Salvaging Merkle-Damgård for Practical Applications

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

[1] Dept. of Computer Science, New York University.
dodis@cs.nyu.edu, http://www.cs.nyu.edu/~dodis/

[2] Dept. of Computer Science & Engineering, University of California San Diego
tristenp@cs.ucsd.edu, http://www-cse.ucsd.edu/users/tristenp

[3] Dept. of Computer Science, Portland State University
Faculty of Informatics, University of Lugano
teshrim@cs.pdx.edu, http://www.cs.pdx.edu/~teshrim

Abstract. Many cryptographic applications of hash functions are analyzed in the random oracle model. Unfortunately, most concrete hash functions, including the SHA family, use the iterative (strengthened) Merkle-Damgård transform applied to a corresponding compression function. Moreover, it is well known that the resulting "structured" hash function cannot be generically used as a random oracle, even if the compression function is assumed to be ideal. This leaves a large disconnect between theory and practice: although no attack is known for many concrete applications utilizing existing (Merkle-Damgård based) hash functions, there is no security guarantee either, even by idealizing the compression function.

Motivated by this question, we initiate a rigorous and modular study of developing new notions of (still idealized) hash functions which would be (a) natural and elegant; (b) sufficient for arguing security of important applications; and (c) provably met by the (strengthened) Merkle-Damgård transform, applied to a "strong enough" compression function. In particular, we develop two such notions satisfying (a)-(c): a *preimage aware* function ensures that the attacker cannot produce a "useful" output of the function without already "knowing" the corresponding preimage, and a *public-use random oracle*, which is a random oracle that reveals to attackers messages queried by honest parties.

1 Introduction

The primary security goal for cryptographic hash functions has historically been collision-resistance. Consequently, in-use hash functions, such as the SHA family of functions [28], were designed using the (strengthened[4]) Merkle-Damgård

[4] We do not mean to imply that there is a *weak* MD transform, but this name seems to be in common use.

(MD) transform [27, 17]: the input message M is suffix-free encoded (e.g. by appending a message block containing the length of M) and then digested by the cascade construction using an underlying fixed-input-length (FIL) compression function. The key security feature of the strengthened MD transformation is that it is *collision-resistance preserving* [17, 27]. Namely, as long as the FIL compression function is collision-resistant, the resulting variable-input-length (VIL) hash function will be collision-resistant too.

Random oracle model. Unfortunately, the community has come to understand that collision-resistance alone is insufficient to argue the security of many important applications of hash functions. Moreover, many of these applications, like Fiat-Shamir [22] signatures or RSA [4] encryption, are such that no standard model security assumption about the hash function appears to suffice for proving security. On the other hand, no realistic attacks against these applications have been found. Motivated in part by these considerations, Bellare and Rogaway [4] introduced the Random Oracle (RO) model, which models the hash function as a public oracle implementing a random function. Using this abstraction, Bellare and Rogaway [4–6] and literally thousands of subsequent works managed to formally argue the security of important schemes. Despite the fact that a proof in the RO model does *not* always guarantee security when one uses a real (standard model) hash function [13], such a proof does provide evidence that the scheme is structurally sound.

Is Merkle-Damgård a good design? Given the ubiquity of MD-based hash functions in practice, and the success of the RO model in provable security, it is natural to wonder if a MD-based hash function H is reasonably modeled as a RO, at least when the compression function is assumed to be ideal. But even without formalizing this question, one can see that the answer is negative. For example, the well-known *extension attack* allows one to take a value $H(x)$ for *unknown* x, and then compute the value $H(x, \langle \ell \rangle, y)$, where ℓ is the length of x and y is an arbitrary suffix. Clearly, this should be impossible for a truly random function. In fact, this discrepancy leads to simple attacks for *natural* schemes proven secure in the random oracle model (see [16]).

Consequently, Coron et al. [16] adapted the indifferentiability framework of Maurer et al. [26] to define formally what it means to build a secure VIL-RO from smaller (FIL) idealized components (such as an ideal compression function or ideal cipher). Not surprisingly, they showed that the strengthened MD transform does not meet this notion of security, even when applied to an ideal compression function. Although [16] (and several subsequent works [2, 3, 25]) presented straightforward fixes to the MD paradigm that yield hash functions indifferentiable from a VIL-RO, we are still faced with a large disconnect between theory and practice. Namely, many applications only enjoy proofs of security when the hash function is modeled as a "monolithic" VIL-RO, while in practice these applications use existing MD-based hash functions which (as we just argued) are demonstrably differentiable from a monolithic RO (even when compression functions are ideal). Yet despite this gap, *no* practical attacks on the MD-based design (like the extension attack) seem to apply for these important applications.

"Salvaging" Merkle-Damgård. The situation leads us to a question not addressed prior to this work: *given a current scheme that employs an* MD*-based hash function H and yet does not seem vulnerable to extension-type attacks, can we prove its security (at least if the compression function f is assumed to be ideal)*? The most direct way to answer this question would be to re-prove, from scratch, the security of a given application when an MD-based hash function is used. Instead, we take a more modular approach consisting of the following steps:

(1) Identify a natural (idealized) property X that is satisfied by a random oracle.
(2) Argue that X *suffices* for proving the security of a given (class of) application(s), originally proved secure when H is modeled as a monolithic RO.
(3) Argue that the *strengthened MD-transform is property-preserving for* X; that is, as long as the compression function f satisfies X, then the VIL hash H satisfies X.
(4) Conclude that, as long as the compression function f satisfies X, the given (class of) application(s) is secure with an MD-based hash function H.

Although this approach might not be applicable to all scenarios, when it *is* applicable it has several obvious advantages over direct proofs. First, it supports proofs that are easier to derive, understand, and verify. Second, proving that a hash function satisfying X alone is enough (as opposed to being like a "full-blown" RO) for a given application elucidates more precisely which (idealized) property of the hash function is essential for security. Third, if the property X is natural, it is interesting to study in its own right. Indeed, we will show several applications of our notions which are quite general and not necessarily motivated by salvaging the MD transform. Finally, due to point (4), it suffices to argue/assume "only" that the compression function f — a smaller and much-better-studied object — satisfies property X.

So which properties X? We introduce two: *preimage awareness* and indifferentiability from a *public-use random oracle*.

1.1 Preimage Aware Functions

Intuitively, a function being Preimage Aware (PrA) means that if an attacker is able to find a "later useful" output y of the hash function H, then it must "already know" a corresponding preimage x. A bit more precisely, assume we build H using some ideal primitive P (which could be a compression function or a block cipher). Then, if the attacker A produces a value y at some point in time, either one can immediately "extract" the corresponding (unique) preimage x of y from the transcript of calls that A made to P so far, or, if one fails to do so, A is exceedingly unlikely to find a valid preimage of y even with the benefit of additional calls to P. Our notion is very similar in spirit to the notion of plaintext awareness for encryption schemes [4, 1] and the notion of extractability for perfectly one-way functions [11, 12]; we discuss these further below.

We notice that random oracles are clearly PrA. In fact, preimage awareness precisely captures the spirit behind a common proof technique used in the RO

model, often referred to as *extractability*, making it an interesting notion to consider. We also show that preimage awareness is a natural strengthening of collision-resistance (CR). That preimage awareness lies between being a RO and CR turns out to be quite useful: informally, a PrA function is "strong enough" to be a good replacement for a RO in some applications (where CR is insufficient), and yet the notion of preimage awareness is "weak enough" to be preserved by strengthened MD (like CR).

MERKLE-DAMGÅRD PRESERVES PREIMAGE AWARENESS. We show that the *(strengthened) MD transform preserves preimage awareness*, in stark contrast to the fact that it does *not* preserve indifferentiability from a RO [16]. Thus, to design a variable-input-length preimage aware (VIL-PrA) function, it is sufficient to construct a FIL-PrA function, or, even better, argue that existing compression functions are PrA, *even when they are not necessarily (indifferentiable from) random oracles*. The proof of this is somewhat similar to (but more involved than) the corresponding proof that MD preserves collision-resistance.

APPLICATION: DOMAIN EXTENSION FOR ROS. A PrA hash function is *exactly* what is needed to argue secure domain extension of a random oracle. More precisely, assuming h is a FIL-RO, and H is a VIL-PrA hash function (whose output length matches that of the input of f), then $F(x) = h(H(x))$ is indifferentiable from a VIL-RO. Ironically, when H is just CR, the above construction of F was used by [16] to argue that CR functions are not sufficient for domain extension of a RO. Thus, the notion of PrA can be viewed simultaneously as a non-trivial strengthening of CR, which makes such domain extension work, while also a non-trivial weakening of RO, which makes it more readily achieved.

RECIPE FOR HASH DESIGN. The previous two properties of PrA functions give a general recipe for how to construct hash functions suitable for modeling as a VIL-RO. First, invest as must as needed to construct a strong FIL function h (i.e. one suitable for modeling as a FIL-RO.) Even if h is not particularly efficient, this is perhaps acceptable because it will only be called once per message (on a short input). Second, specify an efficient construction of a VIL-PrA hash function built from some cryptographic primitive P. But for this we use the fact that MD is PrA-preserving; hence, it is sufficient to focus on constructing a FIL-PrA compression function f from P, and this latter task could be much easier than building from P an object suitably like a FIL-RO.

Adopting our more modular point-of-view, several existing hash constructions in the literature [16, 2, 3, 30, 19] enjoy an easier analysis. For example, the NMAC construction of [16] becomes an example of our approach, where the outer h and the inner f are both implemented to be like (independent) FIL-ROs. In [16] it is argued directly, via a difficult and long argument, that the inner f can be replaced by the Davies-Meyer construction (in the ideal-cipher model), despite the fact that Davies-Meyer is *not* itself indifferentiable from a FIL-RO. We can instead just prove that Davies-Meyer is PrA (which requires only a few lines due to the existing proof of CR [7]) and then conclude.

LIFTING FROM CR TO PRA. Another important aspect of preimage awareness is that, for many important constructions, it gives a much more satisfactory

security target than collision resistance. Indeed, there exists a large body of work [29, 7, 23, 24, 32, 33, 31, 19] building FIL-CR hash functions from idealized block ciphers and permutations. On the one hand, it seems very hard to prove the security of such schemes in the standard model, since there exists a black-box separation [34] between collision-resistant hash functions and standard-model block ciphers (which are equivalent to one-way functions). On the other hand, it seems quite unsatisfactory that one starts with such a "powerful" idealized primitive (say, an ideal cipher), only to end up with a much "weaker" standard model guarantee of collision resistance (which is also insufficient for many applications of hash functions). The notion of preimage awareness provides a useful solution to this predicament. We show that *all* the constructions proven CR in [29, 7, 33, 31, 19] are provably PrA. This is interesting in its own right, but also because one can now use these practical constructions within our aforementioned recipe for hash design. We believe (but offer no proof) that most other CR ideal-primitive-based functions, e.g. [23, 24, 32], are also PrA.

Other applications/connections? We believe that PrA functions have many more applications than the ones so far mentioned. As one example, PrA functions seem potentially useful for achieving straight-line extractability for various primitives, such as commitments or zero-knowledge proofs. These, in turn, could be useful in other contexts. As already mentioned, preimage awareness seems to be quite related to the notion of *plaintext awareness* in public-key encryption schemes [5, 1], and it would be interesting to formalize this potential connection. PrA functions are also very related to so called *extractable hash functions* (EXT) recently introduced by Canetti and Dakdouk [11, 12]. However, there are some important differences between EXT and PrA, which appear to make our respective results inapplicable to each other: (a) EXT functions are defined in the standard model, while PrA functions in an idealized model; (b) EXT functions are keyed (making them quite different from in-use hash functions), while PrA functions can be keyed or unkeyed; (c) EXT functions do not permit the attacker to sample *any* "unextractable" image y, while PrA functions only exclude images y which could be later "useful" to the attacker; (d) EXT functions allow the extractor to depend on the attacker, while PrA functions insist on a universal extractor.

1.2 Public-Use Random Oracles

Next, we consider applications that never evaluate a hash function on secret data (i.e. data that must be hidden from adversaries). This means that whenever the hash function is evaluated on some input x by an honest party C, it is safe to immediately give x to the attacker A. We model this by formalizing the notion of a *public-use random oracle* (pub-RO); such a RO can be queried by adversaries to reveal all so-far-queried messages. This model was independently considered, under a different motivation, by Yoneyama et al. [36] using the name leaky random oracle. Both of our papers observe that this weakening of the RO model is actually enough to argue security of many (but, certainly, not all) classical schemes analyzed in the random oracle model. In particular, a vast

majority of digital signature schemes, including Full Domain Hash (FDH) [4], probabilistic FDH [15], Fiat-Shamir [22], BLS [10], PSS [6] and many others, are easily seen secure in the pub-RO model. For example, in the FDH signature scheme [4], the RO H is only applied to the message m supplied by the attacker, to ensure that the attacker cannot invert the value $H(m)$ (despite choosing m). Other applications secure in the pub-RO model include several identity-based encryption schemes [9, 8], where the random oracle is only used to hash the user identity, which is public.

We go on to formalize this weakening of ROs in the indifferentiability framework of Maurer et al. [26]. This allows us to define what it means for a hash function H (utilizing some ideal primitive P) to be indifferentiable from a *public-use* random oracle (pub-RO). As our main technical result here, we argue that the MD transform preserves indifferentiability from a pub-RO, even though it does not preserve general indifferentiability from a (regular) RO. To get some intuition about this fact, it is instructive to examine the extension attack mentioned earlier, which was the root of the problem with MD for general indifferentiability. There one worried about adversaries being able to infer the hash output on a message with unknown prefix. In the public-use setting, this is not an issue at all: the security of a public-use application could never be compromised by extension attacks since all messages are known by the attacker.

As a corollary of this result (and the composition theorem of [26]), we'll see that if the compression function f is indifferentiable from a FIL pub-RO, we can immediately give a security proof for the above-mentioned public-use applications. In particular, this is true when f is modeled as an ordinary FIL-RO.

2 Preliminaries

When S is a set, $x \xleftarrow{\$} S$ means to sample uniformly from S and assign the value to x. When Dom and Rng are non-empty sets, let $\mathsf{RF}_{\mathit{Dom},\mathit{Rng}}$ be the algorithm that implements a lazily-sampled random oracle mapping from Dom to Rng. We shorten this to $\mathsf{RF}_{\mathit{Dom},n}$ or $\mathsf{RF}_{N,n}$ when the range (resp.) domain and range are bit strings of fixed lengths $N, n > 0$; $\mathsf{RF}_{*,\tau}$ is a random oracle with constant output stretch τ. Let $\kappa, n > 0$ be integers. A block cipher is a map $E\colon \{0,1\}^{\kappa} \times \{0,1\}^{n} \to \{0,1\}^{n}$ such that $E(k,\cdot)$ is a permutation for all $k \in \{0,1\}^{\kappa}$. Let $\mathsf{BC}(\kappa, n)$ be the set of all such block ciphers.

For any algorithm f that accepts inputs from $\mathit{Dom} \subseteq \{0,1\}^*$, we write $\mathsf{Time}(f, m)$ to mean the maximum time to run $f(x)$ for any input $x \in \{0,1\}^m \subseteq \mathit{Dom}$. When f is a function with domain $\mathit{Dom} \subseteq \{0,1\}^*$, we define $\mathsf{Time}(f, m)$ to be the minimum, over all programs T_f that implement the mapping f, of the size of T_f plus the worst case running time of T_f over all elements $x \in \{0,1\}^m \subseteq \mathit{Dom}$. In either case, when we suppress the second argument, writing just $\mathsf{Time}(f)$, we mean to maximize over all strings in the domain. Running times are relative to some fixed underlying RAM model of computation, which we do not specify here.

As a small abuse of standard notation, we write $\mathcal{O}(X)$ to hide fixed, absolute constants that are much smaller than the argument X.

INTERACTIVE TMs. An Interactive Turing Machine (ITM) accepts inputs via an input tape, performs some local computation using internal state that persists across invocations, and replies via an output tape. An ITM might implement various distinct functionalities $f_1, f_2, \ldots$ that are to be exposed to callers. We write $P = (f_1, f_2, \ldots)$ for an ITM implementing $f_1, f_2, \cdots$. When functionalities f_i, f_j (say) do not share state, we say that f_i and f_j are *independent* functionalities; these will be explicitly noted. We write M^P if an ITM M has access to all interfaces of P and write M^{f_i} if M has access only to a particular interface f_i of P. We sometimes use the moniker *ideal primitive* to refer to an ITM; this is to emphasize the use of an ITM as building block for some larger functionality. We write $\mathcal{F} = \mathsf{RF}_{Dom,Rng}$ to signify that $\mathcal{F}$ is the ideal primitive that implements the algorithm $\mathsf{RF}_{Dom,Rng}$.

HASH FUNCTIONS AND MERKLE-DAMGÅRD. Let $Dom \subseteq \{0,1\}^*$ be a non-empty set of strings, and Rng be a non-empty set (typically $\{0,1\}^n$ for some integer $n > 0$). A *hash function* is then a map $H\colon Dom \to Rng$. We will be concerned with hash functions that use (oracle access to) an underlying ideal primitive P. We write H^P when we want to make this dependency explicit. When the primitive is clear from context, we will sometimes suppress reference to it. When computing $\mathsf{Time}(H, \cdot)$, calls to P are unit cost. Similar to our definition of $\mathsf{Time}(H, m)$, we write $\mathsf{NumQueries}(H, m)$ for the minimum, over all programs T_H that compute H, of the maximum number of queries to P required to compute $H^P(x)$ for any $x \in \{0,1\}^m \subseteq Dom$.

The primary method by which hash functions are constructed from underlying primitives is the strengthened Merkle-Damgård transform (SMD). For integers $n, d > 0$, let $f^P\colon \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^n$ be a *compression function* (using idealized primitive P). Let $y_0 = IV$, a fixed constant. We write $\mathrm{SMD}[f^P]$ for the algorithm that works on input $M \in \{0,1\}^*$ by first computing $m_1 \cdots m_\ell \leftarrow \mathsf{sfpad}(M)$, running $y_i \leftarrow f^P(y_{i-1}, m_i)$ for each $i \in [1\,..\,\ell]$ and returning y_ℓ. Here $\mathsf{sfpad}(\cdot)$ is a function that returns a *suffix-free encoding* of M that is parsed into ℓ blocks of d-bits each, where $\ell \geq \lceil |M|/d \rceil$ is defined by the encoding. A suffix-free encoding has the property that for any M, M' such that $|M| < |M'|$ the string returned by $\mathsf{sfpad}(M)$ is not a suffix of $\mathsf{sfpad}(M')$. (For example, appending to a message its length.) Similarly, we write $\mathrm{MD}[f^P]$ for the algorithm that splits input $M \in (\{0,1\}^d)^+$ into blocks $m_1, \cdots, m_\ell$, each of size d bits, then runs $y_i \leftarrow f^P(y_{i-1}, m_i)$ for each $i \in [1\,..\,\ell]$, and returns y_ℓ. When the underlying compression function is itself an idealized primitive provided as an oracle, the algorithms SMD and MD work in the obvious manner.

PSEUDORANDOM ORACLES. Following [16], we utilize the indifferentiability framework [26] to formalize the notion of "behaving like a random oracle", which we call being a pseudorandom oracle (PRO) following [2, 3]. Fix non-empty sets Dom, Rng. Let P be an ideal primitive and let $\mathcal{F} = \mathsf{RF}_{Dom,Rng}$. A simulator $\mathcal{S}$ is an ITM that matches the number of interfaces of P and has oracle access to $\mathcal{F}$.

A PRO adversary A has access to oracles and outputs a bit. We define the pro advantage of a PRO adversary A against a function H^P mapping from Dom to Rng by

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

where the first probability is over the coins used by A and primitive P, and the second is over the coins used by A, $\mathcal{F}$, and $\mathcal{S}$. Note that a crucial aspect of the definition is that the simulator, while able to query $\mathcal{F}$ itself, does *not* get to see the queries made by the adversary to $\mathcal{F}$.

3 Preimage Awareness

Suppose H is a hash function built from an (ideal) primitive P. We seek to, roughly speaking, capture a notion which states that an adversary who knows a "later useful" output z of H^P must "already know" (be aware of) a particular corresponding preimage x. We can capture the spirit of this notion using a deterministic algorithm called an *extractor*. Consider the following experiment. An adversary A initially outputs a range point z. The extractor is run on two inputs: z and an *advice string* α. The latter contains a description of all of A's queries so far to P and the corresponding responses. The extractor outputs a value x in the domain of H. Then A runs again and attempts to output a preimage x' such that $H^P(x) = z$ but $x \neq x'$. Informally speaking, if no adversary can do so with high probability, then we consider H to be preimage aware. We now turn to formalizing a notion based on this intuition, but which allows multiple, adaptive attempts by the adversary to fool the extractor.

Fix sets $Dom \subseteq \{0,1\}^*$ and Rng, and let A be an adversary that outputs a string $x \in Dom$. In the *preimage awareness* (pra) experiment defined in Figure 1, the adversary is provided with two oracles. First, an oracle P that provides access to the (ideal) primitive P, but which also records all the queries and their responses in an advice string α. (We assume that when P is providing an interface to multiple primitives, it is clear from the advice string to which primitive each query was made.) Second, an *extraction oracle* Ex. The extraction oracle provides an interface to an *extractor* $\mathcal{E}$, which is a deterministic algorithm that takes as input a point $z \in Rng$ and the advice string α, and returns a point in $Dom \cup \{\perp\}$.

For hash function H, adversary A, and extractor $\mathcal{E}$, we define the advantage relation

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

where the probabilities are over the coins used in running the experiments. We will assume that an adversary never asks a query outside of the domain of the queried oracle. We use the convention that the running time of the adversary A does not include the time to answer its queries (i.e. queries are unit cost). When there exists an efficient extractor $\mathcal{E}$ such that $\mathbf{Adv}^{\mathrm{pra}}_{H,\mathcal{E}}(A)$ is small for all reason-

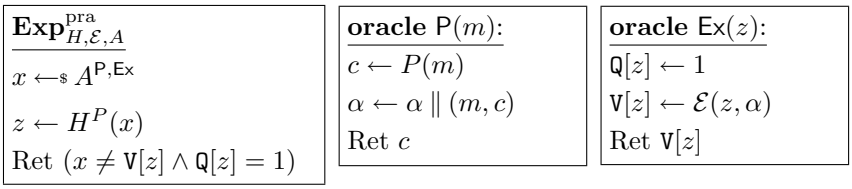

Fig. 1. (Left) Experiment for defining preimage awareness (PrA) for hash function H, extractor $\mathcal{E}$ and adversary A. **(Right)** Description of the oracles used in the PrA experiment extractor $\mathcal{E}$. The (initially empty) advice string α, the (initially empty) array $\mathtt{V}$, and the (initially everywhere $\perp$) array $\mathtt{Q}$ are global.

able adversaries A, we say that the hash function H is preimage aware (PrA). (Here "efficient", "small", and "reasonable" are meant informally.)

Remarks. As mentioned, the above formalization allows multiple, adaptive challenge queries to the extraction oracle. This notion turned out to be most convenient in applications. One can instead restrict the above notion to a single query (or to not allow adaptivity) resulting in a definition with slightly simpler mechanics. In the full version [21] we discuss such alternative formulations of preimage awareness.

4 Relationships between PrA, CR, and Random Oracles

Our new notion preimage awareness is an interesting middle point in the continuum between objects that are CR (on one end) and those that are random oracles (on the other). More formally speaking, we'll see in a moment that preimage awareness is a strictly stronger notion than CR while it is easy to see that it is a strictly weaker notion than indifferentiability from a random oracle. This is interesting for several reasons. First, we show that a PrA function is a secure domain extender for fixed-input-length random oracles, unlike CR functions [16]. (This already suggests that CR does not necessarily imply PrA.). Preimage awareness is consequently a very useful strengthening of CR, not to mention that it provides rigor to the folklore intuition that CR functions are insufficient for this application due to a lack of extractability. Second, the MD transform preserves preimage awareness. This is in stark contrast to the fact that MD (even if one uses strengthening) does *not* preserve indifferentiability from a random oracle (i.e. PRO-Pr). In the rest of this section, we explore these facts in more detail.

One can view preimage awareness as a strengthening of collision resistance in the following way. Say that queries to P allow the adversary to compute distinct domain points x, x' such that $H^P(x) = H^P(x') = z$. The adversary can make an extraction query on z, and then succeed in the PrA game by returning whichever of x and x' is not extracted from (z, α) by the extractor.

On the other hand, it is not hard to see that a RO is a PrA function. (consider an extractor that simply scans the advice string looking for the challenge

point z). We now turn to the two claims mentioned above. The next theorem captures that any PrA function is a good domain extender for an FIL random oracle. The proof is given in the full version [21].

Theorem 1. [RO domain extension via PrA] Let P be an ideal primitive and $H^P\colon \mathit{Dom} \to \mathit{Rng}$ be a hash function. Let R be an ideal primitive with two interfaces that implements independent functionalities P and $\mathcal{R} = \mathsf{RF}_{\mathit{Rng},\mathit{Rng}}$. Define $F^R(M) = \mathcal{R}(H^P(M))$. Let $\mathcal{F} = \mathsf{RF}_{\mathit{Dom},\mathit{Rng}}$. Let $\mathcal{E}$ be an arbitrary extractor for H. Then there exists a simulator $\mathcal{S} = (\mathcal{S}_1, \mathcal{S}_2)$ such that for any PRO adversary A making at most (q_0, q_1, q_2) queries to its three oracle interfaces, there exists a PrA adversary B such that

$$\mathbf{Adv}^{\mathrm{pro}}_{F,\mathcal{S}}(A) \le \mathbf{Adv}^{\mathrm{pra}}_{H,\mathcal{E}}(B)\ .$$

Simulator $\mathcal{S}$ runs in time $\mathcal{O}(q_1 + q_2 \cdot \mathsf{Time}(\mathcal{E}))$. Let ℓ_{max} the the length (in bits) of the longest query made by A to it's first oracle. Adversary B runs in time that of A plus $\mathcal{O}(q_0 \cdot \mathsf{Time}(H, \ell_{max}) + q_1 + q_2)$, makes $q_1 + q_0 \cdot \mathsf{NumQueries}(H, \ell_{max})$ primitive queries, q_2 extraction queries, and outputs a preimage of length at most ℓ_{max}. □

Theorem 1 shows that preimage awareness is a strong enough notion to provide secure domain extension for random oracles. At the same time, the next theorem shows that it is "weak" enough to be preserved by SMD. We consider SMD based on any suffix-free padding function $\mathsf{sfpad}\colon \{0,1\}^* \to (\{0,1\}^d)^+$ that is injective. Further we assume it is easy to strip padding, namely that there exists an efficiently computable function $\mathsf{unpad}\colon (\{0,1\}^d)^+ \to \{0,1\}^* \cup \{\perp\}$ such that $x = \mathsf{unpad}(\mathsf{sfpad}(x))$ for all $x \in \{0,1\}^*$. Inputs to unpad that are not valid outputs of sfpad are mapped to $\perp$ by unpad.

Theorem 2. [SMD is PrA-preserving] Fix $n, d > 0$ and let P be an ideal primitive. Let $h^P\colon \{0,1\}^{n+d} \to \{0,1\}^n$ be a compression function, and let $H = \mathrm{SMD}[h^P]$. Let $\mathcal{E}_h$ be an arbitrary extractor for the PrA-experiment involving h. Then there exists an extractor $\mathcal{E}_H$ such that for all adversaries A making at most q_p primitive queries and q_e extraction queries and outputting a message of at most $\ell_{max} \ge 1$ blocks there exists an adversary B such that

$$\mathbf{Adv}^{\mathrm{pra}}_{H,\mathcal{E}_H}(A) \le \mathbf{Adv}^{\mathrm{pra}}_{h,\mathcal{E}_h}(B)\ .$$

$\mathcal{E}_H$ runs in time at most $\ell_{max}\,(\mathsf{Time}(\mathcal{E}_h) + \mathsf{Time}(\mathsf{unpad}))$. B runs in time at most that of A plus $\mathcal{O}(q_e \ell_{max})$, makes at most $\ell_{max} \cdot \mathsf{NumQueries}(h, \ell_{max}) + q_p$, and makes at most $q_e \ell_{max}$ extraction queries. □

Proof. We start by defining the adversary B; the extractor $\mathcal{E}_H$ is implicit in its description.

adversary $B^{\mathsf{P},\mathsf{Ex}}(\varepsilon)$:

$x^* \xleftarrow{\$} A^{\mathsf{P},\mathsf{SimEx}}$
$x^*_\ell \cdots x^*_1 \xleftarrow{d} \mathsf{sfpad}(x^*)$; $c^*_{\ell+1} \leftarrow IV$
For $i = \ell$ down to 1 do
  $c^*_i \leftarrow h^{\mathsf{P}}(c^*_{i+1} \,\|\, x^*_i)$
  If $\mathtt{Q}[c^*_i] = 1$ and $\mathtt{E}[c^*_i] \neq c^*_{i+1} \,\|\, x^*_i$ then
    Ret $c^*_{i+1} \,\|\, x^*_i$
Ret $\perp$

subroutine $\mathsf{SimEx}(z, \alpha)$:

$i \leftarrow 1$; $c_1 \leftarrow z$
While $i \leq \ell_{max}$ do
  $c_{i+1} \,\|\, x_i \leftarrow \mathsf{Ex}(c_i, \alpha)$
  $\mathtt{Q}[c_i] \leftarrow 1$; $\mathtt{E}[c_i] \leftarrow c_{i+1} \,\|\, x_i$
  If $c_{i+1} = \perp$ then Ret $\perp$
  $x \leftarrow \mathsf{unpad}(x_i \cdots x_1)$
  If $c_{i+1} = IV$ and $x \neq \perp$ then
    Ret x
  $i \leftarrow i + 1$
Ret $\perp$

Adversary B answers A's primitive queries by forwarding to its own oracle P. It answers A's extraction queries using the subroutine SimEx (which makes use of B's extraction oracle). The code $c_{i+1} \,\|\, x_i \leftarrow \mathsf{Ex}(c_i, \alpha)$ means take the string returned from the query and parse it into an n-bit string c_{i+1} and a d-bit string x_i. If the oracle returns $\perp$, then c_{i+1} and x_i are both assigned $\perp$. The code $x^*_\ell \cdots x^*_1 \xleftarrow{d} \mathsf{sfpad}(x^*)$ means take the output of $\mathsf{sfpad}(x^*)$ and parse it into ℓ d-bit blocks $x^*_\ell, \ldots, x^*_1$. The tables $\mathtt{Q}$ and $\mathtt{E}$, which record if a value was queried to Ex and the value returned by the query, are initially everywhere $\perp$.

The extractor $\mathcal{E}_H$ works exactly the same as the code of SimEx except that queries to Ex are replaced by directly running $\mathcal{E}_h$ and the tables $\mathtt{Q}$ and $\mathtt{E}$ can be omitted. Loosely, extractor $\mathcal{E}_H$, when queried on a challenge image z, uses $\mathcal{E}_h$ to compute (backwards) the preimages of each iteration of h leading to z. When a chaining variable equal to IV is extracted, the function unpad is applied to the extracted message blocks. If it succeeds, then the result is returned.

Note that we reverse the (usual) order of indices for message blocks and chaining variables (starting high and counting down, e.g. $x^*_\ell \cdots x^*_1$) for both the extractor and B due to the extractor working backwards.

To lower bound B's advantage by the advantage of A we first point out that, by construction of $\mathcal{E}_H$, the values returned by the simulated SimEx are distributed identically to the values returned during execution of $\mathbf{Exp}^{\mathrm{pra}}_{H,\mathcal{E}_H,A}$. Thus we have that $\mathbf{Adv}^{\mathrm{pra}}_{H,\mathcal{E}_H}(A) = \Pr[x^* \text{ satisfies}]$ where the event "x^* satisfies", defined over the experiment $\mathbf{Exp}^{\mathrm{pra}}_{h,\mathcal{E}_B,B}$, occurs when the message x^* satisfies the conditions of winning for A. Namely that $H^P(x^*)$ was queried to SimEx and the reply given was not equal to x^*. We call x^* a satisfying preimage for A. We will show that whenever x^* is a satisfying preimage for A, with $x^*_\ell \cdots x^*_1 \xleftarrow{d} \mathsf{sfpad}(x^*)$, there exists a k with $1 \leq k \leq \ell$ for which adversary B returns the string $c^*_{k+1} \,\|\, x^*_k$ and this string is a satisfying preimage for B (i.e. one that wins the PrA experiment against h for B). This will establish that

$$\Pr[\, x^* \text{ satisfies} \,] \leq \mathbf{Adv}^{\mathrm{pra}}_{h,\mathcal{E}_h}(B) \,. \tag{1}$$

Consider the query $\mathsf{SimEx}(H^P(x^*))$ necessarily made by A. Let $c_{j+1} \| x_j, \ldots, c_2 \| x_1$ be the sequence of values returned by the Ex queries made by SimEx in the course of responding to A's query. Necessarily $1 \leq j \leq \ell_{max}$ and $1 \leq \ell \leq \ell_{max}$.

We will show that there exists a k such that $1 \leq k \leq \min\{j, \ell\}$ and $c_{k+1} \| x_k \neq c^*_{k+1} \| x^*_k$. (This includes the possibility that $c_{k+1} = \perp$ and $x_k = \perp$.) First we use this fact to conclude. Since $k \leq j$ it means that c_k was queried to Ex. If $c_k = c^*_k = H^P(c^*_{k+1} \| x^*_k)$ we are done, because then $c^*_{k+1} \| x^*_k$ is a satisfying preimage for B. Otherwise, $c_k \neq c^*_k$ and we can repeat the reasoning for $k-1$. At $k = 1$ we have that, necessarily, $c_k = c^*_k$ since this was the image queried by A. Thus there must exist a satisfying preimage, justifying (1).

We return to showing the existence of k such that $1 \leq k \leq \min\{j, \ell\}$ and $c_{k+1} \| x_k \neq c^*_{k+1} \| x^*_k$. Assume for contradiction that no such k exists, meaning that $c^*_{i+1} \| x^*_i = c_{i+1} \| x_i$ for $1 \leq i \leq \min\{j, \ell\}$. If $j > \ell$, then since $c_j = IV$ and $x^*_\ell \cdots x^*_1 = x_\ell \cdots x_1$ we have a contradiction because in such a situation the loop in SimEx would have halted at iteration ℓ. If $j = \ell$, then having $x^*_\ell \cdots x^*_1 = x_\ell \cdots x_1$ and $c_{\ell+1} = c^*_{\ell+1} = IV$ would imply that SimEx returned $x = x^*$, contradicting that x^* is a satisfying preimage for A. If $j < \ell$, then the loop in SimEx must have stopped iterating because $c_{j+1} = IV$ (if $c_{j+1} = \perp$ we would already have contradicted our assumption regarding k) and $x \neq \perp$. But by assumption we have that $x^*_j \cdots x^*_1 = x_j \cdots x_1$ and so there exist two strings x and x^* for which $\mathsf{sfpad}(x)$ is a suffix of $\mathsf{sfpad}(x^*)$. This contradicts that sfpad provides a suffix-free encoding. ∎

Recall that if a compression function h is both CR and hard to invert for range point the IV, then the plain MD iteration of h is a CR function [17, 20]. We prove an analogous theorem for plain MD and preimage awareness in [21]. This is particularly useful in our context, because for the compression functions we will consider (e.g. a FIL random oracle or an ideal cipher based compression function) it is easy to verify that it is difficult to invert a fixed range point. Note that this extra property on h (difficulty of inverting IV) is, in fact, *necessary* for plain MD to provide preimage awareness.

5 Applying Preimage Awareness

The results of the previous section allow us to more elegantly and modularly prove that a hash function construction is a pseudorandom oracle (PRO). Particularly, Theorems 1 and 2 mean that the task of building a PRO is reduced to the significantly simpler task of building a compression function that is PrA. For example, in the case that the compression function is itself suitable to model as an FIL-RO, then it is trivially PrA and so one is finished. However, even if the compression function has some non-trivial structure, such as when based on a block cipher, it is still (relatively) straightforward to prove (suitable compression functions) are PrA. In the rest of this section we show that most CR functions built from an ideal primitive are, in fact, PrA.

Are there applications of preimage awareness beyond analysis of hash functions? We believe the answer is yes. For example, one might explore applications of CR functions, instead analyzing these applications assuming a PrA function. (As one potential application, the CR-function using statistically hiding com-

mitment scheme of [18] conceivably achieves straight-line extractability given instead a PrA function.) We leave such explorations to future work.

PrA FOR CR CONSTRUCTIONS. There is a long line of research [29, 7, 23, 24, 32, 33, 31, 19] on building compression functions (or full hash functions) that are provably collision-resistant in some idealized model, e.g. the ideal cipher model. We show that in most cases one can generalize these results to showing the constructions are also PrA. We start by showing that the Davies-Meyer and other so-called "type-1" PGV compression functions [29, 7] are not only CR but PrA. We also give bounds on the PrA-security of the Shrimpton-Stam compression function [33] (see Theorem 4) and the first two steps of the MCM construction [30] (see Theorem 5); previously these were only known to be CR.

Let us begin by examining the Davies-Meyer compression function, which is defined by $\mathsf{DM}^E(c, m) = E_m(c) \oplus c$ where E is a block cipher. This compression function and the rest of the 12 "type-1" PGV [29] block cipher-based compression functions were proved to be collision-resistant (to the birthday bound) in the ideal-cipher model in [7]. We leverage their results in the proof of the following theorem, given in the full version [21] where results for the other "type-1" functions also appear.

Theorem 3. [Davies-Meyer is PrA.] Fix $\kappa, n > 0$, let $E \leftarrow\!\!\$\ \mathsf{BC}(\kappa, n)$. Let P be an oracle providing an interface to E and E^{-1}. Let $H^P(c, m) = \mathsf{DM}^E(c, m)$. There exists an extractor $\mathcal{E}$ such that for any adversary A making at most q_p queries to P and q_e extraction queries we have

$$\mathbf{Adv}^{\mathrm{pra}}_{H,\mathcal{E}}(A) \leq \frac{q_e q_p}{2^{n-1}} + \frac{q_p(q_p+1)}{2^n}$$

where $\mathcal{E}$ runs in time at most $\mathcal{O}(q_p)$. □

Next we show that it is possible to build a PrA compression function from *non-compressing* random functions[5]. In particular, we examine the compression function recently designed by Shrimpton and Stam [33]. They proved that this compression function is nearly optimally collision resistant (i.e. to the birthday bound), and we will now show that it is also PrA. The proof of the following is in [21].

Theorem 4. [Shrimpton-Stam is PrA] Fix $n > 0$. Let $P = (f_1, f_2, f_3)$ be an ideal primitive providing interfaces to independent functionalities $f_1 = \mathsf{RF}_{n,n}, f_2 = \mathsf{RF}_{n,n}$ and $f_3 = \mathsf{RF}_{n,n}$. Define a compression function $H^P(c, m) = f_3(f_1(m) \oplus f_2(c)) \oplus f_1(m)$. Then there exists an extractor $\mathcal{E}$ such that for any adversary A making q_p queries to each of f_1, f_2, f_3 (via P) and q_e extraction query, we have

$$\mathbf{Adv}^{\mathrm{pra}}_{H,\mathcal{E}}(A) = \mathcal{O}(q_e q_p^2 / 2^n)$$

where the extractor runs in time $\mathcal{O}(q_p^2)$. □

[5] One can view a block cipher as a compressing primitive, since it takes $k+n$ bits and produces n bits.

Dodis et al. [19] also offer a compression function from non-compressing primitives, this being $f(c, m) = f_1(c) \oplus f_2(m)$. A straightforward extension of the argument in [19] shows that this function is PrA for ideal f_1 and f_2. See [21].

Finally, we show that the "mix-compress" portion of the "mix-compress-mix" construction from [30] is PrA as long as the compress step is CR and relatively balanced. First we must define a measure of balance. Associated to any function $F\colon \{0,1\}^* \to \{0,1\}^n$ is the set $\mathsf{Prelm}_F(\ell, z) = \{y \mid y \in \{0,1\}^* \wedge |y| = \ell \wedge F(y) = z\}$ for all $\ell > 0$ and $z \in \{0,1\}^n$. That is, $\mathsf{Prelm}_F(\ell, z)$ contains the length ℓ preimages of z under F. We also define the function $\delta_F(\ell, z) = |(|\mathsf{Prelm}_F(\ell, z)| - 2^{\ell-n})/2^{\ell}|$ related to F. The δ_F function measures how far a particular preimage set deviates from the case in which F is regular. Let $\Delta_F = \max\{\delta_F(\ell, z)\}$, where the maximum is taken over all choices of ℓ and z. The proof of the following is given in [21].

Theorem 5. [Mix-Compress is PrA.] Fix $\tau, n > 0$, let $F\colon \{0,1\}^* \to \{0,1\}^n$ and let P be an ideal primitive implementing $\mathsf{RF}_{*,\tau}$. Let $H^P(m) = F(P(m))$. Let A be a PrA adversary making q_p primitive queries and q_e extraction queries. Then there exists a CR adversary B and an extractor $\mathcal{E}$ such that

$$\mathbf{Adv}^{\mathrm{pra}}_{H,\mathcal{E}}(A) \leq q_e q_p(\frac{1}{2^n} + \Delta_F) + \mathbf{Adv}^{\mathrm{cr}}_{H,F}(B)$$

$\mathcal{E}$ runs in time at most $\mathcal{O}(q_p)$. B runs in time at most that of A plus $\mathcal{O}(q_p)$. □

6 Indifferentiability for Public-Use Random Oracles

In numerous applications, hash functions are applied only to public messages. Such public-use occurs in most signature schemes (e.g. full-domain-hash [4], probabilistic FDH [15], Fiat-Shamir [22], BLS [10], PSS [6]) and even some encryption schemes (e.g. a variant of Boneh-Franklin IBE [14] and Boneh-Boyen IBE [8]). It is easy to verify that the provable security of such schemes is retained even if all hashed messages are revealed to adversaries. We introduce the notion of a public-use random oracle (pub-RO). This is an ideal primitive that exposes two interfaces: one which performs the usual evaluation of a random oracle on some domain point and a second which reveals all so-far evaluated domain points. All parties have access to the first interface, while access to the latter interface will only be used by adversaries (and simulators).

A wide class of schemes that have proofs of security in the traditional random oracle model can easily be shown secure in this public-use random oracle model. Consider any scheme and security experiment for which all messages queried to a RO can be inferred from an adversary's queries during the experiment. Then one can prove straightforwardly the scheme's security in the pub-RO model, using an existing proof in the full RO model as a "black box". For example, these conditions are met for unforgeability under chosen-message attacks of signature schemes that use the RO on messages and for message privacy of IBE schemes that use the RO on adversarially-chosen identities. All the schemes listed in the previous paragraph (and others) fall into these categories.

The pub-RO model was independently considered by Yoneyama et al. [36] (there called the leaky random oracle model) under different motivation. They directly prove some schemes secure when hash functions are modeled as a *monolithic* pub-RO. They do not analyze the underlying structure of MD-based functions.

We utilize the indifferentiability framework of Maurer et al. [26] to formalize a new notion of security for hash constructions: indifferentiability from a public-use RO, which we will call being a *public-use pseudorandom oracle* (pub-PRO). This new security property is weaker than that of being a PRO. We show that iterating a fixed-input-length public-use random oracle (i.e. a compression function) via MD yields a variable-input-length public-use random oracle. Put another way, MD preserves the property of being a pub-PRO.

PUBLIC-USE ROS. Fix sets Dom, Rng. A public-use random oracle (pub-RO) for domain Dom and range Rng is an ideal primitive $\mathcal{F} = (\mathcal{F}_{eval}, \mathcal{F}_{reveal})$ defined as follows. Let $\rho = \mathsf{RF}_{Dom,Rng}$. The evaluation interface $\mathcal{F}_{eval}$, on input $M \in Dom$, first adds the pair $(M, \rho(M))$ to an initially-empty set $\mathcal{M}$ and then returns $\rho(M)$. The reveal interface $\mathcal{F}_{reveal}$ takes no input and returns $\mathcal{M}$ (suitably encoded into a string). We say that $\mathcal{F}$ is a fixed-input-length (FIL) pub-RO if Dom only includes messages of a single length.

INDIFFERENTIABILITY FROM A PUB-RO. Fix sets Dom, Rng. Let P be an ideal primitive and let $\mathcal{F} = (\mathcal{F}_{eval}, \mathcal{F}_{reveal})$ be a pub-RO for domain Dom and range Rng. Let $\mathcal{S}$ be a simulator with oracle access to (both interfaces of) $\mathcal{F}$. Then we define the pub-pro advantage of an adversary A against a construction H^P by

$$\mathbf{Adv}^{\text{pub-pro}}_{H,\mathcal{S}}(A) = \Pr\left[\, A^{H,P} \Rightarrow 1 \,\right] - \Pr\left[\, A^{\mathcal{F}_{eval},\mathcal{S}} \Rightarrow 1 \,\right]$$

where the first probability is over the coins used by A and primitive P, and the second is over the coins used by A, $\mathcal{F}$, and $\mathcal{S}$. In the second probability experiment, while A has access only to $\mathcal{F}_{eval}$, the simulator $\mathcal{S}$ has oracle access to both interfaces of $\mathcal{F}$. The simulator's ability to call $\mathcal{F}_{reveal}$, thereby seeing all queries so-far-made by A to $\mathcal{F}_{eval}$, is the crucial difference between pub-PRO and PRO.

The composition theorem in [26] (recast to use ITMs in [16]) can be applied to pub-PROs. That is, a cryptographic scheme using a pub-PRO hash construction H^P for some ideal primitive P can have its security analyzed in a setting where H^P is replaced by a monolithic pub-RO $\mathcal{F}$. In this setting, adversaries attacking the scheme can perform queries to $\mathcal{F}_{reveal}$.

MERKLE-DAMGÅRD PRESERVES PUB-PRO. Let $f = (f_{eval}, f_{reveal})$ be a FIL pub-RO. Then the next theorem statement, whose proof appears in [21], asserts that MD$[f_{eval}]$ is indifferentiable from a pub-RO. (That SMD$[f_{eval}]$ is a pub-PRO is an immediate corollary.)

Theorem 6. [MD preserves pub-PRO] Fix $n, d > 0$. Let $f = (f_{eval}, f_{reveal})$ be a FIL pub-RO for domain $\{0,1\}^{n+d}$ and range $\{0,1\}^n$. There exists a simu-

lator $\mathcal{S} = (\mathcal{S}_{eval}, \mathcal{S}_{reveal})$ so that for any adversary A

$$\mathbf{Adv}^{\text{pub-pro}}_{\text{MD},\mathcal{S}}(A) \leq \frac{(\sigma q_0 + q_1)^2}{2^n} + \frac{\sigma q_0 + q_1 + 1}{2^n}$$

where q_0 is the maximal number of queries by A to its first oracle, these of length at most σ blocks of d bits, and q_1 is the maximal number of queries by A to either f_{eval} or $\mathcal{S}_{eval}$. Let q_2 be the number of queries by A to either f_{reveal} or $\mathcal{S}_{reveal}$. Then S runs in time that of A plus $\mathcal{O}(q_0\sigma(q_1 + q_2))$ and makes at most $2q_0 + q_0q_1\sigma$ queries. □

Davies-Meyers compression function. One might hope that the Davies-Meyers compression function is pub-PRO analogously to the fact that it is PrA. Unfortunately, this is not the case. Consider the following attack, due to [35]. Let A against $\mathsf{DM}^E(c, x) = E_x(c) \oplus c$ work as follows. It picks a random z and m and then queries its third oracle interface on m, z. When interacting with the pub-RO $\mathcal{F}$ and any simulator $\mathcal{S}$, we see that $\mathcal{S}$ would need to respond with a value c such that $\mathcal{F}_{eval}(c, x) = c \oplus z$. This corresponds to inverting $\mathcal{F}$ on some fixed range point, which is hard. (Note that A has not, before querying the simulator, submitted any queries to $\mathcal{F}$.) Thus the adversary will win easily. On the other hand, we conjecture that although DM is not itself pub-PRO, applying MD to it results in a VIL pub-PRO (in the ideal cipher model). We discuss this in more detail in the full version [21].

Acknowledgments

We thank Ilya Mironov, Martijn Stam, and Mihir Bellare for useful discussions regarding this work. We thank Lei Wang for pointing out an error in an earlier version of this work. Yevgeniy Dodis was supported in part by NSF Grants 0831299, 0716690, 0515121, and 0133806. Thomas Ristenpart was supported in part by Mihir Bellare's NSF grants CNS 0524765 and CNS 0627779 and a gift from Intel corporation. He thanks the Faculty of Informatics at the University of Lugano, Switzerland for hosting and supporting him while a portion of this work was done. Thomas Shrimpton was supported by NSF grant CNS 0627752 and SNF grant 200021-122162.

References

1. M. Bellare and A. Palacio. Towards Plaintext-Aware Public-Key Encryption without Random Oracles. *Advances in Cryptology – ASIACRYPT '04*, LNCS vol. 3329, pp. 48–62, 2004.
2. M. Bellare and T. Ristenpart. Multi-property-preserving Hash Domain Extension and the EMD Transform. *Advances in Cryptology – ASIACRYPT '06*, LNCS vol. 4284, Springer, pp. 299–314, 2006.
3. M. Bellare and T. Ristenpart. Hash Functions in the Dedicated-key Setting: Design Choices and MPP Transforms. *International Colloquium on Automata, Languages, and Programming – ICALP '07*, LNCS vol. 4596, Springer, pp. 399–410, 2007.

4. M. Bellare and P. Rogaway. Random oracles are practical: a paradigm for designing efficient protocols. In: CCS '93, ACM Press (1993) 62–73.
5. M. Bellare and P. Rogaway. Optimal asymmetric encryption – How to encrypt with RSA. *Advances in Cryptology – EUROCRYPT '94*, LNCS vol. 950, pp. 92–111, 1994.
6. M. Bellare and P. Rogaway. 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.
7. J. Black, P. Rogaway, and T. Shrimpton. Black-Box Analysis of the Block-Cipher-Based Hash-Function Constructions from PGV.. *Advances in Cryptology – CRYPTO '02*, LNCS vol. 2442, Springer, pp. 320–325, 2002.
8. D. Boneh and X. Boyen. Efficient Selective-ID Secure Identity Based Encryption Without Random Oracles. *Advances in Cryptology – EUROCRYPT '04*, LNCS vol. 3027, Springer, pp. 223–238, 2004.
9. D. Boneh and M. Franklin. Identity-Based Encryption from the Weil Pairing. *Advances in Cryptology – CRYPTO '01*, LNCS vol. 2139, Springer, pp. 213–229, 2001.
10. D. Boneh, B. Lynn, and H. Shacham. Short signatures from the Weil pairing. *Advances in Cryptology – ASIACRYPT '01*, LNCS vol. 2248, Springer, pp. 514–532, 2001.
11. R. Canetti and R. Dakdouk. Extractable Perfectly One-Way Functions. *ICALP '08*, LNCS vol. 5126, Springer, pp. 449–460, 2008.
12. R. Canetti and R. Dakdouk. Towards a Theory of Extractable Functions. *Theory of Cryptography Conference – TCC '09*, 2009, to appear.
13. R. Canetti, O. Goldreich, and S. Halevi. The random oracle methodology, revisited. J. ACM **51**(4), pp. 557–594, 2004.
14. R. Canetti, S. Halevi, and J. Katz. A Forward-Secure Public-Key Encryption Scheme. J. Cryptology (JOC) 20(3):265-294 (2007)
15. J.S. Coron. Optimal Security Proofs for PSS and Other Signature Schemes. *Advances in Cryptology – EUROCRYPT '02*, LNCS vol. 2332, Springer, pp. 272–287, 2002.
16. J.S. Coron, Y. Dodis, C. Malinaud, and P. Puniya. Merkle-Damgard Revisited: How to Construct a Hash Function. *Advances in Cryptology – CRYPTO '05*, LNCS vol. 3621, Springer, pp. 21–39, 2005.
17. I. Damgård. A design principle for hash functions. *Advances in Cryptology – CRYPTO '89*, LNCS vol. 435, Springer, pp. 416–427, 1989.
18. I. Damgård, T. Pedersen, and B. Pfitzmann. On the Existence of Statistically Hiding Bit Commitment Schemes and Fail-Stop Signatures. *Advances in Cryptology – CRYPTO '93*, LNCS vol. 773, Springer, pp. 250–265, 1993.
19. Y. Dodis, K. Pietrzak, and P. Puniya. A New Mode of Operation for Block Ciphers and Length-Preserving MACs. *Advances in Cryptology – EUROCRYPT '08*. LNCS vol. 4965, Springer, pp. 198–219, 2008.
20. Y. Dodis and P. Puniya. Getting the Best Out of Existing Hash Functions or What if We Are Stuck with SHA?. *Applied Cryptography and Network Security – ACNS '08*. LNCS vol. 5037, Springer, pp. 156–173, 2008.
21. Y. Dodis, T. Ristenpart, and T. Shrimpton. Salvaging Merkle-Damgård for Practical Applications (full version of this paper). IACR ePrint Archive, 2009.
22. A. Fiat and A. Shamir. How to Prove Yourself: Practical Solutions to Identification and Signature Problems. *Advances in Cryptology – CRYPTO '86*, LNCS vol. 263, Springer, pp. 186–194, 1987.

23. S. Hirose. Provably Secure Double-Block-Length Hash Functions in a Black-Box Model. *Information Security and Cryptology – ICISC '04*, LNCS vol. 3506, Springer, pp. 330–342, 2005.
24. S. Hirose. Some Plausible Constructions of Double-Length Hash Functions. *Fast Software Encryption – FSE '06*, LNCS vol. 4047, Springer, pp. 210–225, 2006.
25. S. Hirose, J. Park, and A. Yun. A Simple Variant of the Merkle-Damgård Scheme with a Permutation. Advances in Cryptology – ASIACRYPT '07, LNCS vol. 4833, Springer, pp. 113–129, 2007.
26. U. Maurer, R. Renner, and C. Holenstein, Indifferentiability, Impossibility Results on Reductions, and Applications to the Random Oracle Methodology. *Theory of Cryptography Conference – TCC '04*, LNCS vol. 2951, Springer, pp. 21–39, 2004.
27. R. Merkle. One way hash functions and DES. *Advances in Cryptology – CRYPTO '89*, LNCS vol. 435, Springer, pp. 428–446, 1989.
28. National Institute of Standards and Technology. FIPS PUB 180-1: Secure Hash Standard. (1995) Supersedes FIPS PUB 180 1993 May 11.
29. B. Preneel, R. Govaerts, and J. Vandewalle. Hash functions based on block ciphers: A synthetic approach. *Advances in Cryptology – CRYPTO'93*, LNCS vol. 773, Springer, pp. 368–378, 1994.
30. T. Ristenpart and T. Shrimpton. How to Build a Hash Function from any Collision-Resistant Function. *Advances in Cryptology – ASIACRYPT '07*. LNCS vol. 4833, Springer, pp. 147–163, 2007.
31. P. Rogaway and J. Steinberger. Security/Efficiency Tradeoffs for Permutation-Based Hashing. *Advances in Cryptology – EUROCRYPT '08*, LNCS vol. 4965, Springer, pp. 220–365, 2008.
32. P. Rogaway and J. Steinberger. Constructing Cryptographic Hash Functions from Fixed-Key Blockciphers. *Advances in Cryptology – CRYPTO '08*, LNCS vol. 5157, Springer, pp. 443–450, 2008.
33. T. Shrimpton and M. Stam. Building a Collision-Resistant Compression Function from Non-compressing Primitives. *ICALP '08*, LNCS vol. 5126, Springer, pp. 643-654, 2008.
34. D. Simon. Finding Collisions on a One-Way Street: Can Secure Hash Functions Be Based on General Assumptions? *Advances in Cryptology – EUROCRYPT '98*, LNCS vol. 1403, Springer, pp. 334-345, 1998.
35. L. Wang. Personal correspondence. 2009.
36. K. Yoneyama, S. Miyagawa, and K. Ohta. Leaky Random Oracle (Extended Abstract). Provable Security – ProvSec '08, LNCS vol. 5324, pp. 226–240, 2008.