Multi-property-preserving Hash Domain Extension and the EMD Transform

Mihir Bellare and Thomas Ristenpart

Dept. of Computer Science & Engineering 0404, University of California San Diego
9500 Gilman Drive, La Jolla, CA 92093-0404, USA
{mihir,tristenp}@cs.ucsd.edu
http://www-cse.ucsd.edu/users/{mihir,tristenp}

Abstract. We point out that the seemingly strong *pseudorandom oracle preserving* (PRO-Pr) property of hash function domain-extension transforms defined and implemented by Coron et. al. [1] can actually *weaken* our guarantees on the hash function, in particular producing a hash function that fails to be even collision-resistant (CR) even though the compression function to which the transform is applied is CR. Not only is this true in general, but we show that *all* the transforms presented in [1] have this weakness. We suggest that the appropriate goal of a domain extension transform for the next generation of hash functions is to be multi-property preserving, namely that one should have a *single* transform that is simultaneously at least collision-resistance preserving, pseudorandom function preserving and PRO-Pr. We present an efficient new transform that is proven to be multi-property preserving in this sense.

1 Introduction

Background. Recall that hash functions are built in two steps. First, one designs a compression function $h\colon \{0,1\}^{d+n} \to \{0,1\}^n$, where d is the length of a data block and n is the length of the chaining variable. Then one specifies a *domain extension transform* H that utilizes h as a black box to implement the hash function $H^h\colon \{0,1\}^* \to \{0,1\}^n$ associated to h. All widely-used hash functions use the Merkle-Damgård (MD) transform [2, 3] because it has been proven [2, 3] to be *collision-resistance preserving* (CR-Pr): if h is collision-resistant (CR) then so is H^h. This means that the cryptanalytic validation task can be confined to the compression function.

A rising bar. Current usage makes it obvious that CR no longer suffices as the security goal for hash functions. In order to obtain MACs and PRFs, hash functions were keyed. The canonical construct in this domain is HMAC [4, 5] which is widely standardized and used. (NIST FIPS 198, ANSI X9.71, IETF RFC 2104, SSL, SSH, IPSEC, TLS, IEEE 802.11i, and IEEE 802.16e are only some instances.) Hash functions are also used to instantiate random oracles [6] in public-key schemes such as RSA-OAEP [7] and RSA-PSS [8] in the RSA

PKCS#1 v2.1 standard [9]. CR is insufficient for arguing the security of hash function based MACs or PRFs, let alone hash-function based random oracles. And it does not end there. Whether hash function designers like it or not, application builders will use hash functions for all kinds of tasks that presume beyond-CR properties. Not all such uses can be sanctified, but the central and common ones should be. We think that the type of usage we are seeing for hash functions will continue, and it is in the best interests of security to make the new hash functions rise as far towards this bar as possible, by making them strong and versatile tools that have security attributes beyond CR.

THIS PAPER. Towards the goal of building strong, multi-purpose hash functions, our focus is on domain extension, meaning we wish to determine which domain extension transforms are best suited to this task. The first part of our work examines a natural candidate, namely transforms that are *pseudorandom oracle preserving* as per [1], and identifies some weaknesses of this goal. This motivates the second part, where we introduce the notion of a *multi-property preserving (MPP) transform*, argue that this should be the target goal, and present and prove the correctness of an efficient MPP transform that we refer to as EMD. Let us now look at all this in more depth.

RANDOM-ORACLE PRESERVATION. Coron, Dodis, Malinaud and Puniya [1] make the important observation that random oracles are modeled as monolithic entities (i.e., are black boxes working on domain $\{0,1\}^*$), but in practice are instantiated by hash functions that are highly structured due to the design paradigm described above, leading for example to the extension attack. Their remedy for this logical gap is to suggest that a transform H be judged secure if, when modeling h as a fixed-input-length random oracle, the resulting scheme H^h behaves like a random oracle. They give a formal definition of "behaving like a random oracle" using the indifferentiability framework of Maurer et al. [10]. We use the moniker *pseudorandom oracle* to describe any construction that is indifferentiable from a random oracle. (Note that a random oracle itself is always a pseudorandom oracle.) The framework has the desirable property that any scheme proven secure in the random oracle model of [6] is still secure when we replace the random oracles with pseudorandom oracles. We call the new security goal of [1] *pseudorandom oracle preservation* (PRO-Pr). They propose four transforms which they prove to be PRO-Pr.

PRO-Pr seems like a very strong property to have. One reason one might think this is that it appears to automatically guarantee that the constructed hash function has many nice properties. For example, that the hash function created by a PRO-Pr transform would be CR. Also that the hash function could be keyed in almost any reasonable way to yield a PRF and MAC. And so on. This would be true, because random oracles have these properties, and hence so do pseudorandom oracles. Thus, one is lead to think that one can stop with PRO-Pr: once the transform has this property, we have all the attributes we desire from the constructed hash function.

Weakness of PRO-Pr. The first contribution of this paper is to point out that the above reasoning is flawed and there is a danger to PRO-Pr in practice. Namely, the fact that a transform is PRO-Pr *does not guarantee* that the constructed hash function is CR, even if the compression function is CR. We demonstrate this with a counter-example. Namely we give an example of a transform that is PRO-Pr, yet there is a CR compression function such that the hash function resulting from the transform is not CR. That is, the transform is PRO-Pr but not CR-Pr, or, in other words, PRO-Pr does not imply CR-Pr. What this shows is that using a PRO-Pr transform could be *worse* than using the standard, strengthened Merkle-Damgård transform from the point of view of security because at least the latter guarantees the hash function is CR if the compression function is, but the former does not. If we blindly move to PRO-Pr transforms, our security guarantees are actually going down, not up.

How can this be? It comes about because PRO-Pr provides guarantees only if the compression function is a random oracle or pseudorandom oracle. But of course any real compression function is *provably not* either of these. (One can easily differentiate it from a random oracle because it can be computed by a small program.) Thus, when a PRO-Pr transform works on a real compression function, we have essentially no provable guarantees on the resulting hash function. This is in some ways analogous to the kinds of issues pointed out in [11, 12] about the sometimes impossibility of instantiating random oracles.

The transforms of [1] are not CR-Pr. The fact that a PRO-Pr transform need not in general be CR-Pr does not mean that some *particular* PRO-Pr transform is not CR-Pr. We therefore investigate each of the four PRO-Pr schemes suggested by [1]. The schemes make slight modifications to the MD transform: the first applies a prefix-free encoding, the second "throws" away some of the output, and the third and fourth utilize an extra compression function application. Unfortunately, we show that none of the four transforms is CR-Pr. We do this by presenting an example CR compression function h such that applying each of the four transforms to it results in a hash function for which finding collisions is trivial. In particular, this means that these transforms do not provide the same guarantee as the existing and in-use Merkle-Damgård transform. For this reason we think these transforms should not be considered suitable for use in the design of new hash functions.

What this means. We clarify that we are *not* suggesting that the pseudorandom oracle preservation goal of [1] is unimportant or should not be achieved. In fact we think it is a very good idea and should be a property of any new transform. This is so because in cases where we are (heuristically) assuming the hash function is a random oracle, this goal reduces the assumption to the compression function being a random oracle. What we have shown above, however, is that *by itself*, it is not enough because it can weaken existing, standard-model guarantees. This leads to the question of what exactly *is* enough, or what we should ask for in terms of a goal for hash domain extension transforms.

MPP TRANSFORMS. The two-step design paradigm in current use is compelling because it reduces the cryptanalytic task of providing CR of the hash function to certifying only that the compression function has the same property. It makes sense to seek other attributes via the appropriate extension of this paradigm. We suggest that, if we want a hash function with properties $\mathsf{P}_1, \ldots, \mathsf{P}_n$ then we should (1) design a compression function h with the goal of having properties $\mathsf{P}_1, \ldots, \mathsf{P}_n$, and (2) apply a domain extension transform H that *provably* preserves P_i for every $i \in [1..n]$. We call such a compression function a multi-property one, and we call such a transform a *multi-property-preserving domain extension transform* (from now on simply an MPP transform). Note that we want a *single* transform that preserves multiple properties, resulting in a single, multi-property hash function, as opposed to a transform per property which would result in not one but numerous hash functions. We suggest that multi-property preservation is the goal a transform should target.

PROPERTIES TO PRESERVE. Of course the next question to ask is which properties our MPP domain extension transform should preserve. We wish, of course, that the transform continue to be CR-Pr, meaning that it preserve CR. The second thing we ask is that it be pseudorandom function preserving (PRF-Pr). That is, if an appropriately keyed version of the compression function is a PRF then the appropriately keyed version of the hash function must be a PRF too. This goal is important due to the many uses of hash functions as MACs and PRFs via keying as mentioned above. Indeed, if we have a compression function that can be keyed to be a PRF and our transform is PRF-Pr then obtaining a PRF or MAC from a hash function will be simple and the construction easy to justify. The final goal we will ask is that the transform be PRO-Pr. Compelling arguments in favor of this goal were made at length in [1] and briefly recalled above.

To be clear, we ask that, for a transform H to be considered suitable, one should do the following. First, prove that H^h is CR using only the fact that h is CR. Then show that H^h is a pseudorandom oracle when h is a pseudorandom oracle. Finally, use some natural keying strategy to key H^h and assume that h is a good PRF, then prove that H^h is also a good PRF. We note that such a MPP transform will not suffer from the weakness of the transforms of [1] noted above because it will be not only PRO-Pr but also CR-Pr and PRF-Pr.

NEW TRANSFORM. There is to date no transform with all the properties above. (Namely, that it is PRO-Pr, CR-Pr and PRF-Pr.) The next contribution of this paper is a new transform EMD (Enveloped Merkle-Damgård) which is the first to meet our definition of hash domain extension security: EMD is proven to be CR-Pr, PRO-Pr, and PRF-Pr. The transform is simple and easy to implement in practice (see the figure in Section 5). It combines two mechanisms to ensure that it preserves all the properties of interest. The first mechanism is the well-known Merkle-Damgård strengthening [2]: we always concatenate an input message with the 64-bit encoding of its length. This ensures that EMD is CR-Pr. The second mechanism is the use of an "envelope" to hide the internal MD iteration — we apply the compression function in a distinguished way to the output of the plain MD iteration. Envelopes in this setting were previously used by the NMAC and

Transform	CR-Pr	PRO-Pr	PRF-Pr	Uses of h for $\|M\| = b \geq d$
Plain MD (MD)	No	No	No	$\lceil (b+1)/d \rceil$
Strengthened MD (SMD)	[2, 3]	No	No	$\lceil (b+1+64)/d \rceil$
Prefix-Free (PRE)	No	[1]	[13]	$\lceil (b+1)/(d-1) \rceil$
Chop Solution (CHP)	No	[1]	?	$\lceil (b+1)/d \rceil$
NMAC Construction (NT)	No	[1]	?	$1 + \lceil (b+1)/d \rceil$
HMAC Construction (HT)	No	[1]	?	$2 + \lceil (b+1)/d \rceil$
Enveloped MD (EMD)	[2]	Thm. 1	Thm. 2	$\lceil (b+1+64+n)/d \rceil$

Fig. 1. Comparison of transform security and efficiency when applied to a compression function h: $\{0,1\}^{n+d} \to \{0,1\}^n$. The last column specifies the number of calls to h needed to hash a b-bit message M (where $b \geq d$) under each transform and a typical padding function (which minimally adds a bit of overhead).

HMAC constructions [4] to build PRFs out of compression functions, and again in two of the PRO-Pr transforms of [1], which were also based on NMAC and HMAC. We utilize the envelope in a way distinct from these prior constructions. Particularly, we include message bits as input to the envelope, which increases the efficiency of the scheme. Second, we utilize a novel reduction technique in our proof that EMD is PRO-Pr to show that simply fixing n bits of the envelope's input is sufficient to cause the last application of the random oracle to behave independently with high probability. This simple solution allows our transform to be PRO-Pr using a single random oracle without using the other work-arounds previously suggested (e.g., prefix-free encodings or prepending a block of zeros to input messages). A comparison of various transforms is given in Fig. 1.

Patching existing transforms. We remark that it is possible to patch the transforms of [1] so that they are CR-Pr. Namely, one could use Merke-Damgård strengthening, which they omitted. However our transform still has several advantages over their transforms. One is that ours is cheaper, i.e. more efficient, and this matters in practice. Another is that ours is PRF-Pr. A result of [13] implies that one of the transforms of [1] is PRF-Pr, but whether or not this is true for the others is not clear.

Whence the compression function? We do not address the problem of constructing a multi-property compression function. We presume that this can and will be done. This assumption might seem questionable in light of the recent collision-finding attacks [14, 15] that have destroyed some hash functions and tainted others. But we recall that for block ciphers, the AES yielded by the NIST competition was not only faster than DES but seems stronger and more elegant. We believe it will be the same for compression functions, namely that the planned NIST hash function competition will lead to compression functions having the properties (CR and beyond) that we want, and perhaps without increase, or even with decrease, in cost, compared to current compression functions. We also note that we are not really making new requirements on the compression function; we are only making explicit requirements that are implicit even in current usage.

Families of compression functions. Several works [16–18] consider a setting where compression and hash functions are families rather than individual functions, meaning, like block ciphers, have an extra, dedicated key input. In contrast, we, following [4, 1, 5], adopt the setting of current practical cryptographic compression and hash functions where there is no such dedicated key input. An enveloping technique similar to that of EMD is used in the Chain-Shift construction of Maurer and Sjödin [18] for building a VIL MAC out of a FIL MAC in the dedicated key input setting. We further discuss this setting, and their work, in the full version of the paper [19].

2 Definitions

Notation. Let $D = \{0,1\}^d$ and $D^+ = \cup_{i\geq 1}\{0,1\}^{id}$. We denote pairwise concatenation by $||$, e.g. $M \,||\, M'$. We will often write the concatenation of a sequence of string by $M_1 \cdots M_k$, which translates to $M_1 \,||\, M_2 \,||\, \ldots \,||\, M_k$. For brevity, we define the following semantics for the notation $M_1 \cdots M_k \stackrel{d}{\leftarrow} M$ where M is a string of $|M|$ bits: 1) define $k = \lceil |M|/d \rceil$ and 2) if $|M| \bmod d = 0$ then parse M into M_1, M_2, …, M_k where $|M_i| = d$ for $1 \leq i \leq k$, otherwise parse M into M_1, M_2, …, M_{k-1}, M_k where $|M_i| = d$ for $1 \leq i \leq k-1$ and $|M_k| = |M| \bmod d$. For any finite set S we write $s \stackrel{\$}{\leftarrow} S$ to signify uniformly choosing a value $s \in S$.

Oracle TMs, random oracles, and transforms. Cryptographic schemes, adversaries, and simulators are modeled as Oracle Turing Machines (OTM) and are possibly given zero or more oracles, each being either a random oracle or another OTM (note that when used as an oracle, an OTM maintains state across queries). We allow OTMs to expose a finite number of interfaces: an OTM $\mathsf{N} = (\mathsf{N}_1, \mathsf{N}_2, \ldots, \mathsf{N}_l)$ exposes interfaces $\mathsf{N}_1, \mathsf{N}_2, \ldots, \mathsf{N}_l$. For brevity, we write M^{N} to signify that M gets to query all the interfaces of N. For a set *Dom* and finite set *Rng* we define a *random function* by the following TM accepting inputs $X \in \mathit{Dom}$:

Algorithm $\mathsf{RF}_{\mathit{Dom},\mathit{Rng}}(X)$:
  if $T[X] = \perp$ **then** $T[X] \stackrel{\$}{\leftarrow} \mathit{Rng}$
  ret $T[X]$

where T is a table everywhere initialized to $\perp$. This implements a random function via lazy sampling (which allows us to reason about the case in which *Dom* is infinite). In the case that $\mathit{Dom} = \{0,1\}^d$ and $\mathit{Rng} = \{0,1\}^r$ we write $\mathsf{RF}_{d,r}$ in place of $\mathsf{RF}_{\mathit{Dom},\mathit{Rng}}$. We similarly define $\mathsf{RF}_{d,\mathit{Rng}}$ and $\mathsf{RF}_{\mathit{Dom},r}$ in the obvious ways and write $\mathsf{RF}_{*,r}$ in the special case that $\mathit{Dom} = \{0,1\}^*$. A *random oracle* is simply a public random function: all parties (including the adversary) are given access. We write $f, g, \ldots = \mathsf{RF}_{\mathit{Dom},\mathit{Rng}}$ to signify that f, g, … are independent random oracles from *Dom* to *Rng*. A *transform* C describes how to utilize an arbitrary compression function to create a variable-input-length hash function. When we fix a particular compression function f, we get the associated cryptographic scheme $C^f \equiv C[f]$.

Collision resistance. We consider a function F to be collision resistant (CR) if it is computationally infeasible to find any two messages $M \neq M'$ such that $F(M) = F(M')$. For the rest of the paper we use h to always represent a collision-resistant compression function with signature $h\colon \{0,1\}^{d+n} \to \{0,1\}^n$.

Note our definition of CR is informal. The general understanding in the literature is that a formal treatment requires considering keyed families. But practical compression and hash functions are not keyed when used for CR. (They can be keyed for use as MACs or PRFs.) And in fact, our results on CR are still formally meaningful because they specify explicit reductions.

PRFs. Let $F\colon \mathit{Keys} \times \mathit{Dom} \to \mathit{Rng}$ be a function family. Informally, we consider F a pseudorandom function family (PRF) if no reasonable adversary can succeed with high probability at distinguishing between $F(K,\cdot)$ for $K \xleftarrow{\$} \mathit{Keys}$ and a random function $f = \mathsf{RF}_{\mathit{Dom},\mathit{Rng}}$. More compactly we write the *prf-advantage* of an adversary A as

$$\mathbf{Adv}_F^{\mathrm{prf}}(A) = \Pr\left[K \xleftarrow{\$} \mathit{Keys}; A^{F(K,\cdot)} \Rightarrow 1\right] - \Pr\left[A^{f(\cdot)} \Rightarrow 1\right]$$

where the probability is taken over the random choice of K and the coins used by A or by the coins used by f and A. For the rest of the paper we use e to always represent a PRF with signature $e\colon \{0,1\}^{d+n} \to \{0,1\}^n$ that is keyed through the low n bits of the input.

PROs. The indifferentiability framework [10] generalizes the more typical indistinguishability framework (e.g., our definition of a PRF above). The new framework captures the necessary definitions for comparing an object that utilizes public components (e.g., fixed-input-length (FIL) random oracles) with an ideal object (e.g., a variable-input-length (VIL) random oracle). Fix some number l. Let $C^{f_1,\ldots,f_l}\colon \mathit{Dom} \to \mathit{Rng}$ be a function for random oracles $f_1,\ldots,f_l = \mathsf{RF}_{D,R}$. Then let $S^{\mathcal{F}} = (S_1,\ldots,S_l)$ be a simulator OTM with access to a random oracle $\mathcal{F} = \mathsf{RF}_{\mathit{Dom},\mathit{Rng}}$ and which exposes interfaces for each random oracle utilized by C. (The simulator's goal is to mimic $f_1,\ldots,f_l$ in such a way as to convince an adversary that $\mathcal{F}$ is C.) The *pro-advantage* of an adversary A against C is the difference between the probability that A outputs a one when given oracle access to $C^{f_1,\ldots,f_l}$ and $f_1,\ldots,f_l$ and the probability that A outputs a one when given oracle access to $\mathcal{F}$ and $S^{\mathcal{F}}$. More succinctly we write that the pro-advantage of A is

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

where, in the first case, the probability is taken over the coins used by the random oracles and A and, in the second case, the probability is over the coins used by the random oracles, A, and S. For the rest of the paper we use f to represent a random oracle $\mathsf{RF}_{d+n,n}$.

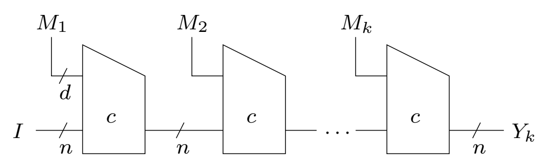
Resources. We give concrete statements about the advantage of adversaries using certain resources. For prf-adversaries we measure the total number of queries q made and the running time t. For pro-adversaries we measure the total number of *left queries* q_L (which are either to C or $\mathcal{F}$) and the number of *right queries* q_i made to each oracle f_i or simulator interface S_i. We also specify the resources utilized by simulators. We measure the total number of queries q_S to $\mathcal{F}$ and the maximum running time t_S. Note that these values are generally functions of the number of queries made by an adversary (necessarily so, in the case of t_S).

Pointless queries. In all of our proofs (for all notions of security) we assume that adversaries make no *pointless queries*. In our setting this particularly means that adversaries are never allowed to repeat a query to an oracle.

3 Domain Extension using Merkle-Damgård

The Merkle-Damgård transform. We focus on variants of the Merkle-Damgård transform. Let c: $\{0,1\}^{d+n} \to \{0,1\}^n$ be an arbitrary fixed-input-length function. Using it, we wish to construct a family of variable-input-length functions F^c: $\{0,1\}^n \times \{0,1\}^* \to \{0,1\}^n$. We start by defining the Merkle-Damgård iteration c^+: $D^+ \to \{0,1\}^n$ by the algorithm specified below.

Algorithm $c^+(I, M)$:
$M_1 \cdots M_k \xleftarrow{d} M$; $Y_0 \leftarrow I$
for $i = 1$ to k **do**
  $Y_i \leftarrow c(M_i \,||\, Y_{i-1})$
ret Y_k



Since I is usually fixed to a constant, the function c^+ only works for strings that are a multiple of d bits. Thus we require a padding function $\mathsf{pad}(M)$, which for any string $M \in \{0,1\}^*$ returns a string Y for which $|Y|$ is a multiple of d. We require that pad is one-to-one (this requirement is made for all padding functions in this paper). A standard instantiation for pad is to append to the message a one bit and then enough zero bits to fill out a block. Fixing some $IV \in \{0,1\}^n$, we define the *plain Merkle-Damgård transform* $\mathrm{MD}[c] = c^+(IV, \mathsf{pad}(\cdot))$.

Keying strategies. In this paper we discuss transforms that produce keyless schemes. We would also like to utilize these schemes as variable-input-length PRFs, but this requires that we use some keying strategy. We focus on the *key-via-IV strategy*. Under this strategy, we replace constant initialization vectors with randomly chosen keys of the same size. For example, if e is a particular PRF, then keyed MD^e would be defined as $\mathrm{MD}^e_K(M) = e^+(K, \mathsf{pad}(M))$ (it should be noted that this is not a secure PRF). We will always signify the keyed version of a construction by explicitly including the keys as subscripts.

Multi-property preservation. We would like to reason about the security of MD and its variants when we make assumptions about c. Phrased another way, we want to know if a transform such as MD *preserves* security properties of the underlying compression function. We are interested in collision-resistance preservation, PRO preservation, and PRF preservation. Let C be a transform that works on functions from $\{0,1\}^{d+n}$ to $\{0,1\}^n$. Let h: $\{0,1\}^{d+n} \to \{0,1\}^n$

be a collision-resistant hash function. Then we say that C is *collision-resistance preserving* ($\mathsf{CR\text{-}Pr}$) if the scheme C^h is collision-resistant. Let $f = \mathsf{RF}_{d+n,n}$ be a random oracle. Then we say that C is *pseudorandom oracle preserving* ($\mathsf{PRO\text{-}Pr}$) if the scheme C^f is a pseudorandom oracle. Let $e\colon \{0,1\}^{d+n} \to \{0,1\}^n$ be an arbitrary PRF (keyed via the low n bits). Then we say that C is *pseudorandom function preserving* ($\mathsf{PRF\text{-}Pr}$) if the keyed-via-IV scheme C^e_K is a PRF. A transform for which all of the above holds is considered *multi-property preserving.*

Security of MD and SMD. It is well known that MD is neither $\mathsf{CR\text{-}Pr}$, $\mathsf{PRO\text{-}Pr}$, or $\mathsf{PRF\text{-}Pr}$ [2, 3, 13, 1]. The first variant that was proven $\mathsf{CR\text{-}Pr}$ was so-called MD with strengthening, which we denote by SMD. In this variant, the padding function is replaced by one with the following property: for M and M' with $|M| \neq |M'|$ then $M_k \neq M'_k$ (the last blocks after padding are distinct). A straightforward way to achieve a padding function with this property is to include an encoding of the message length in the padding. In many implementations, this encoding is done using 64 bits [20], which restricts the domain to strings of length no larger than 2^{64}. We therefore fix some padding function $\mathsf{pad64}(M)$ that takes as input a string M and returns a string Y of length kd bits for some number k such that the last 64 bits of Y are an encoding of $|M|$. Using this padding function we define the *strengthened MD transform* $\mathrm{SMD}[c] = c^+(IV, \mathsf{pad64}(\cdot))$. We emphasize the fact that preservation of collision-resistance is *strongly dependent* on the choice of padding function. However, this modification to MD is alone insufficient for rendering SMD either $\mathsf{PRF\text{-}Pr}$ or $\mathsf{PRO\text{-}Pr}$ due to simple length-extension attacks [13, 1].

4 Orthogonality of Property Preservation

In this section we illustrate that property preservation is orthogonal. Previous work [1] has already shown that collision-resistance preservation does not imply pseudorandom oracle preservation. We investigate the inverse: does a transform being $\mathsf{PRO\text{-}Pr}$ imply that it is also $\mathsf{CR\text{-}Pr}$? We answer this in the negative by showing how to construct a $\mathsf{PRO\text{-}Pr}$ transform that is not $\mathsf{CR\text{-}Pr}$. While this result is sufficient to refute the idea that $\mathsf{PRO\text{-}Pr}$ is a stronger security goal for transforms, it does not necessarily imply anything about specific $\mathsf{PRO\text{-}Pr}$ transforms. Thus, we investigate the four transforms proposed by Coron et al. and show that all four fail to preserve collision-resistance. Finally, lacking a formally meaningful way of comparing pseudorandom oracle preservation and pseudorandom function preservation (one resulting in keyless schemes, the other in keyed), we briefly discuss whether the proposed transforms are $\mathsf{PRF\text{-}Pr}$.

4.1 PRO-Pr does not imply CR-Pr

Let $n, d > 0$ and $h\colon \{0,1\}^{d+n} \to \{0,1\}^n$ be a collision-resistant hash function and $f = \mathsf{RF}_{d+n,n}$ be a random oracle. Let Dom, Rng be non-empty sets and let C_1 be a transform for which $C_1^f \equiv C_1[f]$ is a pseudorandom oracle $C_1^f\colon Dom \to Rng$.

procedure Initialize	**procedure** $C(X)$	$\boxed{\text{Game G0}}$ Game G1
$f = \mathsf{RF}_{d+n,n}$	$Y \leftarrow C_1^f(X)$	
procedure $f(x)$	**if** $f(0^{d+n}) = 0^n$ **then** $bad \leftarrow$ true; $\boxed{Y \leftarrow 0^n}$	
ret $f(x)$	**ret** Y	

Fig. 2. Games utilized in the proof of Proposition 1 to show that C_2^f is a PRO.

We create a transform C_2 that is $\mathsf{PRO\text{-}Pr}$ but is *not* $\mathsf{CR\text{-}Pr}$. In other words the resulting scheme $C_2^f\colon \mathit{Dom} \to \mathit{Rng}$ is indifferentiable from a random oracle, but it is trivial to find collisions against the scheme C_2^h (even without finding collisions against h). We modify $C_1[c]$ to create $C_2[c]$ as follows. First check if $c(0^{d+n})$ is equal to 0^n and return 0^n if that is the case. Otherwise we just follow the steps specified by $C_1[c]$. Thus the scheme C_2^f returns 0^n for any message if $f(0^{d+n}) = 0^n$. Similarly the scheme C_2^h returns 0^n for any message if $h(0^{d+n}) = 0^n$. The key insight, of course, is that the differing assumptions made about the oracle impact the likelihood of this occurring. If the oracle is a random oracle, then the probability is small: we prove below that C_2^f is a pseudorandom oracle. On the other hand, we now show how to easily design a collision-resistant hash function h that causes C_2^h to not be collision resistant. Let $h'\colon \{0,1\}^{d+n} \to \{0,1\}^{n-1}$ be some collision-resistant hash function. Then $h(M)$ returns 0^n if $M = 0^{d+n}$, otherwise it returns $h'(M) \,||\, 1$. Collisions found on h would necessarily translate into collisions for h', which implies that h is collision-resistant. Furthermore since $h(0^{d+n}) = 0^n$ we have that $C_2^h(M) = 0^n$ for any message M, making it trivial to find collisions against C_2^h.

Proposition 1. [C_2 **is** $\mathsf{PRO\text{-}Pr}$] *Let $n, d > 0$ and $\mathit{Dom}, \mathit{Rng}$ be non-empty sets and $f = \mathsf{RF}_{d+n,n}$ and $\mathcal{F} = \mathsf{RF}_{\mathit{Dom},\mathit{Rng}}$ be random oracles. Let C_1^f be a pseudorandom oracle. Let C_2^f be the scheme as described above and let S be an arbitrary simulator. Then for any adversary A_2 that utilizes q_L left queries, q_R right queries, and runs in time t, there exists an adversary A_1 such that*

$$\mathbf{Adv}^{\mathrm{pro}}_{C_2,S}(A_2) \le \mathbf{Adv}^{\mathrm{pro}}_{C_1,S}(A_1) + \frac{1}{2^n}\,.$$

with A_1 utilizing the same number of queries and time as A_2.

Proof. Let $f = \mathsf{RF}_{d+n,n}$ and $\mathcal{F} = \mathsf{RF}_{\mathit{Dom},\mathit{Rng}}$ be random oracles. Let A be some pro-adversary against C_2^f. Let S be an OTM with an interface S_f that on $(d+n)$-bit inputs returns n-bit strings. We utilize a simple game-playing argument in conjunction with a hybrid argument to bound the indifferentiability of C_2 by that of C_1 (with respect to simulator S). Figure 2 displays two games, game G0 (includes boxed statement) and game G1 (boxed statement removed). The first game G0 exactly simulates the oracles C_2^f and f. The second game G1 exactly simulates the oracles C_1^f and f. We thus have that $\Pr[A^{C_2^f,f} \Rightarrow 1] = \Pr[A^{\mathrm{G0}} \Rightarrow 1]$

Prefix-free MD: $\mathrm{PRE}[c] = c^+(IV, \mathsf{padPF}(\cdot))$ where padPF: $\{0,1\}^* \to D^+$ is a prefix-free padding function	NMAC Transform: $\mathrm{NT}[c,g] = g(c^+(IV, \mathsf{pad}(\cdot)))$ where g: $\{0,1\}^n \to \{0,1\}^n$ is a function
Chop Solution: $\mathrm{CHP}[c] =$ first $n - s$ bits of $c^+(IV, \mathsf{pad}(\cdot))$	HMAC Transform: $\mathrm{HT}[c] =$ $c(c^+(IV, 0^d \mid\mid \mathsf{pad}(\cdot)) \mid\mid 0^{d-n} \mid\mid IV)$

Fig. 3. The four MD variants proposed in [1] that are PRO-Pr but not CR-Pr.

and $\Pr[A^{C_1^f, f} \Rightarrow 1] = \Pr[A^{\mathrm{G1}} \Rightarrow 1]$. Since G0 and G1 are identical-until-bad we have by the fundamental lemma of game playing [21] that $\Pr[A^{\mathrm{G0}} \Rightarrow 1] - \Pr[A^{\mathrm{G1}} \Rightarrow 1] \leq \Pr[A^{\mathrm{G1}} \text{ sets } bad]$. The right hand side is equal to 2^{-n} because f is a random oracle. Thus,

$$\begin{aligned}
\mathbf{Adv}^{\mathrm{pro}}_{C_2,S}(A_2) &= \Pr\left[A^{\mathrm{G0}} \Rightarrow 1\right] - \Pr\left[A^{\mathrm{G1}} \Rightarrow 1\right] + \\
&\qquad \Pr\left[A^{\mathrm{G1}} \Rightarrow 1\right] - \Pr\left[A^{\mathcal{F}, S^{\mathcal{F}}} \Rightarrow 1\right] \\
&\leq \Pr\left[A^{\mathrm{G1}} \text{ sets } bad\right] + \Pr\left[A^{C_1^f, f} \Rightarrow 1\right] - \Pr\left[A^{\mathcal{F}, S^{\mathcal{F}}} \Rightarrow 1\right] \\
&= \frac{1}{2^n} + \mathbf{Adv}^{\mathrm{pro}}_{C_1,S}(A_1)\,.
\end{aligned}$$

$\square$

4.2 Insecurity of Proposed PRO-Pr Transforms

Collision-resistance preservation. The result above tells us that PRO-Pr does not imply CR-Pr for arbitrary schemes. What about MD variants? One might hope that the mechanisms used to create a PRO-Pr MD variant are sufficient for rendering the variant CR-Pr also. This is not true. In fact *all* previously proposed MD variants proven to be PRO-Pr *are not* CR-Pr. The four variants are summarized in Fig. 3 and below, see [1] for more details.

The first transform is *Prefix-free MD* specified by $\mathrm{PRE}[c] = c^+(IV, \mathsf{padPF}(\cdot))$. It applies a prefix-free padding function padPF to an input message and then uses the MD iteration. The padding function padPF must output strings that are a multiple of d bits with the property that for any two strings $M \neq M'$, $\mathsf{padPF}(M)$ is not a prefix of $\mathsf{padPF}(M')$. The *Chop solution* simply drops s bits from the output of the MD iteration applied to a message. The *NMAC transform* applies a second, distinct compression function to the output of an MD iteration; it is defined by $\mathrm{NT}[c,g] = g(c^+(IV, \mathsf{pad}(\cdot)))$, where g is a function from n bits to n bits (distinct from h). Lastly, the *HMAC Transform* is defined by $\mathrm{HT}[c] = c(c^+(IV, 0^d \mid\mid \mathsf{pad}(\cdot)) \mid\mid 0^{d-n} \mid\mid IV)$. This transform similarly utilizes enveloping: the MD iteration is fed into c in a way that distinguishes this last call from the uses of c inside the MD iteration. The prepending of a d-bit string of zeros to an input message helps ensure that the envelope acts differently than the first compression function application.

Let $IV = 0^n$. We shall use the collision-resistant hash function h that maps 0^{d+n} to 0^n (defined in Sect. 4.1). We first show that the PRE construction, while being $\mathsf{PRO\text{-}Pr}$ for all prefix-free encodings, is not $\mathsf{CR\text{-}Pr}$ for all prefix-free encodings. Let $\mathsf{padPF}(M) = g_2(M)$ from Sect. 3.3 of [1]. Briefly, $g_2(M) = 0 \mid\mid M_1, \ldots, 0 \mid\mid M_{k-1}, 1 \mid\mid M_k$ for $M_1 \mid\mid \cdots \mid\mid M_k \overset{d-1}{\leftarrow} M \mid\mid 10^r$, where $r = (d-1) - ((|M|+1) \bmod d-1)$. (That is we append a one to M, and then enough zero's to make a string with length a multiple of $d-1$.) Now let $X = 0^{d-1}$ and $Y = 0^{2(d-1)}$. Then we have that $\mathrm{PRE}^h(X) = \mathrm{PRE}^h(Y)$ and no collisions against h occur. We should note that *some* prefix-free encodings will render PRE $\mathsf{CR\text{-}Pr}$, for example any that also include strengthening. The important point here is that strengthening does not ensure prefix-freeness and vice-versa.

For the other three constructions, we assume that $\mathsf{pad}(M)$ simply appends a one and then enough zeros to make a string with length a multiple of d. Let $X = 0^d$ and $Y = 0^{2d}$. Then we have that $\mathrm{CHP}^h(X) = \mathrm{CHP}^h(Y)$ and $\mathrm{NT}^h(X) = \mathrm{NT}^h(Y)$ and $\mathrm{HT}^h(X) = \mathrm{HT}^h(Y)$. Never is there a collision generated against h.

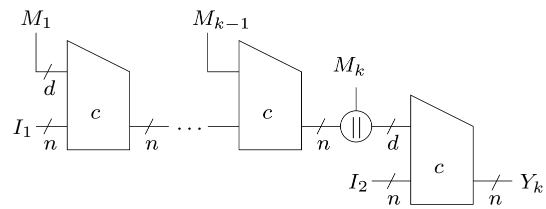
The straightforward counter-examples exploit the weakness of the basic MD transform. As noted previously, the MD transform does not give any guarantees about collision resistance, and only when we consider particular padding functions (i.e., $\mathsf{pad64}$) can we create a $\mathsf{CR\text{-}Pr}$ transform. Likewise, we have illustrated that the mechanisms of prefix-free encodings, dropping output bits, and enveloping do nothing to help ensure collision-resistance is preserved, even though they render the transforms $\mathsf{PRO\text{-}Pr}$. To properly ensure preservation of both properties, we must specify transforms that make use of mechanisms that ensure collision-resistance preservation *and* mechanisms that ensure pseudorandom oracle preservation. In fact, it is likely that adding strengthening to these transforms would render them $\mathsf{CR\text{-}Pr}$. However, as we show in the next section, our new construction (with strengthening) is already more efficient than these constructions (without strengthening).

PRF PRESERVATION. It is not formally meaningful to compare PRF preservation with PRO preservation, since the resulting schemes in either case are different types of objects (one keyed and one keyless). However we can look at particular transforms. Of the four proposed by Coron et al. only PRE is known to be $\mathsf{PRF\text{-}Pr}$. Let e be a PRF. Since we are using the key-via-IV strategy, the keyed version of PRE^e is $\mathrm{PRE}^e_K(M) = e^+(K, \mathsf{padPF}(M))$. This is already known to be a good PRF [13]. As for the other three transforms, it is unclear whether any of them are $\mathsf{PRF\text{-}Pr}$. For NT, we note that the security will depend greatly on the assumptions made about g. If g is a separately keyed PRF, then we can apply the proof of NMAC [4]. On the other hand, if g is not one-way, then an adversary can determine the values produced by the underlying MD iteration and mount simple length-extension attacks. Instead of analyzing these transforms further (which are not $\mathsf{CR\text{-}Pr}$ anyway), we look at a new construction.

5 The EMD Transform

We propose a transform that is $\mathsf{CR\text{-}Pr}$, $\mathsf{PRO\text{-}Pr}$, and $\mathsf{PRF\text{-}Pr}$. Let n, d be numbers such that $d \geq n + 64$. Let $c\colon \{0,1\}^{d+n} \to \{0,1\}^n$ be a function and let $D^\circ = \cup_{i \geq 1} \{0,1\}^{(i+1)d-n}$. Then we define the enveloped Merkle-Damgård iteration $c^\circ\colon \{0,1\}^{2n} \times D^\circ \to \{0,1\}^n$ on c by the algorithm given below.

Algorithm $c^\circ(I_1, I_2, M)$:
$M_1 \cdots M_k \xleftarrow{d} M$
$P \leftarrow M_1 \cdots M_{k-1}$
ret $c(c^+(I_1, P) \mid\mid M_k \mid\mid I_2)$



To specify our transform we require a padding function $\mathsf{padEMD}\colon \{0,1\}^{\leq 2^{64}} \to D^\circ$ for which the last 64 bits of $\mathsf{padEMD}(M)$ encodes $|M|$. Fix $IV1, IV2 \in \{0,1\}^n$ with $IV1 \neq IV2$. Then we specify the *enveloped Merkle-Damgård transform* $\mathrm{EMD}[c] = c^\circ(IV1, IV2, \mathsf{padEMD}(\cdot))$.

EMD utilizes two main mechanisms for ensuring property preservation. The first is the well-known technique of strengthening: we require a padding function that returns a string appended with the 64-bit encoding of the length. This ensures that EMD preserves collision-resistance. The second technique consists of using an 'extra' compression function application to envelope the internal MD iteration. It is like the enveloping mechanism used by Maurer and Sjoden in a different setting [18] (which is discussed in more detail in the full version of the paper [19]), but distinct from prior enveloping techniques used in the current setting. First, it includes message bits in the envelope's input (in NMAC/HMAC and HT, these bits would be a fixed constant, out of adversarial control). This results in a performance improvement since in practice it is always desirable to have d as large as possible relative to n (e.g., in SHA-1 $d = 512$ and $n = 160$). Second, it utilizes a distinct initialization vector to provide (with high probability) domain separation between the envelope and internal applications of the compression function. This mechanism allows us to avoid having to use other previously proposed domain separation techniques while still yielding a $\mathsf{PRO\text{-}Pr}$ transform. (The previous techniques were prefix-free encodings or the prepending of 0^d to messages, as used in the HT transform; both are more costly.)

5.1 Security of EMD

Collision-resistance preservation. Let $h\colon \{0,1\}^{d+n} \to \{0,1\}^n$ be a collision resistant hash function. Then any adversary which finds collisions against EMD^h (two messages $M \neq M'$ for which $\mathrm{EMD}^h(M) = \mathrm{EMD}^h(M')$) will necessarily find collisions against h. This can be proven using a slightly modified version of the proof that SMD is collision-resistant [2, 3], and we therefore omit the details. The important intuition here is that embedding the length of messages in the last block is crucial; without the strengthening the scheme would *not* be collision resistant (similar attacks as those given in Section 4 would be possible).

PRO PRESERVATION. Now we show that EMD is PRO-Pr. We first prove a slightly different transform is PRO-Pr and then show that EMD reduces to this other transform. Let $f, g = \mathsf{RF}_{d+n,n}$ be random oracles. For any strings $P_1 \in D^+$ and $P_2 \in \{0,1\}^{d-n}$ we define the function $gf^+ \colon D^\circ \to \{0,1\}^n$ by $gf^+(P \mid\mid S) = g(f^+(IV1, P_1) \mid\mid P_2 \mid\mid IV2)$. This function is essentially EMD^f, except that we replace the envelope with an independent random oracle g. The following lemma states that gf^+ is a pseudorandom oracle.

Lemma 1. [gf^+ **is a PRO**] *Let $f, g = \mathsf{RF}_{d+n,n}$. Let A be an adversary that asks at most q_L left queries, q_f right f-queries, q_g right g-queries and runs in time t. Then*

$$\mathbf{Adv}^{\mathrm{pro}}_{gf^+,\,SB}(A) \leq \frac{(q_L + q_g)^2 + q_f^2 + q_g q_f}{2^n}$$

where $SB = (SB_f, SB_g)$ is defined in Fig. 4 and $q_{SB} \leq q_g$ and $t_{SB} = \mathcal{O}(q_f^2 + q_g q_f)$.

We might hope that this result is given by Theorem 4 from [1], which states that $\mathrm{NT}^{f,g}$ is indifferentiable from a random oracle. Unfortunately, their theorem statement does not allow for adversarially-specified bits included in the input to g. Thus we give a full proof of Lemma 1, found in the full version of the paper [19]. The next theorem captures the main result, and its proof is also in the full version. For completeness, we provide the simulators $SB = (SB_f, SB_g)$ and SA in Fig. 4.

Theorem 1. [**EMD is PRO-Pr**] *Fix n, d, and let $IV1, IV2 \in \{0,1\}^n$ with $IV1 \neq IV2$. Let $f = \mathsf{RF}_{d+n,n}$ and $\mathcal{F} = \mathsf{RF}_{*,n}$ be random oracles. Let A be an adversary that asks at most q_L left queries (each of length no larger than ld bits), q_1 right queries with lowest n bits not equal to $IV2$, q_2 right queries with lowest n bits equal to $IV2$, and runs in time t. Then*

$$\mathbf{Adv}^{\mathrm{pro}}_{\mathrm{EMD},\,SA}(A) \leq \frac{(q_L + q_2)^2 + q_1^2 + q_2 q_1}{2^n} + \frac{l q_L^2}{2^n}\,.$$

where the simulator SA is defined in Fig. 4 and $q_{SA} \leq q_2$ and $t_{SA} = \mathcal{O}(q_1^2 + q_2 q_1)$.

PRF PRESERVATION. We utilize the key-via-IV strategy to create a keyed version of our transform, which is $\mathrm{EMD}^e_{K_1,K_2}(M) = e^\circ(K_1, K_2, M)$ (for some PRF e). The resulting scheme is very similar to NMAC, which we know to be PRF-Pr [5]. Because our transform allows direct adversarial control over a portion of the input to the envelope function, we can not directly utilize the proof of NMAC (which assumes instead that these bits are fixed constants). However, the majority of the proof of NMAC is captured by two lemmas, The first (Lemma 3.1 [5]) shows (informally) that the keyed MD iteration is unlikely to have outputs that collide. The second lemma (Lemma 3.2 [5]) shows that composing the keyed MD iteration with a separately keyed PRF yields a PRF. We omit the details.

<u>On query $SB_f(X)$:</u>

$Y \xleftarrow{\$} \{0,1\}^n$
Parse X into $U \,||\, V$ s.t.
  $|U| = d$, $|V| = n$
if $V = IV1$ **then** $\textsc{NewNode}(U) \leftarrow Y$
if $M_1 \cdots M_i \leftarrow \textsc{GetNode}(V)$ **then**
  $\textsc{NewNode}(M_1 \cdots M_i U) \leftarrow Y$
ret Y

<u>On query $SB_g(X)$:</u>

Parse X into $V \,||\, U \,||\, W$ s.t.
  $|V| = n$, $|U| = d - n$, $|W| = n$
if $W = IV2$ **and**
  $M_1 \cdots M_i \leftarrow \textsc{GetNode}(V)$ **then**
  ret $\mathcal{F}(M_1 \cdots M_i U)$
ret $Y \xleftarrow{\$} \{0,1\}^n$

<u>On query $SA(X)$:</u>

Parse X into $V \,||\, U \,||\, W$ s.t.
  $|V| = n$, $|U| = d - n$, $|W| = n$
if $W = IV2$ **then**
  if $M_1 \cdots M_i \leftarrow \textsc{GetNode}(V)$ **then**
    ret $\mathcal{F}(M_1 \cdots M_i U)$
  else ret Y
Parse X into $U \,||\, V$ s.t.
  $|U| = d$, $|V| = n$
if $V = IV1$ **then** $\textsc{NewNode}(U) \leftarrow Y$
if $M_1 \cdots M_i \leftarrow \textsc{GetNode}(V)$ **then**
  $\textsc{NewNode}(M_1 \cdots M_i U) \leftarrow Y$
ret Y

Fig. 4. Pseudocode for simulators SB (Lemma 1) and SA (Theorem 1).

Theorem 2. [EMD is PRF-Pr] *Fix n, d and let e: $\{0,1\}^{d+n} \rightarrow \{0,1\}^n$ be a function family keyed via the low n bits of its input. Let A be a prf-adversary against keyed* EMD *using q queries of length at most m blocks and running in time t. Then there exists prf-adversaries A_1 and A_2 against e such that*

$$\mathbf{Adv}^{\mathrm{prf}}_{\mathrm{EMD}^e_{K_1,K_2}}(A) \leq \mathbf{Adv}^{\mathrm{prf}}_e(A_1) + \binom{q}{2}\left[2m \cdot \mathbf{Adv}^{\mathrm{prf}}_e(A_2) + \frac{1}{2^n}\right]$$

where A_1 utilizes q queries and runs in time at most t and A_2 utilizes at most two oracle queries and runs in time $\mathcal{O}(mT_e)$ where T_e is the time for one computation of e.

Acknowledgments

The authors are supported in part by NSF grant CNS 0524765 and a gift from Intel Corporation.

References

1. Coron, J.S., Dodis, Y., Malinaud, C., Puniya, P.: Merkle-Damgard Revisited: How to Construct a Hash Function. In: Advances in Cryptology - CRYPTO '05. Volume 3621 of Lecture Notes in Computer Science, Springer (2004) 21–39.
2. Merkle, R.C.: One way hash functions and DES. In: Advances in Cryptology - CRYPTO '89. Volume 435 of Lecture Notes in Computer Science, Springer (1989) 428–446.
3. Damgård, I.: A design principle for hash functions. In: Advances in Cryptology - CRYPTO '89. Volume 435 of Lecture Notes in Computer Science, Springer (1989) 416–427.

4. Bellare, M., Canetti, R., Krawczyk, H.: Keying Hash Functions for Message Authentication. In: Advances in Cryptology - CRYPTO '96. Volume 1109 of Lecture Notes in Computer Science, Springer (1996) 1–15.
5. Bellare, M.: New Proofs for NMAC and HMAC: Security Without Collision-Resistance. In: Advances in Cryptology - CRYPTO '06. Volume 4117 of Lecture Notes in Computer Science, Springer (2006) 602–619.
6. Bellare, M., Rogaway, P.: Random oracles are practical: a paradigm for designing efficient protocols. In: CCS '93, ACM Press (1993) 62–73.
7. Bellare, M., Rogaway, P.: Optimal Asymmetric Encryption. In: Advances in Cryptology - EUROCRYPT '94. Volume 950 of Lecture Notes in Computer Science, Springer (1994) 92–111.
8. Bellare, M., Rogaway, P.: The Exact Security of Digital Signatures - How to Sign with RSA and Rabin. In: Advances in Cryptology - EUROCRYPT '96. Volume 1070 of Lecture Notes in Computer Science, Springer (1996) 399–416.
9. RSA Laboratories: RSA PKCS #1 v2.1: RSA Cryptography Standards (2002).
10. Maurer, U.M., Renner, R., Holenstein, C.: Indifferentiability, Impossibility Results on Reductions, and Applications to the Random Oracle Methodology. In: TCC '04. Volume 2951 of Lecture Notes in Computer Science, Springer (2004) 21–39.
11. Canetti, R., Goldreich, O., Halevi, S.: The random oracle methodology, revisited. J. ACM **51**(4) (2004) 557–594.
12. Bellare, M., Boldyreva, A., Palacio, A.: An Uninstantiable Random-Oracle-Model Scheme for a Hybrid-Encryption Problem. In Cachin, C., Camenisch, J., eds.: Advances in Cryptology - EUROCRYPT '04. Volume 3027 of Lecture Notes in Computer Science, Springer (2004) 171–188.
13. Bellare, M., Canetti, R., Krawczyk, H.: Pseudorandom functions revisited: the cascade construction and its concrete security. In: FOCS '96: Proceedings of the 37th Annual Symposium on Foundations of Computer Science, IEEE Computer Society (1996) 514–523.
14. Wang, X., Yin, Y.L., Yu, H.: Finding Collisions in the Full SHA-1. In: Advances in Cryptology - CRYPTO '05. Volume 3621 of Lecture Notes in Computer Science, Springer (2005) 17–36.
15. Wang, X., Yu, H.: How to Break MD5 and Other Hash Functions. In: Advances in Cryptology - EUROCRYPT '05. Volume 3494 of Lecture Notes in Computer Science, Springer (2005) 19–35.
16. An, J.H., Bellare, M.: Constructing VIL-MACs from FIL-MACs: message authentication under weakened assumptions. In: Advances in Cryptology - CRYPTO '99. Volume 1666 of Lecture Notes in Computer Science, Springer (1999) 252–269.
17. Bellare, M., Rogaway, P.: Collision-Resistant Hashing: Towards Making UOWHFs Practical. In: Advances in Cryptology - CRYPTO '97. Volume 1294 of Lecture Notes in Computer Science, Springer (1997) 470–484.
18. Maurer, U., Sjödin, J.: Single-key AIL-MACs from any FIL-MAC. In: ICALP '05. Volume 3580 of Lecture Notes in Computer Science, Springer (2005) 472–484.
19. Bellare, M., Ristenpart, T.: Multi-property-preserving Hash Domain Extension and the EMD Transform (full version of this paper) (2006) `http://www.cse.ucsd.edu/users/mihir`.
20. National Institute of Standards and Technology: FIPS PUB 180-1: Secure Hash Standard. (1995) Supersedes FIPS PUB 180 1993 May 11.
21. Bellare, M., Rogaway, P.: The Security of Triple Encryption and a Framework for Code-Based Game-Playing Proofs. In: Advances in Cryptology - EUROCRYPT '06. Volume 4004 of Lecture Notes in Computer Science, Springer (2006) 409–426.