions for our mixing steps, utilizing any PRO. However, this would be entirely insufficient for our application because, while a PRO appears injective with high probability, it is *not* necessarily injective by construction. Once we step outside of idealized models we would then have a standard model object that *does not* suffice for the collision-resistance guarantee of Section 3. So for clarity of exposition and analysis, we found it useful to draw a distinction between the two objects.

Building a PRIO that is injective by construction from a blockcipher (modeled as ideal) proves a challenging task. Our object must be publically computable, so no secret keys are allowed. A minimum intuitive security requirement for the object is that the outputs resulting from applying it to two messages that differ in a single bit must appear to have been chosen independently at random, even when adversaries have direct access to the underlying blockcipher. This rules out the straightforward use of existing blockcipher modes of operation, such as CBC, with a public key and fixed IV or even the more complex variable-length enciphering schemes (e.g. [21, 20, 30, 19]).

The TE construction. Our construction utilizes two blockciphers and a trapdoor one-way permutation. Note that in the ideal cipher model one can easily derive two ciphers from a single cipher $\hat{E}$ at the cost of one bit of keying material: $E(K, M) \equiv \hat{E}(1 \,||\, K, M)$ and $E'(K, M) \equiv \hat{E}(0 \,||\, K, M)$. For simplicity then we assume access to two ciphers $E$: $\{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$ and $E'$: $\{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$. The cipher $E$ will be used in a blockcipher mode

Algorithm $\mathcal{E}(M)$:
$T \leftarrow \mathcal{F}(M)$
$M_1 M_2 \cdots M_m \xleftarrow{n} M$
For $i = 1$ to $m$ do
  $Y_i \leftarrow E(T, i) \oplus M_i$
$Y_0 \leftarrow f(T)$
Ret $Y_0 \,||\, Y_1 \,||\, \cdots \,||\, Y_m$

Algorithm $\mathcal{F}(M)$:
$M_1 M_2 \cdots M_l \xleftarrow{n} \mathsf{PadPF}(M)$
$X_0 \leftarrow IV1$
For $i = 1$ to $l$ do
  $X_i \leftarrow E'(M_i, X_{i-1}) \oplus X_{i-1}$
Ret $X_l$

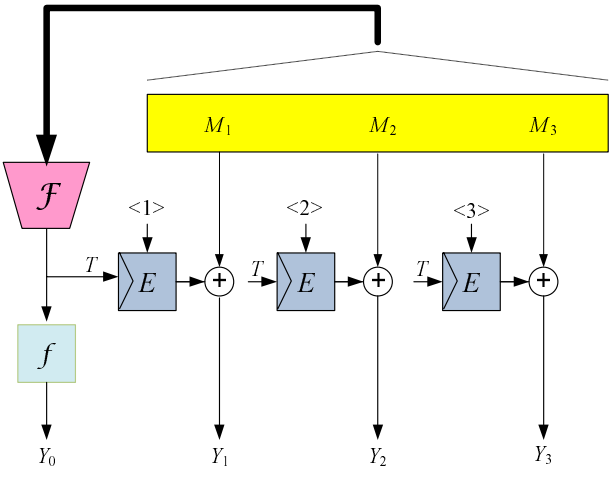


**Fig. 2.** (Left) Algorithm $\mathcal{E} = \mathsf{TE}[E, \mathcal{F}, f]$ and the description of function $\mathcal{F}$. (Right) A diagram of $\mathcal{E}$ applied to a message $M$ for which $|M| = 3n$.

much like CTR mode encryption. The cipher $E'$ will be utilized to build a function $\mathcal{F}$ for generating *tags* that will be (with high probability) unique to each input message. A message's tag then serves as the key for the CTR-mode-like enciphering step. In fact our function $\mathcal{F}$ will realize a blockcipher-based construction of a pseudorandom oracle, originally suggested in [13] and proven secure in [11]. Finally, a trapdoor one-way permutation $f$ is applied to the tag value, the result being the first portion of the output. This step ensures the injectivity of the construction, while the one-wayness "hides" the tag. We will require the trapdoor property in the proof.

Formally, we define the injection $\mathcal{E} = \mathsf{TE}[E, \mathcal{F}, f]$ by the algorithms in Figure 2. The padding function $\mathsf{PadPF}$: $\{0,1\}^* \to (\{0,1\}^n)^+$ is any prefix-free encoding function: for any two messages $M, M' \in \{0,1\}^*$ with $|M| \neq |M'|$ the string $\mathsf{PadPF}(M)$ is not a prefix of $\mathsf{PadPF}(M')$. (Such functions are simple, one example is to unambiguously pad $M$ to sequence of $n-1$ bit blocks. Then append a zero to all the blocks except the last, to which a one is appended.) The domain of $\mathcal{E}$ is $\mathcal{M} = \{0,1\}^{\leq L'}$ where $L' = n \cdot 2^{128}$. It maps a string $X$ to a string of length $|X| + k$.

THE SECURITY OF TE. To analyze the security of $\mathsf{TE}$, we start by treating the function $\mathcal{F}$ as a random oracle. This is justified by the proof that $\mathcal{F}$ is a PRO, found in [11], and the composability guarantees of the indifferentiability framework established in [23]. Thus from now on $\mathcal{F} = \mathsf{RF}_{\mathcal{M},n}$. We overload our notation to define $\mathsf{TE}$ in terms of idealized components. Let $E$ be an ideal cipher, i.e. $E \xleftarrow{\$} \mathrm{BC}(k,n)$, let $\mathcal{F} = \mathsf{RF}_{\mathcal{M},n}$, and let $\mathsf{TGen}$ be the trusted setup oracle described in Section 2. Now let $\mathcal{E} = \mathsf{TE}[E,\mathcal{F},\mathsf{TGen}]$ be the ITM that follows Algorithm $\mathcal{E}$ of Figure 2 except it utilizes $\mathsf{TGen}$ to get $f$ initially and queries $E$ and $\mathcal{F}$ oracles where appropriate. The next theorem captures the main result of this section.

**Theorem 3.** *Let $E$: $\{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$ be an ideal cipher, $\mathcal{F} = \mathsf{RF}_{\mathcal{M},k}$, and $\mathsf{TGen}$ be the oracle described above. Let $\mathcal{E} = \mathsf{TE}[E,\mathcal{F},\mathsf{TGen}]$. Let $\mathcal{A}$ be an adversary that asks at most $(q_1,q_2,q_3,q_4,1)$ oracle queries, each of length at most $\mu$ bits, and runs in time at most $t$. Then there exists an adversary $\mathcal{C}$ such that*

$$\mathbf{Adv}^{\mathrm{prio}}_{\mathcal{E},\mathcal{S}}(\mathcal{A}) \le q_1 \mathbf{Adv}^{\mathrm{owf}}_{\mathbb{F}}(\mathcal{C}) + \frac{(q_1\sigma + q_2 + q_3)^2}{2^n} + \frac{q_1^2 - q_1}{2^{k+1}}$$

*where $\sigma = \lceil \mu/n \rceil$ and $\mathcal{S}$, the simulator defined in Figure 3, runs in time at most $t_{\mathcal{S}} \le c(\mu + q_2 q_4)$ for some absolute constant $c$ and makes $q_{\mathcal{S}} = q_4$ queries. Adversary $\mathcal{C}$ runs in time $t' \le t + t_{\mathcal{S}} + (q_2+q_4)\mathsf{Time}_f + (q_1+q_2+q_4)\log(q_1+q_2+q_4) + c'\mu$ for some absolute constant $c'$.* ∎

A proof of the theorem is provided in the full version of the paper [31], here we just provide a brief proof sketch. An adversary is given either the oracles $(\mathcal{E}, E, D, \mathcal{F}, \mathsf{TGen})$ or the oracles $(\mathcal{I}, \mathcal{S}_E, \mathcal{S}_D, \mathcal{S}_{\mathcal{F}}, \mathcal{S}_{\mathsf{TGen}})$. Recall that $D$ is the oracle implementing the inverse of $E$. Intuitively the structure of $\mathsf{TE}$ ensures that an adversary, attempting to discover information about the tag and via it the random pad created for some message $M$, must reveal $M$ to the simulator (by querying the fourth oracle). Knowing $M$, the simulator can 'program' the random pad to be consistent with output of the ideal injection $\mathcal{I}$.

The simulator will fail if either of two events occurs. The first event corresponds to when two tags collide in the course of simulating the construction. If this happens the CTR mode must generate the same pad, and no longer hides relationships between input and output bits. Such an event will occur with low probability because $\mathcal{F}$ is a RO. The second kind of event is if the adversary infers a tag value without utilizing its fourth oracle ($\mathcal{F}$ or $\mathcal{S}_{\mathcal{F}}$). If it can do so, then it can compute the pad using the second oracle ($E$ or $\mathcal{S}_{\mathcal{E}}$) before the simulator knows the message the tag corresponds to. This event should happen with low probability because it requires the adversary inverts $f$ on some image returned as the first $k$ bits of a query to the first oracle. We can bound the probability of $\mathcal{A}$ inverting $f$ on some point in terms of its ability to invert on a particular point (hence the $q_1\mathbf{Adv}^{\mathrm{owf}}_{\mathbb{F}}(\mathcal{C})$ term). Since neither event occurs with high probability, we achieve a bound on the adversary's ability to differentiate the two sets of oracles.

DISCUSSION. One might wonder if we can dispense with the one way permutation. In fact it is requisite: omitting it would result in a construction easily differentiable

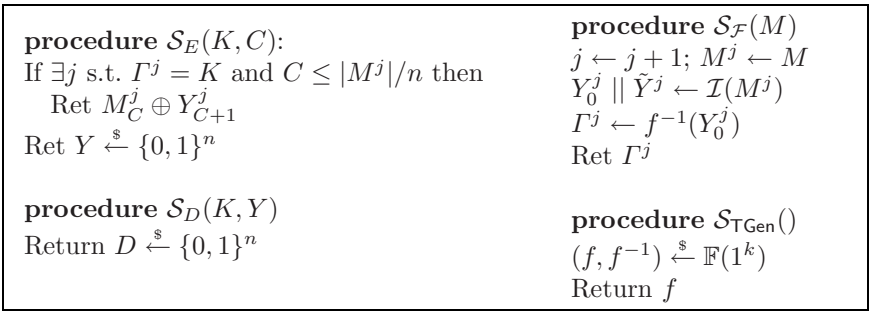


**Fig. 3.** The simulator $\mathcal{S}$ used in proof of Theorem 3. Initially $j = 0$.

from a random injective oracle. An adversary could simply query its first oracle on a random message $M_1$, receiving $T \,||\, Y_1$. Then the adversary could query its third oracle (either $D$ or $\mathcal{S}_D$) on $(T, Y_1)$. At this point the simulator has no knowledge about $M_1$ and will therefore only respond correctly with low probability.

The TE construction is a proof-of-concept: it is the first object to achieve our new goal of being simultaneously constructively injective and indifferentiable from a random injection. On the other hand it has several drawbacks when considering it for practical use. It is length-increasing (outputs are larger than the inputs by at least the number of key bits of the underlying blockcipher). This means that when utilized in MCM the output hash values will be larger compared to the outputs of the provably CR function $H$. Further, the construction requires two passes over the data and the application of a trapdoor permutation. In settings where speed is not essential (e.g., contract signing), the extra expense of using TE over that already incurred by hashing with a standard-model, provably collision-resistant function $H$ might not be prohibitive. All this said, the TE construction *does* show that the MCM approach is feasible. We hope that future research will surface improvements.

## 6 Composability Limitations and Open Problems

Recall that the key benefit of indifferentiability results is the guarantee of composability, as discussed in depth in [23]. For example, a cryptographic scheme $\mathcal{E}$ proven secure when utilizing a (monolithic) random oracle $\mathcal{R}$ remains secure if the random oracle is replaced by a PRO construction $C$. When we say "remains secure" we mean that the existence of an adversary breaking the security of $\mathcal{E}^{\mathcal{R}}$ implies the existence of an adversary that breaks the security of $\mathcal{E}^{C}$. This means we can safely argue about the security of $\mathcal{E}^{C}$ in two steps: show that $C$ is indifferentiable from $\mathcal{R}$ and then that $\mathcal{E}^{\mathcal{R}}$ is secure. Enabling this approach is a significant benefit of simulation-based definitions (the UC framework is another example [8]). Our results also allow for secure composition, but with important (and perhaps subtle) qualifications.

First, we note that both Theorem 2 and Theorem 3 differ from previous indifferentiability results because they are complexity-theoretic in nature. Specifically, the indifferentiability of MCM from a random oracle (Theorem 2) relies on an adversary's inability to find collisions under $H$. The indifferentiability of TE from a random injection (Theorem 3) relies on an adversary's inability to invert the trapdoor permutation $f$. We must bound the computational power of the adversary in both results, since an unbounded adversary can *always* find collisions against $H$ or invert $f$. This means that $\mathcal{E}^{\mathsf{MCM}}$, for example, is secure *only* against computationally-bounded adversaries, even if $\mathcal{E}^{\mathcal{R}}$ is information-theoretically secure. This is a problem for random-oracle-based constructions $\mathcal{E}$ that require information-theoretic security (see, e.g. [7]).

Second, Theorem 3 relies on a simulator that knows the trapdoor of the one-way permutation (i.e., it gets to control generation of the permutation). Effectively then, instantiating TE requires a trusted party to publish a description of $f$, which can be considered a common reference string (CRS). We allow the simulator to choose the CRS in the proof. Recent results by Pass and Canetti et al. [6,9] call into question the (wide) use of such powerful simulators, in that composability of some security properties might be lost. For example, Pass discusses how deniability of non-interactive zero-knowledge proofs (the prover can assert that he never even proved a statement) does not hold if the proof relies on the zero-knowledge simulator choosing the CRS [6]. Indeed interpreting the composability theorem for the indifferentiability framework [23, Thm. 1] in the context of TGen implies that some security properties (e.g., deniability) of constructions using TE will not hold in settings where other parties are allowed to know $f$.

These subtle nuances of our results lead to a host of provocative open questions. What other properties, beyond deniability, are compromised by the weak composability guarantees of TE? Is it (im)possible to build PRIOs without relying on such strong simulators? Can we strengthen the MCM security result, or find other constructions, that simultaneously are provably CR and yet have information-theoretic indifferentiability from a RO?

## Acknowledgments

The authors thank Yevgeniy Dodis for illuminating discussions regarding composability and deniability and the anonymous reviewers for their valuable comments. The first author is supported in part by Mihir Bellare's NSF grant CNS 0524765 and his gift from Intel Corporation. The second author is supported by NSF grant CNS 0627752.

## References

1. Bellare, M., Boldyreva, A., Palacio, A.: An Uninstantiable Random-Oracle-Model Scheme for a Hybrid-Encryption Problem. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 171–188. Springer, Heidelberg (2004)

2. 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)
3. Bellare, M., Ristenpart, T.: Hash Functions in the Dedicated-key Setting: Design Choices and MPP Transforms. In: ICALP 2007. International Colloquim on Automata, Languages, and Programming. LNCS, vol. 4596, pp. 399–410. Springer, Heidelberg (2007)
4. 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)
5. Black, J., Rogaway, P., Shrimpton, T.: Black-Box Analysis of the Block-Cipher-Based Hash-Function Constructions from PGV. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 320–325. Springer, Heidelberg (2002)
6. Pass, R.: On deniability in the common reference String and Random Oracle model. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 316–337. Springer, Heidelberg (2003)
7. Boyen, X., Dodis, Y., Katz, J., Ostrovsky, R., Smith, A.: Secure Remote Authentication Using Biometric Data. In: Cramer, R.J.F. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 147–163. Springer, Heidelberg (2005)
8. Canetti, R.: Universally Composable Security: A New Paradigm for Cryptographic Protocols. In: FOCS 2001. Symposium on Foundations of Computer Science, pp. 136–145. IEEE Computer Society, Los Alamitos (2001)
9. Canetti, R., Dodis, Y., Pass, R., Walfish, S.: Universally Composable Protocols with Global Set-up. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, pp. 61–85. Springer, Heidelberg (2007)
10. Canetti, R., Goldreich, O., Halevi, S.: The random oracle methodology, revisited. J. ACM 51(4), 557–594 (2004)
11. Chang, D., Lee, S., Nandi, M., Yung, M.: Indifferentiable Security Analysis of Popular Hash Functions with Prefix-Free Padding. In: Lai, X., Chen, K. (eds.) ASIACRYPT 2006. LNCS, vol. 4284, pp. 283–298. Springer, Heidelberg (2006)
12. Contini, S., Lenstra, A., Steinfeld, R.: VSH, an Efficient and Provable Collision-Resistant Hash Function. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 165–182. Springer, Heidelberg (2006)
13. Coron, J.S., Dodis, Y., Malinaud, C., Puniya, P.: Merkle-Damgard Revisited: How to Construct a Hash Function. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 21–39. Springer, Heidelberg (2005)
14. Damgård, I.: hash functions and public key signature schemes. In: Price, W.L., Chaum, D. (eds.) EUROCRYPT 1987. LNCS, vol. 304, pp. 416–427. Springer, Heidelberg (1988)
15. Damgård, I.: A design principle for hash functions. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 416–427. Springer, Heidelberg (1990)
16. Dodis, Y., Puniya, P.: Feistel networks made public, and applications. In: Advances in Cryptology– EUROCRYPT 2007. LNCS, vol. 4515, pp. 534–554. Springer, Heidelberg (2007)
17. Dodis, Y., Puniya, P.: On the relation between the ideal cipher and random oracle models. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, pp. 184–206. Springer, Heidelberg (2006)
18. Goldwasser, S., Micali, S.: Probabilistic Encryption. Journal of Computer and System Sciences 28, 270–299 (1984)

19. Halevi, S.: EME*: Extending EME to handle arbitrary-length messages with associated data. In: Canteaut, A., Viswanathan, K. (eds.) INDOCRYPT 2004. LNCS, vol. 3348, pp. 315–327. Springer, Heidelberg (2004)
20. Halevi, S., Rogaway, P.: A parallelizable enciphering mode. In: Okamoto, T. (ed.) CT-RSA 2004. LNCS, vol. 2964, pp. 292–304. Springer, Heidelberg (2004)
21. Halevi, S., Rogaway, P.: A tweakable enciphering mode. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 482–499. Springer, Heidelberg (2003)
22. Lyubashevsky, V., Micciancio, D., Peikert, C., Rosen, A.: Provably secure FFT hashing. In: NIST 2nd Cryptographic Hash Function Workshop (2006)
23. Maurer, U., 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)
24. Merkle, R.: One way hash functions and DES. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 428–446. Springer, Heidelberg (1990)
25. National Institute of Standards and Technology.FIPS PUB 180-1: Secure Hash Standard. Supersedes FIPS PUB 180 1993 May 11(1995)
26. Pointcheval, D.: The Composite Discrete Logarithm and Secure Authentication. In: Imai, H., Zheng, Y. (eds.) PKC 2000. LNCS, vol. 1751, pp. 113–128. Springer, Heidelberg (2000)
27. Preneel, B., Govaerts, R., Vandewalle, J.: Hash functions based on block ciphers: A synthetic approach. In: Stinson, D.R. (ed.) CRYPTO 1993. LNCS, vol. 773, pp. 368–378. Springer, Heidelberg (1994)
28. Rabin, M.: Digital signatures. In: Millo, R.A., et al. (eds.) Foundations of secure computation, Academic Press, London (1978)
29. Rabin, M.: Digital signatures and public key functions as intractable as factorization. MIT Laboratory for Computer Science Report TR-212 (January 1979)
30. Ristenpart, T., Rogaway, P.: How to Enrich the Message Space of a Cipher. In: Fast Software Encryption– FSE 2007. LNCS, vol. 4593, pp. 101–118. Springer, Heidelberg (2007)
31. Ristenpart, T., Shrimpton, T.: How to Build a Hash Function from any Collision-Resistant Function (full version of this paper), `http://www.cse.ucsd.edu/users/tristenp/`
32. Rivest, R., Shamir, A., Adleman, L.: A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. Communications of the ACM 21(2), 120–126 (1978)
33. Rivest, R., Tauman, Y.: Improved online/offline signature schemes. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 355–367. Springer, Heidelberg (2001)
34. Rogaway, P., Shrimpton, T.: Cryptographic Hash-Function Basics: Definitions, Implications, and Separations for Preimage Resistance, Second-Preimage Resistance, and Collision Resistance. In: Roy, B., Meier, W. (eds.) FSE 2004. LNCS, vol. 3017, pp. 371–388. Springer, Heidelberg (2004)
35. RSA Laboratories, RSA PKCS #1 v2.1: RSA Cryptography Standards (2002)
36. Saarinen, M.: Security of VSH in the Real World. In: Barua, R., Lange, T. (eds.) INDOCRYPT 2006. LNCS, vol. 4329, pp. 95–103. Springer, Heidelberg (2006)
37. Wang, X., Yin, Y.L., Yu, H.: Finding Collisions in the Full SHA-1. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 17–36. Springer, Heidelberg (2005)
38. Wang, X., Yu, H.: How to Break MD5 and Other Hash Functions. In: Cramer, R.J.F. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 19–35. Springer, Heidelberg (2005)
39. Whitfield, D., Hellman, M.: Privacy and Authentication: An Introduction to Cryptography. Proceedings of the IEEE 67, 397–427 (1979)