A preliminary version of this paper appears in *Advances in Cryptology – EUROCRYPT ‘09*, Lecture Notes in Computer Science Vol. 5479, pp. 371–388, A. Joux ed., Springer-Verlag, 2009. This is the full version.

Salvaging Merkle-Damgård for Practical Applications

YEVGENIY DODIS [*] THOMAS RISTENPART [†] THOMAS SHRIMPTON [‡]

January 2009

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 show that a fixed-length compressing random oracle, as well as the currently used Davies-Meyer compression function (the latter analyzed in the ideal cipher model) are "strong enough" for the two specific weakenings of the random oracle that we develop. These weaker notions, described below, are quite natural and should be interesting in their own right:

- **Preimage Aware Functions**. Roughly, if an attacker found a "later useful" output y of the function, then it must "already know" the corresponding preimage x. We show that this notion works well with the Merkle-Damgård transform (unlike fixed-length random oracles), and has many applications. Most notably, it yields a variable-length random oracle, when composed with a fixed-length random oracle. Additionally, (compressing) preimage aware functions considerably generalize collision-resistant hash functions. Moreover, we show that existing block-cipher-based hash functions, originally only shown collision-resistant in the ideal cipher model, are in fact preimage aware.
- **Public-Use Random Oracles**. Roughly, these objects are indifferentiable from ordinary random oracles, but only when they are never evaluated on secret inputs. We show that such public-use oracles are enough to argue security of most hash-based signature schemes, including Full Domain Hash and Fiat-Shamir signatures. Moreover, the Merkle-Damgård transform preserves this notion. As a result, all "public-use" applications of random oracles are still secure with existing hash functions (assuming a strong enough compression function, such as a fixed-length random oracle or the Davies-Meyer function).

Keywords: hash functions, random oracle model, indifferentiability framework

[*] Dept. of Computer Science, New York University. 251 Mercer St. New York, NY 10012, USA. Email: `dodis@cs.nyu.edu`. URL: `http://www.cs.nyu.edu/ dodis`

[†] Dept. of Computer Science & Engineering 0404, University of California San Diego, 9500 Gilman Drive, La Jolla, CA 92093-0404, USA. Email: `tristenp@cs.ucsd.edu`. URL: `http://www.cs.ucsd.edu/~tristenp`

[‡] Dept. of Computer Science, Portland State University, Room 120, Forth Avenue Building, 1900 SW 4th Avenue, Portland OR 97201 USA and Faculty of Informatics, University of Lugano Via Buffi 13, CH-6900 Lugano, Switzerland. Email: `teshrim@cs.pdx.edu`, `thomas.shrimpton@unisi.ch`. URL: `http://www.cs.pdx.edu/ teshrim`, `http://www.inf.unisi.ch/`

Contents

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 [30], were designed using the (strengthened) Merkle-Damgård (MD) transform [18, 29]: 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* [18, 29]. 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 (e.g. Fiat-Shamir [24] 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, 5, 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. Moreover, many important in-use cryptographic schemes only have provable security guarantees in the RO model.

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 [17]).

Consequently, Coron et al. [17] adapted the indifferentiability framework of Maurer et al. [28] 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 [17] (and several subsequent works [2, 3, 27]) 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 satisfies* X, as long as its compression function f satisfies some related property Y.

(4) Conclude that, as long as the compression function f satisfies Y, 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 some related property Y. Typically, if Y corresponds to being FIL-RO, it would be easy to conclude that the MD-transform satisfies X, which will already be quite useful. In our examples, however, we will be able to derive this conclusion for *considerably weaker* properties Y, which corresponds to a *wider class* of "admissible" compression functions f. For example, most in-use compression functions f are built from a block cipher E via the Davies-Meyer transform: $f(c, x) = E_x(c) \oplus c$. It was shown in [17] that this construction is *not* indifferentiable from a FIL-RO, even if E is assumed to be an ideal cipher. Despite this, in our examples we will be able to argue, in the ideal cipher model, that the Davies-Meyer compression function satisfies the property Y sufficient to prove that the iterated hash function H satisfies X. As a result, the resulting applications we consider are *provably secure with existing block cipher-based hash functions* (in the ideal-cipher model).

So which properties X (and Y)? We introduce two: *preimage awareness* and indifferentiability from a *public-use random oracle*. For preimage awareness, the corresponding property Y will also be preimage awareness, which means that the Merkle-Damgård transform is *property-preserving* for this new notion. For public-use random oracles, the property Y will be even *weaker* than public-use random oracles, which not only implies property-preservation, but will allow us to justify the use of the Davies-Meyer compression function (which is differentiable from a public-use random oracle, but satisfies this weaker notion). We detail these new notions below.

1.1 Preimage Aware Functions

A function being Preimage Aware (PrA) means, informally, that if an attacker can output a range point y and subsequently produce a preimage x for y, then in fact the attacker "already knew" x when it output y. To get an idea of how we formalize this, consider a hash function H built using some ideal primitive P (which could model a compression function or a block cipher). Then the PrA security experiment is loosely defined as follows. An attacker, using oracle access to P, first outputs a range point y. Then a deterministic algorithm called an *extractor* is run on y and the transcript of the attacker's interaction with P (the queries and their associated responses); it outputs a domain point x'. The attacker wins if it can (using further access to P) output a domain point $x \neq x'$ such that $H(x) = y$. Intuitively, this definition captures that producing a preimage-image pair under H requires actually evaluating H on the preimage in a manner that reveals it to anyone observing the attacker's oracle calls. 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 similarities in more detail, 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 [17]. 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 h), 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 [17] 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 indifferentiable from a FIL-RO.

Adopting our more modular point-of-view, several existing hash constructions in the literature [17, 2, 3, 32, 21] enjoy an easier analysis. For example, the NMAC construction of [17] becomes an example of our approach, where the outer h and the inner f are both implemented to be like (independent) FIL-ROs. In [17] 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 proofs of CR [7, 37]) 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 [31, 7, 25, 26, 34, 35, 33, 21] building FIL-CR hash functions from idealized blockciphers 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 [36] 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 FIL constructions proven CR in [7, 35, 33, 21] 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. [25, 26, 34], are also PrA.

We note that it is also possible to prove that a VIL-CR hash function is PrA even if the underlying compression function is not. Of course, in this case we must step outside of our modular approach (point (4), in particular). As an example, in Appendix C we show that the Group-2 blockcipher-based compression functions from [7, 31] (which are not even CR) do yield a PrA-hash when iterated.

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. [39] 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 [16], Fiat-Shamir [24], 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. [28]. 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. We call such a hash function a public pseudorandom oracle (pub-PRO).

MERKLE-DAMGÅRD CONSTRUCTS PUBLIC-USE ROS. 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.

PUBLIC-USE COMPRESSION FUNCTIONS. Our modular approach allows us to dig deeper, investigating the suitability of various compression function designs for use within MD to build hash functions that enjoy indifferentiability from a pub-RO. It is clear that a FIL RO or FIL pub-RO would suffice. Unfortunately, widely-used compression functions are not suitable for modeling as even pub-ROs, because they are based on block ciphers. (Briefly, that ciphers are invertible obviates hope of such compression functions being indifferentiable from a pub-RO.) This is doubly unfortunate since widely used hash functions, such as the SHA family, are constructed using such compression functions. We therefore formalize a further-restricted variant of a pub-RO for compression functions: a public-use *guarded* RO. Informally, this is an FIL RO that is used by honest parties only within the confines of the MD transform. (Dishonest parties can use the RO in arbitrary manners.) We go on to strengthen our preservation result regarding MD above to show that MD applied to any (object indifferentiable from a) public-use guarded RO results in a full public-use RO. Further, we go on to show that all of the PGV type-2 compression functions applied to an ideal cipher are indifferentiable from public-use guarded ROs. Note that this approach is still entirely modular, allowing independent (and simpler) analyses of compression function and transform.

DISCUSSION. Our results, combined with the composition theorem of [28], give a plethora of new, important provable security results. Namely, for any scheme only proven secure in the RO model and whose security is unaffected by public dissemination of hashed messages, our results give the *first ever* proofs of security (in the ideal cipher model) when using hash functions such as SHA-2. Since SHA-2 (and even SHA-1) will be in use for many years to come, these positive results importantly help explain when such hash functions are secure to use.

2 Preliminaries

When S is a set, $x \leftarrow\!\!\$\; S$ means to sample uniformly from S and assign the resultant value to x. We write D^+ for the set $(\{0,1\}^d)^+$. We write $x \leftarrow\!\!\$\; A$ to denote running algorithm A with fresh random coins and assigning its output to x. For $M \in \{0,1\}^*$, we write $M_1, \ldots, M_\ell \xleftarrow{d} M$ to denote (1) let $\ell = \lfloor |M|/d \rfloor$, (2) let M_i be assigned the i^{th} d-bit substring of M for $1 \le i \le \ell - 1$, and (3) let M_ℓ be the last $|M| \bmod d$ bits of M if $|M| \bmod d \neq 0$ and let M_ℓ be the last d bits of M otherwise. For set S, we write $S \xleftarrow{\cup} s$ to denote $S \leftarrow S \cup \{s\}$.

For any algorithm f that accepts inputs from $Dom \subseteq \{0,1\}^*$, we write $\mathsf{Time}(f, m)$ to mean the maximum time to run $f(x)$ for any input $x \in Dom$ such that $|x| \le m$. When f is a function with domain $Dom \subseteq \{0,1\}^*$, 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 $x \in Dom$ such that $|x| \le m$. 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 absolute constants that are dominated by 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. An *interface* of an ITM specifies that writing one of a certain subset of possible strings on the input tape invokes a particular functionality. For example, writing $i \,\|\, s$ (where the number i is suitably encoded as a string) results in executing f_i on s. When we write $P = (f_1, f_2, \ldots)$, this means that ITM P implements the functionalities $f_1, f_2, \ldots$ using some fixed interface semantics. 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 will (slightly abusing notation) write f_i to refer to accessing an ITM via interface f_i.

We sometimes distinguish between *private interfaces* and *public interfaces* (following terminology from [28]), writing $P = ((f_1, f_2, \ldots), (f'_1, f'_2, \ldots))$ to denote the ITM P that has private interfaces $f_1, f_2, \ldots$ and public interfaces $f'_1, f'_2, \ldots$. (Looking ahead, private interfaces will be used exclusively by honest parties while public interfaces will be used by adversaries and simulators.) We write M^P if an ITM M has access to the private interfaces of P and write $M^{P_{\text{pub}}}$ if M has access only to the public interfaces of P. If P does not have distinguished public and private interfaces, then the public interfaces are just the private interfaces. We write $M^{P_1, P_2, \cdots}$ to denote M having access to multiple (independent) ITMs $P_1, P_2, \ldots$.

Implicitly this means one defines a single ITM $\overline{P} = (P_1, P_2, \ldots)$ with interfaces for the independent functionalities and then give M unfettered access to $\overline{P}$.

IDEAL PRIMITIVES. 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. For non-empty sets Dom, Rng, a random oracle is the ideal primitive $\mathcal{F}_{Dom,Rng}$ with a single interface that consistently maps inputs in Dom to range points randomly chosen from Rng. When $Dom = \{0,1\}^d$ and $Rng = \{0,1\}^r$ for some d, r we write $\mathcal{F}_{d,n}$. We write $\mathcal{F}$ when Dom and Rng are clear from context. 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. An ideal cipher is the ideal primitive $\mathcal{C}_{\kappa,n} = (E, D)$ with two interfaces implementing a cipher chosen randomly from $\mathsf{BC}_{\kappa,n}$ and its inverse, respectively. We write $\mathcal{C}$ when κ and n are clear from context. (Note that in both cases all interfaces are private, and so accessed both by honest parties and adversaries alike.)

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 an algorithm that computes 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. If P has both private and public interfaces, then we use the convention that H^P means H uses the first private interface. 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 $x \in Dom$ such that $|x| \le m$.

For integers $n, d > 0$, we call a hash function $f^P\colon \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^n$ a *compression function* (using idealized primitive P). Let $v_0 = IV$ be a fixed n-bit string. Then the iteration of f^P, denoted by $\mathsf{Itr}[f^P]$, is the algorithm[1] that on input $M \in D^+$ first sets $m_1, \cdots, m_\ell \stackrel{d}{\leftarrow} M$, then computes $v_i \leftarrow f^P(v_{i-1}, m_i)$ iteratively for each $i \in [1 \mathinner{..} \ell]$, and returns v_ℓ. Let $\mathsf{sfpad}\colon \{0,1\}^* \to D^+$ be a *suffix-free padding* function which returns a suffix-free encoding of M. 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, pad an appropriate amount and append an encoding of the length of the message.) Let $\mathrm{SMD}[f^P]$ be the algorithm that on input M runs $\mathsf{sfpad}(M)$ and then applies $\mathsf{Itr}[f^P]$ to the result.

COLLISION RESISTANCE OF HASH FUNCTIONS. Fix sets $Dom \subseteq \{0,1\}^*$ and Rng and let A be an adversary that outputs a pair of strings $x, x' \in Dom$. Let P be an ideal primitive. To hash function $H^P : Dom \to Rng$ and adversary A we associate the advantage relation

$$\mathbf{Adv}^{\mathrm{cr}}_{H,P}(A) = \Pr\left[\, (x,x') \leftarrow\!\!\$\; A^P \;:\; H^P(x) = H^P(x') \wedge x \neq x' \,\right]$$

where the probability is over the coins used by A and primitive P.

THE INDIFFERENTIABILITY FRAMEWORK. We make extensive use of the indifferentiability framework of Maurer, Renner, and Holenstein [28], however we follow more closely the formalizations of it appearing in [17, 2, 3]. Let H be some cryptographic scheme (e.g. a hash function) that utilizes an ideal primitive P. Let Q be a second ideal primitive. A simulator, typically denoted by $\mathcal{S}$, is just an ITM revealing some number of interfaces. Informally, we say that H is indifferentiable from Q if there exists an efficient simulator $\mathcal{S}$ with an interface for each interface of P_{pub} such that for all "reasonable" adversaries A outputing a bit it is the case that

$$\Pr\left[\mathbf{Exp}^{\text{indiff-1}}_{H,P,A} \Rightarrow 1\right] - \Pr\left[\mathbf{Exp}^{\text{indiff-0}}_{Q,\mathcal{S},A} \Rightarrow 1\right]$$

is "small" where the probabilities are taken over the coins used the experiments shown in Figure 1. In the indiff-1 experiment H uses access to the (first) private interface of P while the adversary A has access to the public interfaces of P. In the indiff-0 experiment the adversary has access to Q's private interfaces while $\mathcal{S}$ has access to Q's public interfaces. Note that a crucial aspect of the framework is that the simulator, while able to query Q_{pub} itself, does *not* get to see the queries made by the adversary to Q_{priv}.

We shall formalize several security notions, based on the reference primitive Q that the scheme is compared against (e.g. see below). A key benefit of using indifferentiability is the composition theorem detailed in [28], which states that (intuitively) one can securely use H^P instead of Q in applications.

PSEUDORANDOM ORACLES. Fix non-empty sets Dom, Rng. Let P be an ideal primitive and let $\mathcal{F}_{Dom,Rng}$ be a random

[1]This construction is sometimes referred to as the Merkle-Damgård transform, seemingly due to [29, 18], however its use significantly predates these papers. See e.g. [20].

$\mathbf{Exp}^{\text{indiff-1}}_{H,P,A}$
$b \leftarrow A^{H^P, P_{\text{pub}}}$
Ret b

$\mathbf{Exp}^{\text{indiff-0}}_{Q,\mathcal{S},A}$
$b \leftarrow A^{Q, \mathcal{S}^{Q_{\text{pub}}}}$
Ret b

Figure 1: Experiments used in the indifferentiability framework for scheme H, adversary A, and ideal primitives P and Q.

$\mathbf{Exp}^{\text{pra}}_{H,P,\mathcal{E},A}$
$x \leftarrow\!\!\$\; A^{\mathsf{P},\mathsf{Ex}}$
$z \leftarrow H^P(x)$
Ret $(x \neq \mathtt{V}[z] \wedge \mathtt{Q}[z] = 1)$

oracle $\mathsf{P}(m)$:
$c \leftarrow P(m)$
$\alpha \leftarrow \alpha \,\|\, (m, c)$
Ret c

oracle $\mathsf{Ex}(z)$:
$\mathtt{Q}[z] \leftarrow 1$
$\mathtt{V}[z] \leftarrow \mathcal{E}(z, \alpha)$
Ret $\mathtt{V}[z]$

Figure 2: **(Left)** Experiment for defining preimage awareness (PrA) for hash function H, extractor $\mathcal{E}$ and adversary A. **(Center,Right)** Description of the oracles used in the PrA experiment. The (initially empty) advice string α, the (initially empty) array $\mathtt{V}$, and the (initially everywhere $\perp$) array $\mathtt{Q}$ are global.

oracle. We define the pro advantage of an adversary A against a function H^P mapping from Dom to Rng by

$$\mathbf{Adv}^{\text{pro}}_{H,P,\mathcal{S}}(A) = \Pr\left[\mathbf{Exp}^{\text{indiff-1}}_{H,P,A} \Rightarrow 1\right] - \Pr\left[\mathbf{Exp}^{\text{indiff-0}}_{\mathcal{F},\mathcal{S},A} \Rightarrow 1\right] .$$

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 outputs a range point z, possibly after interacting with an oracle for P. The extractor is then 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 continues 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 2, 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

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

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 $\mathbf{Adv}^{\text{pra}}_{H,P,\mathcal{E}}(A)$ is small for all reasonable 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 Appendix D we discuss such an alternative formulation of preimage awareness.

3.1 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 will show momentarily that a PrA function is also CR, and that a random oracle is PrA. The second point is fairly obvious, but the first is quite interesting. In particular, we will see in Section 4 that a PrA function is a secure domain extender for fixed-input-length random oracles, unlike CR functions [17]. (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. What is more, 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)

Let us begin with the formal results. 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.

Theorem 3.1 [PrA⇒CR] Let P be an ideal primitive and $H^P : \mathit{Dom} \to \mathit{Rng}$ be a hash function. Let $\mathcal{E}$ be an arbitrary extractor. Let A be a CR adversary against H asking a total of q_p queries to P. Then there exists a PrA adversary B such that

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

B runs in time that of A plus $\mathcal{O}(q_p) + \mathsf{Time}(H)$, asks at most q_p primitive queries, and one extraction query. □

Proof: The PrA adversary B starts by running A, using its oracle P to answer A's oracle queries. Eventually, A halts with output of two messages x_0, x_1. When it does, let B compute $z \leftarrow H^P(x_0)$. Then let B make the single query $x' \leftarrow \mathsf{Ex}(z)$. If $x' = x_0$ then B outputs x_1, otherwise it outputs x_0. ▌

On the other hand, it is not hard to see that a RO is a PrA function. The following theorem captures this, and its proof is in Appendix A.

Theorem 3.2 [ROs are PrA.] Fix $\mathit{Dom} \subseteq \{0,1\}^*$ and $n > 0$, let $P = \mathsf{RF}_{\mathit{Dom},n}$ Then the hash function $H^P(x) = P(x)$ is preimage-aware. Specifically, there exists an extractor $\mathcal{E}$ such that for all adversaries A making at most q_e queries to P and q_e extraction queries

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

Moreover, the running time of the extractor is $\mathcal{O}(q)$. □

3.2 Weak Preimage Awareness

Our proofs of preimage-awareness will be aided by considering a related notion that we call *weak preimage awareness* (WPrA) We define WPrA simply by modifying the pra experiment of Figure 2 so that the extractor, when queried on an image z, can return a set of potential preimages (instead of just a single preimage). The adversary wins if it can output a preimage x such that $H(x) = z$ yet x is not in the set returned by the extractor. While this weakening of PrA no longer implies CR, it will be useful for evaluating functions already proven CR.

Fix sets Dom and Rng. A *multi-point extractor* $\mathcal{E}^+$ is a deterministic algorithm that takes input a point $z \in \mathit{Rng}$ and outputs a set $\mathcal{X} \subseteq \mathit{Dom}$. Formally, let $\mathbf{Exp}^{\mathrm{wpra}}_{H,P,\mathcal{E}^+,A}$ work exactly like $\mathbf{Exp}^{\mathrm{pra}}_{H,P,\mathcal{E}^+,A}$ except that the last line of the pra experiment (see Figure 2) is changed to "Ret $(x \notin \mathtt{V}[z] \wedge \mathtt{Q}[z] \neq \bot)$". Then we associate to any hash function H, adversary A, and set extractor $\mathcal{E}$ the advantage relation

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

We say that a multi-point extractor $\mathcal{E}^+$ is *honest* if for any $z \in \mathit{Rng}$ and advice string α it is the case that

$$\Pr\left[\,\forall x \in \mathcal{X}\,.\, H^P(x) = z\ :\ \mathcal{X} \leftarrow \mathcal{E}^+(z,\alpha)\,\right] = 1$$

where the probability is taken over the coins used by P. We will sometimes restrict attention to honest multi-point extractors. Note that this is not, in general, without loss, since $\mathcal{E}$ does not have oracle access to P. However it will be easy to verify that extractors we construct are honest.

We can simplify some of our proofs with the following easy, but useful, results. First, WPrA when allowing only a single query (denoted 1-WPrA) implies WPrA with many queries. The proof (omitted) is by straightforward hybrid argument. Second, and more interestingly, we give a lemma showing that any function that is both CR and WPrA is also PrA. These lemmas greatly simplify some of proofs, because together they reduce the task of showing a CR function fully preimage-aware to showing that it meets the WPrA definition for a single extraction query.

Lemma 3.3 [1-WPrA $\Rightarrow$ WPrA] Let P be an ideal primitive and $H^P : Dom \rightarrow Rng$ be a hash function. Let $\mathcal{E}^+$ be a multi-point extractor. Let A be an adversary making at most q_e extraction queries and running in time t. Then there exists an PrA adversary B, asking at most one extraction query, such that

$$\mathbf{Adv}^{\mathrm{wpra}}_{H,P,\,\mathcal{E}^+}(A) \leq q_e \cdot \mathbf{Adv}^{\mathrm{wpra}}_{H,P,\,\mathcal{E}^+}(B)\ .$$

B runs in time at most $t + \mathcal{O}(q_e \cdot \mathsf{Time}(\mathcal{E}^+))$ and makes the same number of P queries queries as A. $\square$

Lemma 3.4 [WPrA + CR $\Rightarrow$ PrA] Let P be an ideal primitive and $H^P : Dom \rightarrow Rng$ be a hash function. Let $\mathcal{E}^+$ be an arbitrary honest multi-point extractor. Then there exists an extractor $\mathcal{E}$ such that for any pra-adversary A making q_e extraction queries there exists wpra-adversary B and cr-adversary C such that

$$\mathbf{Adv}^{\mathrm{pra}}_{H,P,\,\mathcal{E}}(A) \leq \mathbf{Adv}^{\mathrm{wpra}}_{H,P,\,\mathcal{E}^+}(B) + \mathbf{Adv}^{\mathrm{cr}}_{H,P}(C)\ .$$

B makes the same number of queries as A and runs in time that of A plus $\mathcal{O}(q_e)$. C asks q_p queries and runs in time $t + q_e \cdot \mathsf{Time}(\mathcal{E}^+)$. $\mathcal{E}$ runs in the same time as $\mathcal{E}^+$. $\square$

Proof: Let $\mathcal{E}$ be the extractor that, on input (z, α) runs $\mathcal{X} \leftarrow \mathcal{E}^+(z, \alpha)$ and outputs the first element in $\mathcal{X}$. Let B be a WPA-adversary that works as follows. It runs A, just forwarding oracle queries to P and $\mathcal{E}^+$, returning P-responses directly to A and simulating $\mathcal{E}$ using the responses of $\mathcal{E}^+$. Let B output whatever A does.

In the event space defined by $\mathbf{Exp}^{\mathrm{pra}}_{H,P,\,\mathcal{E},A}$ let Coll denote the event that $\mathcal{E}^+$ outputs a set of size larger than one. Then its clear that $\Pr[\mathsf{Coll}] \leq \mathbf{Adv}^{\mathrm{cr}}_{H}(C)$ for the natural adversary C because $\mathcal{E}^+$ is honest. Note that until event Coll occurs the execution of $\mathbf{Exp}^{\mathrm{pra}}_{H,P,\,\mathcal{E},A}$ is identical that of $\mathbf{Exp}^{\mathrm{pra}}_{H,P,\,\mathcal{E}^+,B}$. Therefore,

$$\Pr\left[\, \mathbf{Exp}^{\mathrm{pra}}_{H,P,\,\mathcal{E},A} \Rightarrow \mathsf{true} \,\right] \leq \Pr\left[\, \mathbf{Exp}^{\mathrm{pra}}_{H,P,\,\mathcal{E}^+,B} \Rightarrow \mathsf{true} \,\right] + \Pr\left[\, \mathsf{Coll} \,\right]$$

implying the theorem statement. ∎

4 Merkle-Damgård as an FIL-RO domain extender

In this section we develop a main result: that an MD-hash is a good domain extender for an FIL random oracle. We do this in two steps. First, in Theorem 4.1 we prove a generic result that *any* PrA function is a good domain extender for an FIL random oracle. This is interesting in itself, because until now no property weaker than being a PRO is known to be sufficient for extending the domain of an FIL-RO; in particular, CR is not sufficient [17]. In the second step, Theorem 4.2, we prove that Merkle-Damgård (with strengthening) yields a VIL-PrA hash function when the underlying compression function is a FIL-PrA function.

Theorem 4.1 [RO domain extension via PrA] Let P be an ideal primitive and $H^P : Dom \rightarrow Rng$ be a hash function. Let R be an ideal primitive with two interfaces that implements independent functionalities P and $\mathcal{R} = \mathsf{RF}_{Rng,Rng}$. Define $F^R(M) = \mathcal{R}(H^P(M))$. Let $\mathcal{F} = \mathsf{RF}_{Dom,Rng}$. 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

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

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}. $\square$

Proof: Let $\mathcal{E}$ be an arbitrary extractor for H. Then $\mathcal{S} = (\mathcal{S}_1, \mathcal{S}_2)$ works as follows. It maintains an internal advice string α (initially empty) that will consist of pairs (u, v) corresponding to A's queries to P (via $\mathcal{S}_1$). When A queries u to $\mathcal{S}_1$ the simulator simulates $u \leftarrow P(v)$ appropriately, sets $\alpha \leftarrow \alpha \parallel (u, v)$, and returns v. For a query Y to $\mathcal{S}_2$, the simulator runs

$X \leftarrow \mathcal{E}(Y,\alpha)$. If $X = \perp$ then the simulator returns a random point. Otherwise it queries $Z \leftarrow \mathcal{F}(X)$ and returns Z to the adversary.

Consider the experiments of the PRO definition, that is A interacting with oracle interfaces $(\mathcal{O}_0, \mathcal{O}_1, \mathcal{O}_2) = (F, P, \mathcal{R})$ or $(\mathcal{O}_0, \mathcal{O}_1, \mathcal{O}_2) = (\mathcal{F}, \mathcal{S}_1, \mathcal{S}_2)$. Informally, there exist only a few events that adversary A can cause that force the two sets of oracles to behave differently. These are:

(i) if A makes a query $\mathcal{O}_2(Y)$ and the extractor outputs $\perp$ when run on $\mathcal{E}(Y,\alpha)$. Later A queries $\mathcal{O}_0(X)$ such that $H^P(X) = Y$;
(ii) if A first makes a query $\mathcal{O}_0(X)$ with $Y = H^P(X)$ and later A queries $\mathcal{O}_2(Y)$, yet $\mathcal{E}(Y,\alpha)$ outputs a value that does not equal X;
(iii) if A first makes a query $\mathcal{O}_2(Y)$, the extractor outputs a point X, and later A queries $\mathcal{O}_0(X')$ such that $X \neq X'$ but $H^P(X') = Y$; or
(iv) if A queries $\mathcal{O}_0(X)$ and $\mathcal{O}_0(X')$ such that $X \neq X'$ but $H^P(X) = H^P(X')$.

Furthermore, each case implies that A has forced a situation that leads to contradicting the preimage awareness of H.

To formalize these observations, we utilize the five games and adversary B shown in Figure 3 and Figure 4. We will justify that

$$
\begin{aligned}
\mathbf{Adv}^{\mathrm{pro}}_{F,\mathcal{S}}(A) &= \Pr\left[\, A^{\mathsf{R0}} \Rightarrow 1 \,\right] - \Pr\left[\, A^{\mathrm{I0}} \Rightarrow 1 \,\right] & (1)\\
&= \Pr\left[\, A^{\mathsf{R1}} \Rightarrow 1 \,\right] - \Pr\left[\, A^{\mathrm{I0}} \Rightarrow 1 \,\right] & (2)\\
&= \Pr\left[\, A^{\mathrm{G0}} \Rightarrow 1 \,\right] - \Pr\left[\, A^{\mathrm{G1}} \Rightarrow 1 \,\right] & (3)\\
&\leq \Pr\left[\, A^{\mathrm{G1}} \text{ sets } \mathsf{bad} \,\right] & (4)\\
&= \Pr\left[\, A^{\mathrm{G2}} \text{ sets } \mathsf{bad} \,\right] & (5)\\
&= \mathbf{Adv}^{\mathrm{pra}}_{H,P,\mathcal{E}}(B)\ . & (6)
\end{aligned}
$$

Game R0 implements the oracles $(F, P, \mathcal{R})$ using a table $\mathtt{R}$ to simulate the random oracle $\mathcal{R}$. Note that the simulation is such that none of the oracle procedures query each other (unlike the obvious implementation of $F, P, \mathcal{R}$). Included in R0 is some extra "book-keeping" code, such as a table $\mathtt{YtoX}$ tracking mappings between range points under H and preimages, usage of the extractor in handling $\mathcal{O}_2$ queries, etc. This extra code will be useful in future games, but in R0 it does not affect the computation of responses to queries by A. Thus, $\Pr[\mathbf{Exp}^{\text{indiff-1}}_{F,R,A} \Rightarrow 1] = \Pr[A^{\mathsf{R0}} \Rightarrow 1]$.

Game R1 implements the same functionality as R0, but using a different way of simulating $\mathcal{R}$. Specifically, the table $\mathtt{R}$ is relegated to only handle points Y that are queried to $\mathcal{O}_2$ and for which the extractor $\mathcal{E}$ outputs bottom. These table entries are only later potentialy used in the handling of a $\mathcal{O}_0$ query. (Note that repeat queries to $\mathcal{O}_2$ are pointless, and therefore disallowed.) Other points are handled by a new table $\mathtt{F}$ in exactly the same manner as done before in game R0. Thus $\Pr[A^{\mathsf{R0}} \Rightarrow 1] = \Pr[A^{\mathsf{R1}} \Rightarrow 1]$.

Game G0 (boxed statements included) is the same as R1 except that we now index $\mathtt{F}$ with preimages of H associated to points Y. To ensure consistency of the simulation with R1, the table $\mathtt{YtoX}$ is used to keep track of what X, Y pairs have been seen thus far. If ever there are are two values X associated to a single Y value, then $\mathtt{F}$ is indexed via the first encountered X used. This provides behavior consistent with game R1. Thus, $\Pr[A^{\mathsf{R1}} \Rightarrow 1] = \Pr[A^{\mathrm{G0}} \Rightarrow 1]$

Game I1 implements the oracles $(\mathcal{F}, \mathcal{S}_1, \mathcal{S}_2)$ using a table $\mathtt{F}$ to simulate the random oracle $\mathcal{F}$. Note that the simulation is such that none of the oracle procedures query each other (unlike the obvious implementation of $\mathcal{F}, \mathcal{S}_1, \mathcal{S}_2$). Like in R0 there is extra book-keeping code that does not affect responses. Thus, $\Pr[\mathbf{Exp}^{\text{indiff-0}}_{\mathcal{F},\mathcal{S},A} \Rightarrow 1] = \Pr[A^{\mathrm{I0}} \Rightarrow 1]$.

Game G1 (boxed statements excluded) is the same as I1 except for the setting of bad and replacing Z with $\mathtt{R}[Y]$ in $\mathcal{O}_2$. Both do not affect query responses because the boxed statements are omitted. Thus, $\Pr[A^{\mathrm{I0}} \Rightarrow 1] = \Pr[A^{\mathrm{G1}} \Rightarrow 1]$. We have so far justified Equations (1), (2), and (3). Since G0 and G1 are identical-until-bad, we can apply the fundamental lemma of game-playing [2] to justify Equation (4).

In game G1 the setting of bad does not affect query responses. We therefore defer the setting of bad until **Finalize** in game G2. To do so, we record a transcript τ of information related to $\mathcal{O}_0$ and $\mathcal{O}_2$ queries. We also replace the setting of $\mathtt{R}$ in $\mathcal{O}_2$ with the setting of a variable Z. Each entry records which oracle was queried along with a domain point X (or $\perp$) and a range point Y (in $\mathcal{O}_0$ this is set to $\perp$ because we will wait until **Finalize** to compute it). Procedure **Finalize** iterates over the resultant transcript, computing the missing Y values, filling out the tables $\mathtt{YtoX}$ and $\mathtt{R}$, and determining if bad is set.

procedure $\mathcal{O}_0(X)$: Game R0

$Y \leftarrow H^P(X)$
$\mathtt{YtoX}[Y] \leftarrow X$
If $\mathtt{R}[Y] \neq \bot$ then Ret $\mathtt{R}[Y]$
Ret $\mathtt{R}[Y] \leftarrow\$ \mathit{Rng}$

procedure $\mathcal{O}_1(u)$:

$v \leftarrow P(u)$; $\alpha \leftarrow \alpha \,\|\, (u, v)$; Ret v

procedure $\mathcal{O}_2(Y)$:

$X \leftarrow \mathcal{E}(Y, \alpha)$
$\mathtt{YtoX}[Y] \leftarrow X$
If $\mathtt{R}[Y] \neq \bot$ then Ret $\mathtt{R}[Y]$
Ret $\mathtt{R}[Y] \leftarrow\$ \mathit{Rng}$

procedure $\mathcal{O}_0(X)$: Game R1

$Y \leftarrow H^P(X)$
If $\mathtt{R}[Y] \neq \bot$ then Ret $\mathtt{R}[Y]$
$\mathtt{YtoX}[Y] \leftarrow X$
If $\mathtt{F}[Y] \neq \bot$ then Ret $\mathtt{F}[Y]$
Ret $\mathtt{F}[Y] \leftarrow\$ \mathit{Rng}$

procedure $\mathcal{O}_1(u)$:

$v \leftarrow P(u)$; $\alpha \leftarrow \alpha \,\|\, (u, v)$; Ret v

procedure $\mathcal{O}_2(Y)$:

$X \leftarrow \mathcal{E}(Y, \alpha)$
If $X = \bot$ then Ret $\mathtt{R}[Y] \leftarrow\$ \mathit{Rng}$
$\mathtt{YtoX}[Y] \leftarrow X$
If $\mathtt{F}[Y] \neq \bot$ then Ret $\mathtt{F}[Y]$
Ret $\mathtt{F}[Y] \leftarrow\$ \mathit{Rng}$

procedure $\mathcal{O}_0(X)$: Game I1

$Y \leftarrow H^P(X)$
$\mathtt{YtoX}[Y] \leftarrow X$
If $\mathtt{F}[X] \neq \bot$ then Ret $\mathtt{F}[X]$
Ret $\mathtt{F}[X] \leftarrow\$ \mathit{Rng}$

procedure $\mathcal{O}_1(u)$:

$v \leftarrow P(u)$; $\alpha \leftarrow \alpha \,\|\, (u, v)$; Ret v

procedure $\mathcal{O}_2(Y)$:

$X \leftarrow \mathcal{E}(Y, \alpha)$
$\mathtt{YtoX}[Y] \leftarrow X$
If $X = \bot$ then Ret $Z \leftarrow\$ \mathit{Rng}$
If $\mathtt{F}[X] \neq \bot$ then Ret $\mathtt{F}[X]$
Ret $\mathtt{F}[X] \leftarrow\$ \mathit{Rng}$

procedure $\mathcal{O}_0(X)$: Games $\boxed{\text{G0}}$ G1

$Y \leftarrow H^P(X)$
If $\mathtt{R}[Y] \neq \bot$ then bad $\leftarrow$ true $\boxed{\text{; Ret } \mathtt{R}[Y]}$
If $\mathtt{YtoX}[Y] \neq \bot \wedge \mathtt{YtoX}[Y] \neq X$ then
  bad $\leftarrow$ true $\boxed{\text{; Ret } \mathtt{F}[\mathtt{YtoX}[Y]]}$
$\mathtt{YtoX}[Y] \leftarrow X$
If $\mathtt{F}[X] \neq \bot$ then Ret $\mathtt{F}[X]$
Ret $\mathtt{F}[X] \leftarrow\$ \mathit{Rng}$

procedure $\mathcal{O}_1(u)$:

$v \leftarrow P(u)$; $\alpha \leftarrow \alpha \,\|\, (u, v)$; Ret v

procedure $\mathcal{O}_2(Y)$:

$X \leftarrow \mathcal{E}(Y, \alpha)$
If $X = \bot$ then Ret $\mathtt{R}[Y] \leftarrow\$ \mathit{Rng}$
If $\mathtt{YtoX}[Y] \neq \bot \wedge \mathtt{YtoX}[Y] \neq X$ then
  bad $\leftarrow$ true $\boxed{\text{; Ret } \mathtt{F}[\mathtt{YtoX}[Y]]}$
$\mathtt{YtoX}[Y] \leftarrow X$
If $\mathtt{F}[X] \neq \bot$ then Ret $\mathtt{F}[X]$
Ret $\mathtt{F}[X] \leftarrow\$ \mathit{Rng}$

Figure 3: The five games R0, R1, I1, G0, and G1 used in the proof of Theorem 4.1. In each game, initially all tables are everywhere $\bot$.

procedure $\mathcal{O}_0(X)$: Game G2

$i \leftarrow i+1$; $\tau \leftarrow \tau \,\|\, (0, X, \bot)$
If $\mathtt{F}[X] \neq \bot$ then Ret $\mathtt{F}[X]$
Ret $\mathtt{F}[X] \leftarrow\!\!\$\; Rng$

procedure $\mathcal{O}_1(u)$:

$v \leftarrow P(u)$; $\alpha \leftarrow \alpha \,\|\, (u, v)$; Ret v

procedure $\mathcal{O}_2(Y)$:

$X \leftarrow \mathcal{E}(Y, \alpha)$
$i \leftarrow i+1$; $\tau \leftarrow \tau \,\|\, (2, X, Y)$
If $X = \bot$ then Ret $Z \leftarrow\!\!\$\; Rng$
If $\mathtt{F}[X] \neq \bot$ then Ret $\mathtt{F}[X]$
Ret $\mathtt{F}[X] \leftarrow\!\!\$\; Rng$

procedure Finalize(b):

$(\gamma_1, X_1, Y_1), \ldots, (\gamma_i, X_i, Y_i) \leftarrow \tau$
For $j = 1$ to i do
    If $Y_j = \bot$ then $Y_j \leftarrow H^P(X_j)$
    If $\gamma_j = 0 \wedge \mathtt{R}[Y_j] \neq \bot$ then $\mathsf{bad} \leftarrow \mathsf{true}$
    If $\gamma_j = 2 \wedge X_j = \bot$ then $\mathtt{R}[Y_j] \leftarrow 1$
    Else
        If $\mathtt{YtoX}[Y_j] \neq \bot \wedge \mathtt{YtoX}[Y_j] \neq X_j$ then
            $\mathsf{bad} \leftarrow \mathsf{true}$
        $\mathtt{YtoX}[Y_j] \leftarrow X_j$

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

Run $A^{\mathcal{O}_0,\mathcal{O}_2,\mathcal{O}_3}$, answering queries by:

query $\mathcal{O}_0(u)$:
    $i \leftarrow i+1$; $\tau \leftarrow \tau \,\|\, (0, X, \bot)$
    If $\mathtt{F}[X] \neq \bot$ then Ret $\mathtt{F}[X]$
    Ret $\mathtt{F}[X] \leftarrow\!\!\$\; Rng$

query $\mathcal{O}_1(u)$:
    Ret $\mathsf{P}(u)$

query $\mathcal{O}_2(Y)$:
    $X \leftarrow \mathsf{Ex}(Y)$
    $i \leftarrow i+1$; $\tau \leftarrow \tau \,\|\, (2, X, Y)$
    If $X = \bot$ then Ret $Z \leftarrow\!\!\$\; Rng$
    If $\mathtt{F}[X] \neq \bot$ then Ret $\mathtt{F}[X]$
    Ret $\mathtt{F}[X] \leftarrow\!\!\$\; Rng$

When A halts with output b:
    $(\gamma_1, X_1, Y_1), \ldots, (\gamma_i, X_i, Y_i) \leftarrow \tau$
    For $j = 1$ to i do
        If $Y_j = \bot$ then $Y_j \leftarrow H^{\mathsf{P}}(X_j)$
        If $\gamma_j = 0 \wedge \mathtt{R}[Y_j] \neq \bot$ then Output X_j
        If $\gamma_j = 2 \wedge X_j = \bot$ then $\mathtt{R}[Y_j] \leftarrow 1$
        Else
            If $\mathtt{YtoX}[Y_j] \neq \bot \wedge \mathtt{YtoX}[Y_j] \neq X_j$ then
                $k \leftarrow \Gamma[Y_j]$
                If $\gamma_j = 2$ then Output X_k
                If $\gamma_k = 2$ then Output X_j
                $X^* \leftarrow \mathsf{Ex}(Y_j)$
                If $X^* = X_k$ then Output X_j
                If $X^* = X_j$ then Output X_k
            $\mathtt{YtoX}[Y_j] \leftarrow X_j$
        $\Gamma[Y_j] \leftarrow j$

Figure 4: The game G2 and adversary B used in the proof of Theorem 4.1. Initially $i = 0$ and all tables are everywhere $\bot$.

Procedure **Finalize** is written so that $\Pr\left[\,A^{\mathrm{G1}} \text{ sets bad }\right] = \Pr\left[\,A^{\mathrm{G2}} \text{ sets bad }\right]$, justifying (5).

We bound the probability of bad being set in game G2 by building a PrA adversary B against H. The adversary B is detailed in Figure 4. It executes A^{G2}, except that the ideal primitive P is replaced by queries to B's primitive oracle P, usage of $\mathcal{E}$ is replaced by queries to B's extraction oracle Ex, and setting bad is replaced by outputing a domain point of H. Note that since all extraction queries are finished by the time **Finalize** is executed, calculating the missing Y values does not affect the advice strings used by the extractor. We will now finish by justifying Equation (6).

By construction setting bad in G2 corresponds to B outputting a non-bottom domain point. Thus we must justify that any time B outputs a domain point, it wins in the PrA game. We do so via a case analysis:

- Line 33: This case corresponds to event (i) discussed above. The conditional on line 33 ensures that there exists a previous transcript entry (γ_k, X_k, Y_k) with $k < j$ such that $\gamma_k = 2$ and $X_k = \bot$. (This is because $\mathtt{R}[Y_j]$ is defined.) But this implies that Ex was queried on $Y_k = Y_j$ with return $\bot$ and that $H^P(X_j) = Y_j = Y_k$. Thus outputting X_j causes B to win the PrA game.
- Line 38: This case corresponds to event (ii) discussed above. The conditional on line 36 implies that there exists a $k < j$ such that transcript entry (γ_k, X_k, Y_k) was such that $Y_k = Y_j$ but $X_k \neq X_j$. Since $\gamma_j = 2$, this means X_k was first queried to $\mathcal{O}_0$ and then $Y_j = Y_k$ was later queried to $\mathcal{O}_2$, but the query to $\mathsf{Ex}(Y_k)$ did not output X_k. Thus outputting X_k causes B to win the PrA game.
- Line 39: This case corresponds to event (iii) discussed above. This case is exactly like the last: the first query was made to $\mathcal{O}_2$ on Y_k, the query $\mathsf{Ex}(Y_k)$ output a value X_k, and a later query X_j to $\mathcal{O}_0$ was such that $H^P(X_j) = Y_j = Y_k$ yet $X_j \neq X_k$. Thus outputting X_j causes B to win the PrA game.
- Lines 41,42: These cases corresponds to event (iv) discussed above. Two queries X_k, X_j to $\mathcal{O}_0$ resulted in $H^P(X_j) = Y_j = Y_k = H^P(X_k)$. The adversary queries Y_j to the extraction oracle and outputs whichever of X_k, X_j is not returned. This results in B winning the PrA game.

In all cases we see that B succeeds, justifying (2). ▮

Theorem 4.1 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 $\mathsf{unpad}\colon (\{0,1\}^d)^+ \to \{0,1\}^* \cup \{\bot\}$ 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 $\bot$ by unpad.

Theorem 4.2 [SMD is PrA-preserving] Fix $n, d > 0$ and let P be an ideal primitive. Let $h^P\colon \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^n$ 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 $\ell_{max} \geq 1$ blocks there exists an adversary B such that

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

$\mathcal{E}_H$ 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 $\ell_{max} \cdot \mathsf{NumQueries}(h, \ell_{max}) + q_p$ ideal primitive queries, 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.

<u>**adversary** $B^{\mathsf{P},\mathsf{Ex}}(\varepsilon)$:</u>

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

<u>**subroutine** $\mathsf{SimEx}(z, \alpha)$:</u>

$i \leftarrow 1\ ;\ v_1 \leftarrow z$
While $i \leq \ell_{max}$ do
  $(v_{i+1}, m_i) \leftarrow \mathsf{Ex}(v_i, \alpha)$
  $\mathtt{Q}[v_i] \leftarrow 1\ ;\ \mathtt{E}[v_i] \leftarrow (v_{i+1}, m_i)$
  If $v_{i+1} = \bot$ then Ret $\bot$
  $m \leftarrow \mathsf{unpad}(m_i \cdots m_1)$
  If $v_{i+1} = IV$ and $m \neq \bot$ then Ret m
  $i \leftarrow i + 1$
Ret $\bot$

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). For the line of code $(v_{i+1}, m_i) \leftarrow \mathsf{Ex}(v_i, \alpha)$ is executed with the oracle returning $\perp$, then both v_{i+1} and m_i are assigned $\perp$. The code $m^*_\ell \cdots m^*_1 \xleftarrow{d} \mathsf{sfpad}(m^*)$ means take the output of $\mathsf{sfpad}(m^*)$ and parse it into ℓ d-bit blocks $m^*_\ell, \ldots, m^*_1$. 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. $m^*_\ell \cdots m^*_1$) 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 $\mathbf{Exp}^{\mathrm{pra}}_{H,P,\mathcal{E}_H,A}$. Thus we have that $\mathbf{Adv}^{\mathrm{pra}}_{H,P,\mathcal{E}_H}(A) = \Pr[m^* \text{ satisfies}]$ where the event "m^* satisfies", defined over the experiment $\mathbf{Exp}^{\mathrm{pra}}_{h,P,\mathcal{E}_B,B}$, occurs when the message m^* satisfies the conditions of winning for A. Namely that $H^P(m^*)$ was queried to SimEx and the reply given was not equal to m^*. We call m^* a satisfying preimage for A. We will show that whenever m^* is a satisfying preimage for A, with $m^*_\ell \cdots m^*_1 \xleftarrow{d} \mathsf{sfpad}(m^*)$, there exists a k with $1 \le k \le \ell$ for which adversary B returns (v^*_{k+1}, m^*_k) and this pair is a satisfying preimage for B (i.e. one that wins the PrA experiment against h for B). This will establish that

$$\Pr[\, m^* \text{ satisfies} \,] \le \mathbf{Adv}^{\mathrm{pra}}_{h,\mathcal{E}_h}(B) \; . \tag{7}$$

Consider the query $\mathsf{SimEx}(H^P(m^*))$ necessarily made by A. Let $(v_{j+1}, x_j), \ldots, (v_2, x_1)$ be the sequence of values returned by the Ex queries made by SimEx in the course of responding to A's query. Necessarily $1 \le j \le \ell_{max}$ and $1 \le \ell \le \ell_{max}$.

We will show that there exists a k such that $1 \le k \le \min\{j, \ell\}$ and $(v_{k+1}, x_k) \ne (v^*_{k+1}, m^*_k)$. (This includes the possibility that $v_{k+1} = \perp$ and $x_k = \perp$.) First we use this fact to conclude. Since $k \le j$ it means that v_k was queried to Ex. If $v_k = v^*_k = H^P(v^*_{k+1}, m^*_k)$ we are done, because then v^*_{k+1}, m^*_k is a satisfying preimage for B. Otherwise, $v_k \ne v^*_k$ and we can repeat the reasoning for $k-1$. At $k = 1$ we have that, necessarily, $c_k = v^*_k$ since this was the image queried by A. Thus there must exist a satisfying preimage, justifying (7).

We return to showing the existence of k such that $1 \le k \le \min\{j, \ell\}$ and $(v_{k+1}, x_k) \ne (v^*_{k+1}, m^*_k)$. Assume for contradiction that no such k exists, meaning that $(v^*_{i+1}, m^*_i) = (v_{i+1}, x_i)$ for $1 \le i \le \min\{j, \ell\}$. If $j > \ell$, then since $v_j = IV$ and $m^*_\ell \cdots m^*_1 = m_\ell \cdots m_1$ we have a contradiction because in such a situation the loop in SimEx would have halted at iteration ℓ. If $j = \ell$, then having $m^*_\ell \cdots m^*_1 = m_\ell \cdots m_1$ and $v_{\ell+1} = v^*_{\ell+1} = IV$ would imply that SimEx returned $m = m^*$, contradicting that m^* is a satisfying preimage for A. If $j < \ell$, then the loop in SimEx must have stopped iterating because $v_{j+1} = IV$ (if $v_{j+1} = \perp$ we would already have contradicted our assumption regarding k) and $x \ne \perp$. But by assumption we have that $m^*_j \cdots m^*_1 = m_j \cdots m_1$ and so there exist two strings m and m^* for which $\mathsf{sfpad}(m)$ is a suffix of $\mathsf{sfpad}(m^*)$. 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 iteration of h is a CR function [18, 22]. We prove an analogous theorem for $\mathsf{Itr}[h]$ and preimage awareness in Appendix B. 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 iteration (without strengthening) to provide preimage awareness (analogously, collision-resistance).

5 Building Preimage-Aware Functions

The results of Section 4 allow us to more elegantly and modularly prove that a hash function construction is a pseudorandom oracle (PRO). Particularly, Theorems 4.1 and 4.2 mean that the task of building a PRO is reduced to the 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 commitment scheme of [19] 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 [31, 7, 25, 26, 34, 35, 33, 21] 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 many cases one can generalize these results to showing the constructions are also PrA. In the rest of this section we show that the Davies-Meyer and other so-called "group-1" PGV compression functions [31, 7] are not only CR but PrA. We also give bounds on the PrA-security of the Shrimpton-Stam compression function [35], the Dodis-Pietrzak-Puniya compression function [21], and the first two steps of the MCM construction [32]. Previously these constructions were only known to be CR.

5.1 CR compression functions from PGV are preimage-aware

Let us begin with block-cipher-based compression functions. Sixty-four schemes, including Davies-Meyer and MMO, were considered by Preneel et al. [31], and twelve of these were later proven to be optimally collision-resistant and preimage-resistant (in the ideal-cipher model) by Black et al. [7]; these were labeled as "group-1". Another group of eight "group-2" compression functions were found to be optimally collision resistant in the iteration (and preimage resistant in the iteration up to the birthday bound) despite not being collision or preimage resistant themselves. Subsequently, Stam [37] gave an alternative classification of these schemes based on a more general analysis of rate-1 block-cipher-based compression functions. He considered compression functions that, on input a chaining variable $v \in \{0,1\}^n$ and message block $m \in \{0,1\}^d$, operate as follows:

$$(k, x) \leftarrow C^{\text{PRE}}(v, m)\ ;\ y \leftarrow E(k, x)\ ;\ \text{Ret } w \leftarrow C^{\text{POST}}(v, m, y)$$

where $C^{\text{PRE}}\colon \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^d \times \{0,1\}^n$ and $C^{\text{POST}}\colon \{0,1\}^d \times \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ are functions called preprocessing and postprocessing, respectively. He also defined an auxillary post-processing function $C^{\text{AUX}}\colon \{0,1\}^d \times \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$. Davies-Meyer, for example, has $C^{\text{PRE}}(v, m) = (m, v)$ and $C^{\text{POST}}(v, m, y) = v \oplus y$ and $C^{\text{AUX}}(k, x, y) = x \oplus y$. Stam called a scheme "Type-I" iff

(1) C^{PRE} is bijective,

(2) for all v, m the mapping $C^{\text{POST}}(v, m, \cdot)$ is bijective, and

(3) for all k, y the mapping $C^{\text{AUX}}(k, \cdot, y)$ is bijective.

When they exist, we let $C^{-\text{PRE}}$ denote the inverse of C^{PRE}, $C^{-\text{POST}}(v, m, \cdot)$ denote the inverse of $C^{\text{POST}}(v, m, \cdot)$, and $C^{-\text{AUX}}(k, \cdot, y)$ denote the inverse of $C^{\text{AUX}}(k, \cdot, y)$. As it turns out, the twelve "group-1" compression functions are also "Type-I". We leverage Stam's results here to show that the group-1/Type-I PGV compression functions are preimage aware. In Appendix C we discuss the group-2 PGV functions, showing these build preimage aware hash functions when used within an iteration. (There too we leverage Stam's generalized framework and results.)

Theorem 5.1 [The Group-1/Type-I PGV schemes are PrA] Fix $\kappa, n > 0$, let $\mathcal{C}_{\kappa,n} = (E, D)$ be an ideal cipher and let $H^{\mathcal{C}}$ be a Type-I block-cipher-based compression function. There exists an extractor $\mathcal{E}$ such that for any adversary A making at most q_p queries to $\mathcal{C}$ and q_e extraction queries we have

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

where $\mathcal{E}$ runs in time at most $\mathcal{O}(q_p(\mathsf{Time}(C^{-\text{PRE}}) + \mathsf{Time}(C^{\text{POST}})))$. □

Proof: We will prove that any such compression function is 1-WPrA-secure, and then use Lemmas 3.3 and 3.4 to give the final bound. We note that Theorem 5 of [37] upperbounds the collision-finding advantage of A by $q_p(q_p + 1)/2(2^n - q_p)$, yielding the second term above.

Let us define the extractor $\mathcal{E}$ as follows:

<u>**algorithm** $\mathcal{E}(z, \alpha)$:</u>
$\mathcal{L} \leftarrow \emptyset$
Parse $(k_1, x_1, y_1), \cdots, (k_r, x_r, y_r) \leftarrow \alpha$
For $i = 1$ to r do
  $(v_i, m_i) \leftarrow C^{-\text{PRE}}(k_i, x_i)$
  If $C^{\text{POST}}(v_i, m_i, y_i) = z$ then $\mathcal{L} \xleftarrow{\cup} (v_i, m_i)$
If $\mathcal{L} \neq \emptyset$ then Return $\mathcal{L}$ else Return $\perp$

Note that because C^{PRE} is a bijection, a pair (v_i, m_i) is uniquely determined by a pair (k_i, x_i). Intuitively, $\mathcal{E}$ simply iterates over the query-response triples and searches for ideal-cipher queries (k, x, y) that would produce z under the compression function. Upon finding them, it adds the corresponding (unique) compression function input (v, m) to the preimage list. Thus all preimages of z that can be determined from the query list α are returned, and if A wins the 1-WPrA experiment, it must find find a new preimage of z. Then one can straightforwardly adapt Stam's results (specifically, the proof of [37, Th. 6]) to show that the probability of this occuring is at most $q/(2^n - q)$. ▮

5.2 Shrimpton-Stam compression function is preimage-aware

Next we show that it is possible to build a PrA compression function from *non-compressing* random functions[2]. In particular, we examine the compression function recently designed by Shrimpton and Stam [35]. They proved that their compression function is nearly optimally collision resistant (i.e. to the birthday bound), and we will now show that it is also PrA.

Theorem 5.2 [Shrimpton-Stam is PrA] Fix $n > 3$. Let f_1, f_2, f_3 be three independent random oracles $\mathcal{F}_{n,n}$. Define a compression function $H^{f_1,f_2,f_3}(c, m) = f_3(f_1(m) \oplus f_2(c)) \oplus f_1(m)$. 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 and q_e extraction queries, we have

$$\mathbf{Adv}^{\text{pra}}_{H,f_1,f_2,f_3,\mathcal{E}}(A) \le \frac{(n+3)q_p q_e}{2^n} + \frac{(3n+13)q_e}{2^n} + \frac{q_p^2(1+4n^2)+1}{2^n}$$

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

Before proving the theorem, we first remark that this bound is conservative in several ways as it uses a hybrid argument and several upperbound approximations to facilitate the proof. With additional effort it is possible that, in particular, the third term in the bound can be tightened a bit.

Proof: We proceed by reasoning about the single-extractor-query WPrA experiment, and then applying Lemmas 3.3 and 3.4 and the collision resistance bound from [35] to get the final result. For convenience, we write q everywhere for q_p (this should cause no confusion as we assume only one extraction query).

We will assume that the advice string has the format

$$\alpha = \langle r \rangle, (a_1, u_1), \ldots, (a_r, u_r), \langle s \rangle, (b_1, v_1), \ldots, (b_s, v_s)\langle t \rangle, (x_1, y_1), \ldots, (x_t, y_t)$$

where $u_i = f_1(a_i)$, $v_i = f_2(b_i)$ and $y_i = f_3(x_i)$, with i over the indicated indices in each case. We now define the (honest) multi-point extractor $\mathcal{E}^+$. Let $\mathcal{L}_1$ and $\mathcal{L}_2$ be initialized to the empty set.

<u>**algorithm** $\mathcal{E}^+(z, \alpha)$:</u>

Parse α as $\langle r \rangle, (a_1, u_1), \ldots, (a_r, u_r), \langle s \rangle, (b_1, v_1), \ldots, (b_s, v_s), \langle t \rangle, (x_1, y_1), \ldots, (x_t, y_t)$
For $i = 1$ to t do
  Let $u'_i = z \oplus y_i$
  If $\exists j$ such that $u'_i = u_j$ then $\mathcal{L}_1 \leftarrow \mathcal{L}_1 \cup \{(x_i, u_j)\}$
For all $(x_i, u_j) \in \mathcal{L}_1$
  if $\exists k$ such that $x_i \oplus u_j = v_k$ then $\mathcal{L}_2 \leftarrow \mathcal{L}_2 \cup \{(a_j, b_k)\}$
if $\mathcal{L}_2 = \emptyset$ then return $\perp$ else return $\mathcal{L}_2$

We note that by construction the extractor returns $\mathcal{L}_2$ (i.e. not $\perp$) only when all $(a, b) \in \mathcal{L}_2$ are preimages of z; thus the extractor is honest. Moreover, this extractor returns all preimages of z that can be computed from the advice string. For A to win the WPrA experiment, it must find a new (possibly first) preimage for z. Without loss, we assume that A outputs a new preimage as soon as one can be computed. Thus there are three cases to consider, namely that A's winning (final) query is to f_1, f_2 or f_3.

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

f_1 query. Consider the case that the winning query is to f_1, and let u^* be the returned value for that query. Since this is the last query, we can assume that all f_2 and f_3 queries have already been made. Let V be the list of f_2 responses. Let $F3$ be the list of f_3 query-pairs (x_i, y_i), and for convenience let X and Y be the lists of x_i and y_i, respectively (order maintained relative to the order in $F3$). If u^* is one of the $q-1$ previously returned f_1 values, then we can assume without loss that A has found a new preimage of z. (This follows because this is assumed to be the final and winning query.) But this happens with probability at most $(q-1)/2^n$. So we proceed under the assumption that u^* does not collide with a previous f_1 response. Consider the multiset $u^* \oplus Y = \{u^* \oplus y_i \colon y_i \in Y\}$. For u^* to be winning for A, the target z must appear in this multiset, and this will happen for any $y_i = z \oplus u^*$. For any fixed $c \in \{0,1\}^n$, the probability that there are at least n strings $y \in Y$ (which are q uniform, independent strings) such that $y = z \oplus c$ is at most 2^{-n}. This follows from a Chernoff bound under the assumption that $q \leq n2^n/6$, which is at least $2^{n/2}$ for all n claimed by the theorem. So under the assumption that $q \leq 2^{n/2}$, we can assume that there are at most n pairs (x_i, y_i) such that $y_i = z \oplus u^*$, this holding except with probability at most 2^{-n}. (The upperbound on q will ultimately be dropped as the collision-resistance bound, and hence the PrA bound, will be vacuous for $q > 2^{n/2}$.)

Now, for each $x_i \in X$, define a set $S_i = \{x_i \oplus v \colon v \in V\}$. (These were already determined at the time of the final f_1 query.) Let $S = \bigcup_{i \in I} S_i$ where I is the set of indices i such that $y_i \oplus u^* = z$. Clearly $|S| \leq nq$, so $u^* \in S$ with probability at most $nq/2^n$. Putting it all together, the probabilty that A wins with an f_1 query is at most $\frac{q-1}{2^n} + \frac{nq}{2^n} + \frac{1}{2^n} \leq \frac{(n+1)q}{2^n} + \frac{1}{2^n}$.

f_2 query. Consider the case that the winning query is to f_2, and let v^* be the returned value for that query. Again, since this is the last query, we can assume that all f_1 and f_3 queries have already been made. Let $F3$, X and Y be as in the previous case. As before, if v^* collides with a previously returned f_2 query, then we can assume that the adversary has found a new preimage of z; this happens with probability at most $(q-1)/2^n$. Now, consider the q distinct values of $v^* \oplus X = \{v^* \oplus x_i \colon x_i \in X\}$. (These are distinct because the x_i are.) Since v^* is winning only if some previously defined f_1-response u is in $v^* \oplus X$, we will assume that for each of the q strings in $v^* \oplus X$ there is such a u; thus, by a union bound, we want to upperbound $\sum_{i=1}^{q} \Pr(v^* \oplus x_i = y_i \oplus z)$. But each probability in the sum is over the uniform v^*, so this sum is at most $q/2^n$. In total then, the probability that A wins with an f_2 query is at most $\frac{q-1}{2^n} + \frac{q}{2^n} \leq \frac{2q}{2^n}$.

f_3 query. Consider the case that the winning query is to f_3, and let y^* be the returned value for that query. As before, we can assume that all f_1 and f_2 queries have been made, and we let U and V be the corresponding multisets of responses. Let N_c be the number of times that $c \in \{0,1\}^n$ appears in the multiset $U \oplus V = \{u \oplus v \colon u \in U, v \in V\}$. We assume for the moment that, for all c, $N_c \leq 3n$ except with some (small) probability that we will bound in a moment. Then the final f_3 query yields at most $3n$ opportunities for $y^* \oplus u = z$ to hold, so that y^* is winning with probability at most $3n/2^n$.

It remains to bound $\Pr[N_c > 3n]$, which we do by bounding $\Pr[N_c \geq 3n]$. Notice that the elements of U and V are independent of adversarial choices, so this is a strictly combinatorial problem. In addition, that the elements in U are independent of those in V, so the order in which these lists are populated is irrelevant from the point of view of the event $N_c \geq 3n$ for any value of c. So we assume that q elements in U are selected, and then we begin to fill in V. Let us assume that there are no 4-way collisions in U, which holds except with probability at most

$$2^n \binom{q}{4} \left(\frac{1}{2^n}\right)^4 \leq \frac{2^n}{4!} \left(\frac{q}{2^n}\right)^4 < \frac{1}{2^n}$$

where in the final inequality we have assumed that $q \leq 2^{n/2}$. Now, under the assumption that U contains no 4-way collisions, each element assigned to V increases N_c by at most 3 for any $c \in \{0,1\}^n$. Thus if $N_c \geq 3n$ for some c, it must be the case that at least n values of v increase N_c. The probability that some value of v increases N_c (equivalently that $v = u \oplus c$ for some $u \in U$) is at most $q/2^n$. Thus by a union bound

$$\begin{aligned}
\Pr[\exists c \in \{0,1\}^n \text{ such that } N_c \geq 3n \mid \text{ no 4-way collisions in } U] \quad &\leq \quad 2^n \binom{q}{n} \left(\frac{q}{2^n}\right)^n \\
&\leq \quad \frac{2^n}{n!} \left(\frac{q^2}{2^n}\right)^n \\
&\leq \quad \frac{2^n}{n!} \qquad \text{(again assuming } q \leq 2^{n/2}\text{)} \\
&\leq \quad 2 \cdot 1 \cdot \frac{2}{3} \cdot \frac{2}{4} \cdots \frac{2}{n} \;\leq\; \frac{4/3}{2^{n-3}} \;<\; \frac{11}{2^n}\,.
\end{aligned}$$

So assuming that $q \leq 2^{n/2}$, the probability that A wins on an f_3 query is at most $\frac{3n}{2^n} + \frac{1}{2^n} + \frac{11}{2^n} = \frac{3n}{2^n} + \frac{12}{2^n}$.

Summary. Pulling together all of our cases, the probability that A manages to find a new preimage for z in our single-extractor-query WPrA experiment is at most

$$\left(\frac{(n+1)q}{2^n} + \frac{1}{2^n}\right) + \frac{2q}{2^n} + \left(\frac{3n}{2^n} + \frac{12}{2^n}\right) = \frac{(n+3)q}{2^n} + \frac{3n+13}{2^n}$$

By Lemma 3.3, we multiply through by q_e to get the bound for the multiple-extractor-query case. Finally, we use Lemma 3.4 and the collision resistance bound from [35]. The latter states that for all $n > 0$ and $k, q \geq 0$ the probability of finding a collision is at most $\frac{q^2}{2^n} + \frac{(kq)^2}{2^n} + P_{q,k,n}$ where

$$\begin{aligned}
P_{q,k,n} &= \left(\frac{q!}{(q-k)!}\right)^2 \left(\frac{2^n}{k!}\right)\left(\frac{(2^n-k)!}{2^n!}\right) \\
&\leq \frac{q^{2k}}{k!}\left(\frac{1}{(2^n-1)(2^n-2)\cdots(2^n-(k-1))}\right) \\
&\leq \frac{q^{2k}}{k!}\left(\frac{1}{2^n-(k-1)}\right)^{k-1}
\end{aligned}$$

Now we determine an upperbound on q (as a function of k, n) such that the last line above is at most $1/2^n$. This means

$$\begin{aligned}
q^{2k} &\leq k!\frac{1}{2^n}(2^n-(k-1))^{k-1} \\
q &\leq (k!)^{\frac{1}{2k}} 2^{\frac{-n}{2k}} (2^n-(k-1))^{\frac{k-1}{2k}}
\end{aligned}$$

Setting $k = 2n$ and taking logarithms (base 2),

$$\log(q) \leq \frac{\log(2n!)}{4n} + \frac{-1}{4} + \frac{2n-1}{4n}\log(2^n-(2n-1))$$

Since $n\log(n) \leq \log(2n!)$ for all n, we can use the more conservative bound

$$\log(q) \leq \frac{\log(n)}{4} + \frac{-1}{4} + \frac{2n-1}{4n}\log(2^n-(2n-1)).$$

Now is can be shown that for $n > 3$

$$\begin{aligned}
\frac{\log(n)}{4} + \frac{-1}{4} + \frac{2n-1}{4n}\log(2^n-(2n-1)) &\geq \frac{2n}{4n}\log(2^{n-1}) \\
&= \frac{n-1}{2}
\end{aligned}$$

Thus if $n > 3$ and $\log(q) \leq (n-1)/2$, or equivalently $q \leq 2^{(n-1)/2}$, we have $P_{q,2n,n} \leq 1/2^n$. This yields a collision bound of $\frac{q^2}{2^n} + \frac{(2nq)^2}{2^n} + \frac{1}{2^n} = \frac{q^2(1+4n^2)+1}{2^n}$. But in fact if $q = 2^{(n-1)/2}$ we can see that $\frac{q^2(1+4n^2)}{2^n} > 1$ (i.e., without the addition of $P_{q,2n,n}$), so we can drop the restriction on q. This completes the proof. ▌

5.3 Dodis-Pietrzak-Puniya compression function is preimage-aware

Dodis et al. [21] 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 [21] shows that this function is PrA for ideal f_1 and f_2.

Theorem 5.3 [DPP is PA] Fix $n > 0$. Let f_1, f_2 be independent random oracles $\mathcal{F}_{n,n}$ and let $H^{f_1,f_2}(c,m) = f_1(c) \oplus f_2(m)$. Then, there exists an extractor $\mathcal{E}$ such that for any adversary A making at most q_1, q_2 queries to f_1, f_2 and makes at most q_e extraction queries, we have

$$\mathbf{Adv}^{\mathrm{pra}}_{H,f_1,f_2,\mathcal{E}}(A) \leq \frac{q_1^2 q_2^2 + 2q_1q_2q_e + q_e}{2^n}\ .$$

$\mathcal{E}$ runs in time $\mathcal{O}(q_1 q_2)$. $\square$

Proof: Using Lemmas 3.3 and 3.4, it suffices to show that H is CR and also WPrA for 1 extraction query. The former bound was already shown in [21]: for any attacker C, $\mathbf{Adv}^{\mathrm{cr}}_{H}(C) \leq q_1^2 q_2^2/2^n$.

Next, we prove security relative to the WPrA notion for one extraction query as per Appendix 3.2. We assume that the advice string α is of the form $\alpha_1\alpha_2$ where α_1 is a list of query/response pairs made by A to f_1 and α_2 is the list for f_2. Then let $\mathcal{E}^+$ be the (honest) multi-point extractor that works as shown below.

algorithm $\mathcal{E}^+(z, \alpha_1\alpha_2)$:
Parse $(c_1, d_1), \cdots, (c_r, d_r) \leftarrow \alpha_1$
Parse $(x_1, y_1), \cdots, (x_p, y_p) \leftarrow \alpha_2$
For $i = 1$ to r do
 For $j = 1$ to p do
  If $d_i \oplus y_j = z$ then $\mathcal{X} \stackrel{\cup}{\leftarrow} (c_i, x_j)$
If $\mathcal{X} = \emptyset$ then Ret $\perp$
Ret $\mathcal{X}$

Now suppose that A did not query at least one of f_1 or f_2 before making the extract query z (otherwise $\mathcal{E}^+$ would extract it). Now, each such new query $c_j = c'$ to f_1 can define at most q_2 new values $z'_i = f_1(c') \oplus f_2(x_i)$. Since $f_1(c')$ is random, the chance that z'_i is equal to z is at most 2^{-n}, so the total probability that $f_1(c')$ would define some value $z'_i = z$ is at most $q_2/2^n$. Symmetrically, the total probability that a new query $x_i = x'$ to f_2 would produce a new value $z'_j = f_1(c_j) \oplus f_2(x')$ equal to z is at most $q_1/2^n$. Taking the union bound over all such new queries, the total probability of obtaining a value $z = f_1(c_j) \oplus f_2(x_i)$ (for some i and j) is at most $2q_1q_2/2^n$. If no such value is found, the chance that the value (c, x) output by the attacker is equal to z is at most 2^{-n}. Combining these bounds, for any attacker B, $\mathbf{Adv}^{\mathrm{wpra\text{-}1}}_{H,f_1,f_2,\mathcal{E}}(B) \leq (2q_1q_2 + 1)/2^n$.

Combining, we get that for any A, $\mathbf{Adv}^{\mathrm{pra}}_{H,f_1,f_2,\mathcal{E}}(A) \leq (q_1^2q_2^2 + 2q_1q_2q_e + q_e)/2^n$. ▮

5.4 Mix-Compress is preimage-aware

We show that the "mix-compress" portion of the "mix-compress-mix" construction from [32] 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 $\mathsf{PreIm}_F(\ell, z) = \{y \mid y \in \{0,1\}^* \wedge |y| = \ell \wedge F(y) = z\}$ for all $\ell > 0$ and $z \in \{0,1\}^n$. That is, $\mathsf{PreIm}_F(\ell, z)$ contains the length ℓ preimages of z under F. We also define the function

$$\delta_F(\ell, z) = \left| \frac{|\mathsf{PreIm}_F(\ell, z)| - 2^{\ell-n}}{2^\ell} \right| \tag{8}$$

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. Second, we let $\mathcal{F}_{*,\tau}$ to be the ideal primitive that, on input $x \in \{0,1\}^*$ returns a randomly chosen string $y \in \{0,1\}^{|x|+\tau}$.

Theorem 5.4 [Mix-Compress is PrA.] Fix $\tau, n > 0$, let $F\colon \{0,1\}^* \to \{0,1\}^n$ and let $\mathcal{F}_{*,\tau}$ be the ideal primitive defined above. Let $H^{\mathcal{F}}(m) = F(\mathcal{F}(m))$ be the hash with minimum accepted message length $\nu \geq n - \tau$ if $n > \tau$ and $\nu \geq \tau$ if $n < \tau$. There exists an extractor $\mathcal{E}$ such that for any pra-adversary A making q_p primitive queries and q_e extraction queries there exists a CR adversary B such that

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

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

The restricted domain in the theorem statement (inherited from [32]) ensures that inputs to F have size at least n bits.

Proof: We prove the WPrA notion for one extraction query and then apply Lemmas 3.3 and 3.4. Let $\mathcal{E}^+$ be the (honest) multi-point extractor that works as shown below.

algorithm $\mathcal{E}^+(z, \alpha)$:
Parse $(x_1, y_1), \cdots, (x_r, y_r) \leftarrow \alpha$
For $i = 1$ to r do
  If $z = F(y_i)$ then $\mathcal{X} \stackrel{\cup}{\leftarrow} x_i$
If $\mathcal{X} = \emptyset$ then Ret $\perp$
Ret $\mathcal{X}$

Assume that before the single extractor query no previous query to $\mathcal{F}$ by the adversary led to a point y in the preimage set (under F) of the challenge point z (otherwise the extractor will have succeeded for z). Then we must bound the probability that a new query to $\mathcal{F}$ results in a point y for which $F(y) = z$. If F for each message length is regular then any new random value y has probability 2^{-n} of being mapped to z under F, and so we could finish with bound $q_p/2^n$. Instead we do something slightly more general using the definitions of PreIm_F, δ_F, and Δ_F defined above. Choose $\ell \in \mathbb{N}$ (such that $\ell - \tau \geq \nu$, the minimum message length of H) and $z \in \{0,1\}^n$ to maximize $|\mathsf{PreIm}_F(\ell, z)|$. Then the optimal strategy for A is to first query z to its extraction oracle and only make queries to $\mathcal{F}$ of length $\ell - \tau$. (Primitive queries before the extraction query will only lower A's advantage, since any successful ones will be known to the extractor.) The result of these queries is q_p random ℓ-bit strings, call these $y_1, \ldots, y_{q_p}$. Then we have that

$$\begin{aligned}
\Pr[\, \exists i \,.\, F(y_i) = z \,] &= \Pr[\, \exists y_i \,.\, y_i \in \mathsf{PreIm}_F(\ell, z) \,] \\
&\leq \sum_{1 \leq i \leq q_p} \Pr[\, y_i \in \mathsf{PreIm}_F(\ell, z) \,] \\
&= \sum_{1 \leq i \leq q_p} \frac{|\mathsf{PreIm}_F(\ell, z)|}{2^\ell} \\
&= q_p \cdot \left(\frac{2^{\ell - n}}{2^\ell} + \delta_F(\ell, z) \right) \qquad (9) \\
&\leq \frac{q_p}{2^n} + q_p \cdot \Delta_F
\end{aligned}$$

where the events are defined in the natural manner. In deriving equality (9) we apply (8) (ignoring the absolute values, since $|\mathsf{PreIm}_F(\ell, z)| \geq 2^{\ell - n}$ due to our maximization). Then applying Lemma 3.3 gives a factor q_e to the right hand side of this bound. We apply Lemma 3.4 and this gives our theorem statement. ▮

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 [16], Fiat-Shamir [24], 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 (and their responses) 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. [39] (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 iterative hash functions.

We next utilize the indifferentiability framework of Maurer et al. [28] 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, but nevertheless enjoys the indifferentiability framework's composibility guarantees [28].

procedure $\mathcal{F}_{eval}(M)$:	procedure $f_{geval}(v, m)$:

If $M \notin \mathit{Dom}$ then Ret $\perp$
If $\mathtt{F}[M] = \perp$ then
  $\mathtt{F}[M] \leftarrow\$ \mathit{Rng}$
  $\mathcal{Q} \xleftarrow{\cup} (M, \mathtt{F}[M])$
Ret $\mathtt{F}[M]$
 	
procedure $\mathcal{F}_{reveal}()$:
Ret $\mathcal{Q}$
	If $v \notin \mathcal{W} \cup \{IV\}$ or $(v, x) \notin \{0,1\}^n \times \{0,1\}^d$ then Ret $\perp$
	If $\mathtt{f}[v, x] = \perp$ then
	  $\mathtt{f}[v, x] \leftarrow\$ \{0,1\}^n$
	$\mathcal{W} \xleftarrow{\cup} \mathtt{f}[v, x]$; $\mathcal{Q} \xleftarrow{\cup} ((v, x), \mathtt{f}[v, x])$
	Ret $\mathtt{f}[v, x]$
	
	procedure $f_{eval}()$:
	If $\mathtt{f}[v, x] = \perp$ then $\mathtt{f}[v, x] \leftarrow\$ \{0,1\}^n$
	$\mathcal{Q} \xleftarrow{\cup} ((v, x), \mathtt{f}[v, x])$
	Ret $\mathtt{f}[v, x]$
	
	procedure $f_{reveal}()$:
	Ret $\mathcal{Q}$

Figure 5: **(Left)** The pub-RO ideal primitive $\mathcal{F}_{Dom,Rng} = (\mathcal{F}_{eval}, (\mathcal{F}_{eval}, \mathcal{F}_{reveal}))$. Initially $\mathtt{F}$ is everywhere $\perp$ and $\mathcal{Q}$ is empty. **(Right)** The pub-GRO ideal primitive $f_{n\times d,n} = (f_{geval}, (f_{eval}, f_{reveal}))$. Initially $\mathtt{f}$ is everywhere $\perp$ and $\mathcal{W}, \mathcal{Q}$ are empty. Here $IV \in \{0,1\}^n$ is a fixed string.

6.1 Public-use ROs and PROs

Fix sets $\mathit{Dom}, \mathit{Rng}$. A public-use random oracle (pub-RO) is an ideal primitive $\mathcal{F}_{Dom,Rng} = ((\mathcal{F}_{eval}), (\mathcal{F}_{eval}, \mathcal{F}_{reveal}))$ defined as follows. Let ρ be a random function $\mathit{Dom} \to \mathit{Rng}$. The (private and public) evaluation interface $\mathcal{F}_{eval}$, on input $M \in \mathit{Dom}$, first adds the pair $(M, \rho(M))$ to an initially-empty set $\mathcal{Q}$ and then returns $\rho(M)$. The (public) reveal interface $\mathcal{F}_{reveal}$ takes no input and returns $\mathcal{Q}$ (suitably encoded into a string). Figure 5 details a pub-RO in code. We say that $\mathcal{F}_{Dom,Rng}$ is a fixed-input-length (FIL) pub-RO if Dom only includes messages of a single length. We write $\mathcal{F}_{n\times d,n}$ for the FIL pub-RO with domain $\mathit{Dom} = \{0,1\}^n \times \{0,1\}^d$ and $\mathit{Rng} = \{0,1\}^n$. As usual, we write just $\mathcal{F}$ when Dom and Rng are clear from context.

INDIFFERENTIABILITY FROM A pub-RO. Let $H^P \colon \mathit{Dom} \to \mathit{Rng}$ be a hash function using an ideal primitive P. Let $\mathcal{F}_{Dom,Rng} = (\mathcal{F}_{eval}, (\mathcal{F}_{eval}, \mathcal{F}_{reveal}))$ be a pub-RO. Let $\mathcal{S}$ be a simulator with oracle access to (both interfaces of) $\mathcal{F}$. Then we associate to pub-pro adversary A, primitive P, and simulator $\mathcal{S}$ the pub-pro advantage function

$$\mathbf{Adv}^{\text{pub-pro}}_{H,P,\mathcal{S}}(A) = \Pr\left[\,\mathbf{Exp}^{\text{indiff-1}}_{H,P,A} \Rightarrow 1\,\right] - \Pr\left[\,\mathbf{Exp}^{\text{indiff-0}}_{\mathcal{F},\mathcal{S},A} \Rightarrow 1\,\right] .$$

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. Informally, we say that a construction H is a pub-PRO if there exists an efficient simulator such that all efficient adversaries A have small advantage.

The composition theorem in [28] (recast to use ITMs in [17]) 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 $\mathcal{F}_{reveal}$.

6.2 Public-use guarded ROs and PROs

Many "structured" compression functions are easily differentiable from a FIL pub-RO. For example, consider the following attack against DM, due to [38]. Let A against $\mathsf{DM}^E(v, m) = E_m(v) \oplus v$ work as follows. It picks a random y and m and then queries its third oracle interface (in the "real" setting this would be E^{-1}) on m, y. 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 v such that $\mathcal{F}_{eval}(v, m) = y \oplus v$. 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. Nevertheless brief reflection suggests that iterating DM from a fixed IV should result in a pub-PRO. We could try to argue this directly, but instead we introduce another variant of ROs as a technical tool to allow modular proofs.

Fix $n, d > 0$ and $IV \in \{0,1\}^n$. A public-use *guarded* random oracle (pub-GRO) is an ideal primitive $f_{n\times d,n} = (f_{geval}, (f_{eval}, f_{reveal}))$ that works as detailed in Figure 5. In words, let ρ be a random function from $\{0,1\}^n \times \{0,1\}^d$ to $\{0,1\}^n$. The (private) guarded evaluation interface f_{geval} on input (v, m) returns $\rho(v, m)$ if $v = IV$ or v is equal to a value previously returned by the interface; it returns $\perp$ otherwise. The (public) evaluation interface f_{eval} returns $\rho(v, m)$.

The (public) reveal interface reveals all so-far (guarded or not) evaluated points and their associated outputs. This weaker version of a FIL pub-RO will still be sufficient for building a pub-PRO using MD. At the same time, the weakening does allow us to show that structured compression functions (such as Davies-Meyers) are indifferentiable from a pub-GRO (even though they are not pub-PROs).

INDIFFERENTIABILITY FROM A pub-GRO. Fix $n, d > 0$ and $IV \in \{0,1\}^n$. Let $h^P : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^n$ be a FIL hash function using ideal primitive P. Let $f_{n \times d,n} = (f_{geval}, (f_{eval}, f_{reveal}))$ be a pub-GRO. Let $\mathcal{S}$ be a simulator with oracle access to (all interfaces of) f. Then we associate to pub-gpro-adversary A, h, P, and $\mathcal{S}$ the pub-gpro advantage function

$$\mathbf{Adv}^{\text{pub-gpro}}_{h,P,\mathcal{S}}(A) = \Pr\left[\mathbf{Exp}^{\text{indiff-1}}_{h,P,A} \Rightarrow 1\right] - \Pr\left[\mathbf{Exp}^{\text{indiff-0}}_{f,\mathcal{S},A} \Rightarrow 1\right] .$$

We stress that in the second probability experiment, while A has access only to f_{geval}, the simulator $\mathcal{S}$ has oracle access to f_{eval} and f_{reveal}. Informally, we say that a construction h is a pub-GPRO if there exists an efficient simulator such that all efficient adversaries A have small advantage.

7 Constructing Public-use Random Oracles

In this section we first show that iterating a FIL public-use RO (or public-use guarded RO) results in an object indifferentiable from a monolithic public-use RO. Then we go on to show that common constructions of compression functions are, in fact, indifferentiable from public-use guarded ROs. In particular we show that the Stam Type-II PGV functions are pub-GROs. Since Davies-Meyers is one such function, these results together imply that the structure of existing hash functions (such as SHA-2) is sound for public-use applications.

7.1 Iteration preserves being a pub-PRO

We show that iteration preserves the property of being a pub-PRO. In fact we show something slightly stronger. Given a compression function that is (indifferentiable from) a public-use guarded RO, iterating this compression function results in a pub-PRO. In the following theorem we write $\mathsf{Itr}[f_{geval}]$ to mean $\mathsf{Itr}[g^{f_{geval}}]$ where g is defined by calling f_{geval} on its input and returning the result. We note that in the computation of $\mathsf{Itr}[g^{f_{geval}}]$ the input to f_{geval} is always either the IV or a valid chaining value.

Theorem 7.1 [Itr is pub-PRO-preserving] Fix $n, d > 0$ and $IV \in \{0,1\}^n$. Let $f_{n\times d,n} = (f_{geval}, (f_{eval}, f_{reveal}))$ be a pub-GRO and let $\mathsf{Itr}[f_{geval}]$ be the iteration of f_{geval}. There exists a simulator $\mathcal{S} = (\mathcal{S}_{eval}, \mathcal{S}_{reveal})$ so that for any adversary A

$$\mathbf{Adv}^{\text{pub-pro}}_{\mathsf{Itr},f,\mathcal{S}}(A) \le \frac{(\sigma q_0 + q_1)^2}{2^n} + \frac{q_1(\sigma q_0 + q_1)}{2^n}$$

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 its $f_{eval}/\mathcal{S}_{eval}$ interface. Let q_2 be the number of queries by A to its $f_{reveal}/\mathcal{S}_{reveal}$ interface. Then $\mathcal{S}$ runs in time that of A plus $\mathcal{O}(q_0\sigma(q_1 + q_2))$ and makes at most $q_0\sigma + 2q_1 + q_2$ queries. $\square$

Proof: We specify a simulator $\mathcal{S} = (\mathcal{S}_{eval}, \mathcal{S}_{reveal})$ in Figure 6. Recall that $\mathcal{S}$ must emulate only the two public interfaces of a public-use guarded RO. To do this, it utilizes a subroutine update() which uses $\mathcal{S}$'s access to $\mathcal{F}_{reveal}()$ to simulate a compression function by defining chaining variables in terms of appropriate outputs of $\mathcal{F}_{eval}$. That is, for any sequence of message m queried to $\mathcal{F}_{geval}$ (or $\mathcal{F}_{eval}$) parsed into message blocks $m_1 \ldots, m_\ell$, the simulator defines compression function input/output pairs $((v_{i-1}, m_i), v_i)$ for $1 \le i \le \ell$ where $v_0 = IV$ and v_i is assigned the output of $\mathcal{F}_{eval}(m_1 \cdots m_i)$. The tables $\mathtt{V}$ and $\mathtt{F}$ are used to maintain this simulation (these tables are initially everywhere $\perp$). Queries to $\mathcal{S}_{eval}$ not associated (via the chaining variable input) with a valid sequence of message blocks have random values returned as output.

Intuitively, $\mathcal{S}$ will succeed as long as no two outputs of $\mathcal{F}_{eval}$ collide (which would mean $\mathcal{S}$ might fail to use the correct sequence of message blocks when assigning a chaining value) or if the adversary queries some value v, m to $\mathcal{S}_{eval}$ for which $\mathtt{V}[v] = \perp$, yet later one of the chaining variables (an output of $\mathcal{F}_{eval}$) is assigned the value v. Loosely, the first event will happen with probability at most $(q_0\sigma{+}q_1)^2/2^n$ while the second event will happen with probability at most $q_1(q_0\sigma{+}q_1)/2^n$. We now give a formal argument using the sequences of games $\mathrm{G}_0 \longrightarrow \cdots \longrightarrow \mathrm{G}_6$ and $\mathrm{G}_2 \longrightarrow \mathrm{G}_{2,1} \longrightarrow \mathrm{G}_{2,2}$. All games include a procedure $\mathcal{O}_2()$ which returns $\mathcal{Q}$ (this was not explicitly included in the pseudocode for brevity). The games

procedure $\mathcal{S}_{eval}(v,m)$:

update()
If $\mathtt{V}[v] \neq \bot$ then
  Ret $\mathcal{F}_{eval}(\mathtt{V}[v] \parallel m)$
$w \leftarrow_{\$} \{0,1\}^n$
$\mathcal{Q} \xleftarrow{\cup} ((v,m),w)$
Ret w

procedure $\mathcal{S}_{reveal}()$:

update()
Ret $\mathcal{Q}$

subroutine update():

$(m^1,v^1),\ldots,(m^p,v^p) \leftarrow \mathcal{F}_{reveal}()$
For $i = p_{\text{last}}+1$ to p do
  $m_1^i,\ldots,m_{\ell_i}^i \xleftarrow{d} m^i$
  $v_0^i \leftarrow IV$; $\mathtt{V}[IV] \leftarrow \varepsilon$
  For $j=1$ to ℓ_i do
    $v_j^i \leftarrow \mathtt{F}[m_1^i \cdots m_j^i]$
    If $v_j^i = \bot$ then
      $v_j^i \leftarrow \mathcal{F}_{eval}(m_1^i \cdots m_j^i)$
      $\mathtt{V}[v_j^i] \leftarrow m_1^i \cdots m_j^i$
      $\mathcal{Q} \xleftarrow{\cup} ((v_{j-1}^i, m_j^i), v_j^i)$
    $\mathtt{F}[m_1^i \cdots m_j^i] \leftarrow v_j^i$
$p_{\text{last}} \leftarrow p$

Figure 6: Simulator used in proof of Theorem 7.1. Tables $\mathtt{V}$ and $\mathtt{F}$ are initialized everywhere to $\bot$, set $\mathcal{Q}$ is initially empty, and p_{last} is initialized to 0.

are shown in Figures 7 and 8. We assume that adversary A does not repeat a query to any of its oracles (such queries are pointless).

THE FIRST GAME SEQUENCE. We start by justifying that

$$
\begin{aligned}
\Pr\left[\,\mathbf{Exp}^{\text{indiff-1}}_{\mathsf{ltr},f,A} \Rightarrow 1\,\right] &= \Pr\left[\,A^{G_0} \Rightarrow 1\,\right] && (10)\\
&= \Pr\left[\,A^{G_1} \Rightarrow 1\,\right] && (11)\\
&\leq \Pr\left[\,A^{G_2} \Rightarrow 1\,\right] + \Pr\left[\,A^{G_2} \text{ sets } \mathsf{bad}\,\right] && (12)\\
&= \Pr\left[\,A^{G_3} \Rightarrow 1\,\right] + \Pr\left[\,A^{G_2} \text{ sets } \mathsf{bad}\,\right] && (13)\\
&\leq \Pr\left[\,A^{G_4} \Rightarrow 1\,\right] + \frac{(q_0\sigma+q_1)^2}{2^{n+1}} + \Pr\left[\,A^{G_2} \text{ sets } \mathsf{bad}\,\right] && (14)\\
&\leq \Pr\left[\,A^{G_5} \Rightarrow 1\,\right] + \frac{(q_0\sigma+q_1)^2}{2^{n+1}} + \Pr\left[\,A^{G_2} \text{ sets } \mathsf{bad}\,\right] && (15)\\
&\leq \Pr\left[\,A^{G_6} \Rightarrow 1\,\right] + \frac{(q_0\sigma+q_1)^2}{2^{n}} + \frac{(q_0\sigma+q_1)^2}{2^{n+1}} + \Pr\left[\,A^{G_2} \text{ sets } \mathsf{bad}\,\right] && (16)\\
&= \Pr\left[\,\mathbf{Exp}^{\text{indiff-0}}_{\mathsf{ltr},\mathcal{S},A} \Rightarrow 1\,\right] + \frac{(q_0\sigma+q_1)^2}{2^{n}} + \Pr\left[\,A^{G_2} \text{ sets } \mathsf{bad}\,\right]\ . && (17)
\end{aligned}
$$

We'll later conclude by bounding $\Pr[A^{G_2} \text{ sets } \mathsf{bad}\,]$ via a sequence of games $G_2 \longrightarrow G_{2,1} \longrightarrow G_{2,2}$.

(Game G_0) By construction, the first game implements exactly the oracles $(\mathsf{ltr}[f_{geval}], f_{eval}, f_{reveal})$, justifying (10). Note that we need not include the explicit checks of the guarded evaluation interface, since the pub-GRO primitive (implemented using Choose-f) is only used by $\mathcal{O}_0$ to implement ltr.

(Game G_1, boxed statement included) In game G_1 we modify the way in which queries are handled via some additional book-keeping; as we will see the changes do not modify the implemented functionality. In particular, we establish a table $\mathtt{V}$ mapping chaining variable values to sequences of message blocks. In $\mathcal{O}_0$ the operation $\mathtt{V}[v_i] \leftarrow m_1 \cdots m_i$ is added. In $\mathcal{O}_1$ handling of a query (v,m) is split into two cases. First, if $\mathtt{V}[v] \neq \bot$, then $\mathtt{V}[w]$ is assigned $\mathtt{V}[v] \parallel m$. Here w is the value to be returned to the adversary as chosen by Choose-f. The assignment of $\mathtt{V}[w]$ maps the new chaining value to a (one-block-longer) sequence of blocks. Second, if $\mathtt{V}[w] = \bot$ then we do no updating of $\mathtt{V}$. There remains one further change in G_1: use of Choose-f in $\mathcal{O}_0$ and in the in $\mathcal{O}_1$ for the first case utilizes $s=0$ while in $\mathcal{O}_1$ for the second case utilizes $s=1$. To account for this, subroutine Choose-f has additional checks to ensure consistency of use between $s=0$ and $s=1$. (Looking ahead, dropping these checks will yield independent random functions for the two values of s. The $s=0$ function will become the monolothic random oracle of $\mathbf{Exp}^{\text{indiff-0}}_{\text{MD},\mathcal{S},A}$ while $s=1$ will be used for the unrelated random values sometimes returned by the simulator.) Because of the checks, we have that the implemented functionality is the same

procedure $\mathcal{O}_0(M)$: Game G_0

$m_1,\ldots,m_\ell \xleftarrow{d} M$; $v_0 \leftarrow IV$
For $i = 1$ to ℓ do
  $v_i \leftarrow$ Choose-$f(0, v_{i-1}, m_i)$
  $\mathcal{Q} \xleftarrow{\cup} ((v_{i-1}, m_i), v_i)$
Ret v_ℓ

procedure $\mathcal{O}_1(v, m)$:

$w \leftarrow$ Choose-$f(0, v, m)$
$\mathcal{Q} \xleftarrow{\cup} ((v, m), w)$
Ret w

subroutine Choose-$f(s, v, m)$:

If $\mathtt{f}[s, v, m] = \perp$ then $\mathtt{f}[s, v, m] \xleftarrow{\$} \{0,1\}^n$
Ret $\mathtt{f}[s, v, m]$

procedure $\mathcal{O}_0(M)$: Games $\boxed{\mathrm{G}_1}$, G_2

$m_1,\ldots,m_\ell \xleftarrow{d} M$; $v_0 \leftarrow IV$; $\mathtt{V}[IV] \leftarrow \varepsilon$
For $i = 1$ to ℓ do
  $v_i \leftarrow$ Choose-$f(0, v_{i-1}, m_i)$
  $\mathtt{V}[v_i] \leftarrow m_1 \cdots m_i$
  $\mathcal{Q} \xleftarrow{\cup} ((v_{i-1}, m_i), v_i)$
Ret v_ℓ

procedure $\mathcal{O}_1(v, m)$:

$\mathtt{V}[IV] \leftarrow \varepsilon$
If $\mathtt{V}[v] \neq \perp$ then
  $w \leftarrow$ Choose-$f(0, v, m)$
  $\mathtt{V}[w] \leftarrow \mathtt{V}[v] \,\|\, m$
If $\mathtt{V}[v] = \perp$ then
  $w \leftarrow$ Choose-$f(1, v, m)$
$\mathcal{Q} \xleftarrow{\cup} ((v, m), w)$
Ret w

subroutine Choose-$f(s, v, m)$:

$s' \leftarrow s$
If $\mathtt{f}[1-s, v, m] \neq \perp$ then bad $\leftarrow$ true ; $\boxed{s' \leftarrow 1-s}$
If $\mathtt{f}[s', v, m] = \perp$ then $\mathtt{f}[s', v, m] \xleftarrow{\$} \{0,1\}^n$
Ret $\mathtt{f}[s', v, m]$

procedure $\mathcal{O}_0(M)$: Games G_3 , $\boxed{\mathrm{G}_4}$

$m_1,\ldots,m_\ell \xleftarrow{d} M$; $v_0 \leftarrow IV$; $\mathtt{V}[IV] \leftarrow \varepsilon$
For $i = 1$ to ℓ do
  $v_i \leftarrow$ Choose-$f(0, v_{i-1}, m_i)$
  $\mathtt{V}[v_i] \leftarrow m_1 \cdots m_i$
  $\mathcal{Q} \xleftarrow{\cup} ((v_{i-1}, m_i), v_i)$
Ret v_ℓ

procedure $\mathcal{O}_1(v, m)$:

$\mathtt{V}[IV] \leftarrow \varepsilon$
If $\mathtt{V}[v] \neq \perp$ then
  $w \leftarrow$ Choose-$f(0, v, m)$
  $\mathtt{V}[w] \leftarrow \mathtt{V}[v] \,\|\, m$
If $\mathtt{V}[v] = \perp$ then
  $w \leftarrow$ Choose-$f(1, v, m)$
$\mathcal{Q} \xleftarrow{\cup} ((v, m), w)$
Ret w

subroutine Choose-$f(s, v, m)$:

If $\mathtt{f}[s, v, m] \neq \perp$ then $\mathtt{f}[s, v, m] \xleftarrow{\$} \{0,1\}^n$ $\boxed{\setminus \mathcal{R}}$
$\mathcal{R} \xleftarrow{\cup} \mathtt{f}[s, v, m]$
Ret $\mathtt{f}[s, v, m]$

procedure $\mathcal{O}_0(M)$: Games $\boxed{\mathrm{G}_5}$, G_6

$m_1,\ldots,m_\ell \xleftarrow{d} M$; $v_0 \leftarrow IV$; $\mathtt{V}[IV] \leftarrow \varepsilon$
For $i = 1$ to ℓ do
  $v_i \leftarrow$ Choose-$f(0, \mathtt{V}[v_{i-1}], m_i)$
  $\mathtt{V}[v_i] \leftarrow m_1 \cdots m_i$
  $\mathcal{Q} \xleftarrow{\cup} ((v_{i-1}, m_i), v_i)$
Ret v_ℓ

procedure $\mathcal{O}_1(v, m)$:

$\mathtt{V}[IV] \leftarrow \varepsilon$
If $\mathtt{V}[v] \neq \perp$ then
  $w \leftarrow$ Choose-$f(0, \mathtt{V}[v], m)$
  $\mathtt{V}[w] \leftarrow \mathtt{V}[v] \,\|\, m$
If $\mathtt{V}[v] = \perp$ then
  $w \leftarrow$ Choose-$f(1, v, m)$
$\mathcal{Q} \xleftarrow{\cup} ((v, m), w)$
Ret w

subroutine Choose-$f(s, v, m)$:

If $\mathtt{f}[s, v, m] = \perp$ then $\mathtt{f}[s, v, m] \xleftarrow{\$} \{0,1\}^n$ $\boxed{\setminus \mathcal{R}}$
$\mathcal{R} \xleftarrow{\cup} \mathtt{f}[s, v, m]$
Ret $\mathtt{f}[s, v, m]$

Figure 7: Games used in proof of Theorem 7.1. All games also have a procedure $\mathcal{O}_2$ that returns $\mathcal{Q}$ (not shown for brevity).

procedure $\mathcal{O}_0(M)$: Game $\mathrm{G}_{2,1}$	**procedure** $\mathcal{O}_0(M)$: Game $\mathrm{G}_{2,2}$

$m_1,\ldots,m_\ell \xleftarrow{d} M$; $v_0 \leftarrow IV$; $\mathtt{V}[IV] \leftarrow \varepsilon$
For $i = 1$ to ℓ do
  $v_i \leftarrow$ Choose-$f(0, v_{i-1}, m_i)$
  $\mathtt{V}[v_i] \leftarrow m_1 \cdots m_i$
  $\mathcal{Q} \xleftarrow{\cup} ((v_{i-1}, m_i), v_i)$
Ret v_ℓ
 	
procedure $\mathcal{O}_1(v, m)$:
$\mathtt{V}[IV] \leftarrow \varepsilon$
If $\mathtt{V}[v] \neq \perp$ then
  $w \leftarrow$ Choose-$f(0, v, m)$
  $\mathtt{V}[w] \leftarrow \mathtt{V}[v] \parallel m$
If $\mathtt{V}[v] = \perp$ then
  $w \leftarrow$ Choose-$f(1, v, m)$
$\mathcal{Q} \xleftarrow{\cup} ((v, m), w)$
Ret w
 	
subroutine Choose-$f(s, v, m)$:
$j \leftarrow j + 1$; $(s^j, v^j, m^j) \leftarrow (s, v, m)$
If $\exists i < j$ s.t. $(v^i, m^i) = (v^j, m^j) \wedge s^i \neq s^j$ then
  $\mathsf{bad} \leftarrow \mathsf{true}$
If $\mathtt{f}[s, v, m] = \perp$ then $\mathtt{f}[s, v, m] \leftarrow\!\$\, \{0,1\}^n$
Ret $\mathtt{f}[s, v, m]$
	$m_1,\ldots,m_\ell \xleftarrow{d} M$; $v_0 \leftarrow IV$; $\mathtt{V}[IV] \leftarrow \varepsilon$
	For $i = 1$ to ℓ do
	  $v_i \leftarrow$ Choose-$f(0, v_{i-1}, m_i)$
	  $\mathtt{V}[v_i] \leftarrow m_1 \cdots m_i$
	  $\mathcal{Q} \xleftarrow{\cup} ((v_{i-1}, m_i), v_i)$
	Ret v_ℓ
	
	procedure $\mathcal{O}_1(v, m)$:
	$\mathtt{V}[IV] \leftarrow \varepsilon$
	If $\mathtt{V}[v] \neq \perp$ then
	  $w \leftarrow$ Choose-$f(0, v, m)$
	  $\mathtt{V}[w] \leftarrow \mathtt{V}[v] \parallel m$
	If $\mathtt{V}[v] = \perp$ then
	  $w \leftarrow$ Choose-$f(1, v, m)$
	$\mathcal{Q} \xleftarrow{\cup} ((v, m), w)$
	Ret w
	
	subroutine Choose-$f(s, v, m)$:
	$j \leftarrow j + 1$; $(s^j, v^j, m^j) \leftarrow (s, v, m)$
	If $\exists i < j$ s.t. $(v^i, m^i) = (v^j, m^j) \wedge s^i = 1 \neq 0 = s^j$ then
	  $\mathsf{bad}_1 \leftarrow \mathsf{true}$
	If $\exists i < j$ s.t. $(v^i, m^i) = (v^j, m^j) \wedge s^i = 0 \neq 1 = s^j$ then
	  $\mathsf{bad}_2 \leftarrow \mathsf{true}$
	If $\mathtt{f}[s, v, m] = \perp$ then $\mathtt{f}[s, v, m] \leftarrow\!\$\, \{0,1\}^n$
	Ret $\mathtt{f}[s, v, m]$

Figure 8: Games used in proof of Theorem 7.1. All games also have a procedure $\mathcal{O}_2$ that returns $\mathtt{Q}$ (not shown for brevity).

in G_1 as in G_0, justifying (11).

(Game G_2, boxed statement omitted) Games G_1 and G_2 are identical-until-bad, since their only difference is whether or not the boxed statement is included. The fundamental lemma of game playing [2] justifies (12).

(Game G_3, boxed statement omitted) Game G_3 dispenses with the extra checks in Choose-f (that, in G_1 ensured consistency between calls with $s = 0$ and $s = 1$, but were no longer used in G_2). Additionally a set $\mathcal{R}$ is added that records all the random choices made in Choose-f. The functionality of G_3 is unchanged from G_2, justifying (13).

(Game G_4, boxed statement included) Game G_4 restricts sampling in Choose-f to not allow collisions between any two outputs. Games G_3 and G_4 are identical except for the boxed statement. Since Choose-f can be called a maximum of $q_0\sigma + q_1$ times, we have via a straightforward birthday analysis that

$$\Pr\left[\, A^{\mathrm{G}_3} \Rightarrow 1 \,\right] - \Pr\left[\, A^{\mathrm{G}_4} \Rightarrow 1 \,\right] \leq \frac{(q_0\sigma + q_1)^2}{2^{n+1}} ,$$

justifying (14).

(Game G_5, boxed statement included) Game G_5 is such that $\mathtt{f}[0, \cdot, \cdot]$ is indexed not by chaining variables, message block pairs but rather by message sequences, message block pairs. That is, wherever Choose-f was called on $(0, v, m)$ for chaining variable v and message block m in G_4 it is called on $(0, \mathtt{V}[v], m)$ in G_5. Since G_4 and G_5 never have collisions in outputs of Choose-f in both games there is a one-to-one correspondence between pairs $(0, v)$ and $(0, \mathtt{V}[v])$. Therefore this change does not affect the execution of the game, and so (15) has been justified.

(Game G_6, boxed statement omitted) Game G_5 and G_6 are identical except for the boxed statement, and the analysis justifying (14) above also applies to justify (16). Finally, we argue that G_6 implements oracles that are equivalent to $(\mathcal{F}_{eval}, \mathcal{S}^{\mathcal{F}}_{eval}, \mathcal{S}^{\mathcal{F}}_{reveal})$. In game G_6 the routine Choose-$f(0, \cdot, \cdot)$ implements the monolithic random oracle of the indiff-0 experiment (i.e. this implements $\mathcal{F}_{eval}$). While $\mathcal{S}$ uses a subroutine update() to maintain tables $\mathtt{V}$ and $\mathtt{Q}$, game G_6 updates these tables directly in response to queries to $\mathcal{O}_0$ or $\mathcal{O}_1$. However, since $\mathcal{S}$ calls update() immediately upon any query to it, the two methods for updating the tables are equivalent. Finally, note that in $\mathcal{O}_1$ when $\mathtt{V}[v] = \perp$ a freshly-chosen random point is returned, which is equivalent to the implementation of $\mathcal{S}^{\mathcal{F}}_{eval}$ (recall that A does not make pointless queries). We have justified (17).

UPPER BOUND ON SETTING bad IN G_2. All that remains is bounding the probability that bad is set in G_2, which we do with a second sequence of games $\mathrm{G}_2 \longrightarrow \mathrm{G}_{2,1} \longrightarrow \mathrm{G}_{2,2}$. See Figure 8.

(Game $\mathrm{G}_{2,1}$) This game implements the same functionality as G_2 but changes the way in which bad is set. Now Choose-f records each s, v, m query using the counter j and tuple (s^j, v^j, m^j). The flag bad is set if there exists a previous execution of Choose-f, let it be the i^{th} execution, such that $v^i = v^j$ and $m^i = m^j$ and yet $s^i \neq s^j$. In words, the current v, m query value matches a previous query value, but s does not. Since this is just another way of checking for the same event that caused bad to be set in G_2, we have that

$$\Pr\left[\ A^{\mathrm{G}_2} \text{ sets bad }\right] = \Pr\left[\ A^{\mathrm{G}_{2,1}} \text{ sets bad }\right] .$$

(Game $\mathrm{G}_{2,2}$) We split the setting of bad into two separate cases, represented by the (new) flags bad_1 and bad_2. The first is the case that the previous query had $s = 0$ and the later query had $s = 1$ and the second case is the opposite. A union bound gives that

$$\Pr\left[\ A^{\mathrm{G}_{2,1}} \text{ sets bad }\right] \le \Pr\left[\ A^{\mathrm{G}_{2,1}} \text{ sets } \mathsf{bad}_1\ \right] + \Pr\left[\ A^{\mathrm{G}_{2,1}} \text{ sets } \mathsf{bad}_2\ \right] .$$

Note, however, that the last term is zero. This is true since for bad_2 to be set due to Choose-$f(1, v, m)$ being called from $\mathcal{O}_1$, it must be the case that $\mathtt{V}[v] \neq \perp$. That there exists an $i < j$ for which $s^i = 0$ but $v^i = v$ (so that bad_2 is set) means that necessarily $\mathtt{V}[v^i] = \mathtt{V}[v] \neq \perp$. This contradiction justifies that bad_2 can never be set. Finally, we bound the probability of bad_1 being set. Let i, j be the numbers such that $i < j$ and $(v^i, m^i) = (v^j, m^j)$ and $s^i = 1$ and $s^j = 0$. Then it must be the case that $\mathtt{V}[v^i] = \perp$ at the i^{th} execution of Choose-f. However, at the j^{th} execution it must be the case that $\mathtt{V}[v^j] = \mathtt{V}[v^i] \neq \perp$ since $s^j = 0$. This means that between the i^{th} and j^{th} calls, the table entry $\mathtt{V}[v^i]$ was assigned a value. This can only occur if the output of execution of Choose-$f(0, v, m)$ (for some v, m) equals v^i. Combining the facts that all outputs of Choose-f are uniformly chosen and that there are at most $q_0\sigma + q_1$ executions of Choose-f with $s = 0$, we have that

$$\Pr\left[\ A^{\mathrm{G}_{2,1}} \text{ sets } \mathsf{bad}_1\ \right] \le \frac{q_1(q_0\sigma + q_1)}{2^n} .$$

This concludes the proof. ▮

7.2 Type-II PGV are pub-GPROs

It is easy to see that a RO or a pub-RO are indifferentiable from a pub-GRO, and by Theorem 7.1 can therefore be used within an iteration to build a pub-PRO hash function for arbitrary input lengths. However many hash functions (e.g. those built from block ciphers) do not utilize compression functions that are suitable for modeling as a (pub-)RO. In this section we show that many widely-used block-cipher-based compression functions, while not pub-PROs, are indifferentiable from a pub-GRO. As an example to build intuition, recall that an adversary can differentiate DM from a pub-RO by abusing the inverse oracle (see the attack described in Section 6). In the context of pub-GRO, such attacks fail because the adversary cannot query its first oracle with chaining variables not already returned by it. In fact, we prove that any Type-II PGV function (see Section 5.1) is indifferentiable from a pub-GRO.

Fix a string $IV \in \{0,1\}^n$. Let h^P be a compression function built using ideal primitive P. Then $\bar{h}^P$ is the ITM implementing a guarded version of h. Initially a set $\mathcal{V}$ is empty. Then $\bar{h}^P(v, m)$ returns $\perp$ if $v \notin IV \cup \mathcal{V}$, and otherwise evalutes $h^P(v, m)$ to get value w, adds w to $\mathcal{V}$, and returns w. Then, for example, $\overline{\mathsf{DM}}$ is the guarded version of the Davies-Meyer compression function (defined in Section 2).

We use the guarded versions of compression functions for clarity in our analyses. In particular, the forthcoming results do not rely on implementing guarded versions of compression functions — rather when an unguarded compression function is used within the iteration then only valid v are used with it, meaning our results apply directly. (Formally speaking, we could also just restrict adversaries from querying any inputs which would result in $\perp$ being returned, and the guarding logic is superfluous. We choose the explicit approach to render transparent this usage scenario.) Our results do not apply for unguarded compression functions when they are used outside the context of iteration-based hashing.

Recall that the Type-II PGV functions are those for which C^{PRE} is a bijection, $C^{\text{POST}}(v, m, \cdot)$ is a bijection, and $C_1^{-\text{PRE}}(k, \cdot)$ is a bijection. Here the map $C_1^{-\text{PRE}}\colon \{0,1\}^d \times \{0,1\}^n \to \{0,1\}^n$ is defined by $C_1^{-\text{PRE}}(k, m) = v$ where

subroutine Choose-$E(s,k,x)$:	**subroutine** Choose-$D(s,k,y)$:
$y \leftarrow\!\!\$\ \{0,1\}^n$	$x \leftarrow\!\!\$\ \{0,1\}^n$
If $\mathtt{E}[s,k,x] \neq \bot$ then $y \leftarrow \mathtt{E}[s,k,x]$	If $\mathtt{D}[s,k,y] \neq \bot$ then $x \leftarrow \mathtt{D}[s,k,y]$
$\mathtt{E}[s,k,x] \leftarrow y$; $\mathtt{D}[s,k,y] \leftarrow x$	$\mathtt{E}[s,k,x] \leftarrow y$; $\mathtt{D}[s,k,y] \leftarrow x$
Ret y	Ret x

Figure 9: Subroutines used by games G_0, G_1, G_2, and G_3 for the proof of Theorem 7.2.

$(v, m) = C^{-\text{PRE}}(k, x)$. (This is simply the projection of $C^{-\text{PRE}}$ to its left output.)

It is interesting to note that the four Type-I PGV functions that are not also Type-II are *not* pub-GPRO. Consider the (guarded) MMO compression function $\overline{\mathsf{MMO}}^E(v, m) = E_v(m) \oplus m$ built from an ideal cipher $\mathcal{C}_{n,n}$. For any simulator $\mathcal{S}$, we give an adversary that can differentiate $\overline{\mathsf{MMO}}$ from a pub-GRO that works as follows given oracles $\mathcal{O}_0$, $\mathcal{O}_1$, and $\mathcal{O}_2$ (implementing $(\overline{\mathsf{MMO}}^E, E, D)$ or $(f_{geval}, \mathcal{S}_E, \mathcal{S}_D)$ respectively). It chooses arbitrary points $y \in \{0,1\}^n$ and queries $\mathcal{O}_2(IV, y)$ to get reply x. It then queries $\mathcal{O}_0(IV, x)$, retrieves w, and outputs a one if $y \oplus w = x$. If in the indiff-1 experiment, the adversary will output one always. In the indiff-0 experiment, $\mathcal{S}_D$ must respond with a value x such that $y \oplus f(IV, x) = x$, lest the adversary will return zero. This can't be done efficiently (since f is a random oracle), and so the adversary will succeed with probability close to one. Similar attacks can be fashioned for the three other only Type-I functions.[3]

Intuitively, the Type-II functions resist such attacks because of the third requirement, which ensures that any inverse query corresponds to a (close to) uniform chaining variable. Since uniformly distributed chaining variables are unlikely to ever end up being in the set $\mathcal{V}$ of allowed queries to $\mathcal{O}_0$, the final query of the adversary above wouldn't work for a Type-II function. The next theorem formalizes this, showing that any Type-II PGV function is a pub-GPRO.

Theorem 7.2 [Type-II PGV are pub-GPROs] Fix $d, n > 0$ and fix $IV \in \{0,1\}^n$. Let $\mathcal{C}_{d\times n,n} = (E, D)$ be an ideal cipher and let $\bar{h}^{\mathcal{C}}$ be the guarded implementation of a Type-II PGV function $h^{\mathcal{C}}$. There exists a simulator $\mathcal{S} = (\mathcal{S}_E, \mathcal{S}_D)$ such that for any adversary A making at most (q_0, q_1, q_2) queries, we have

$$\mathbf{Adv}^{\text{pub-gpro}}_{\bar{h},\mathcal{C},\mathcal{S}}(A) \le \frac{(q_0+q_1+q_2)^2}{2^n} + \frac{q_2(q_0+q_1+1)}{2^n}$$

where $\mathcal{S}$ works in time $\mathcal{O}(q_2(q_0+q_1))$ and makes at most q_1 queries. $\square$

Proof: We fix the simulator $\mathcal{S} = (\mathcal{S}_E, \mathcal{S}_D)$ detailed below.

procedure $\mathcal{S}_E(k, x)$:
If $\mathtt{E}[k, x] \neq \bot$ then Ret $\mathtt{E}[k, x]$
$(m, v) \leftarrow C^{-\text{PRE}}(k, x)$
$w \leftarrow f_{eval}(v, m)$
$y \leftarrow C^{-\text{POST}}(v, m, w)$
Ret y

procedure $\mathcal{S}_D(k, y)$:
$((v^1, m^1), w^1), \ldots, ((v^\ell, m^\ell), w^\ell) \leftarrow f_{reveal}()$
For $i = 1$ to ℓ do
  $(k^i, x^i) \leftarrow C^{\text{PRE}}(v^i, m^i)$
  $y^i \leftarrow C^{-\text{POST}}(v^i, m^i, w^i)$
  If $k = k^i$ and $y = y^i$ then Ret x^i
$x \leftarrow\!\!\$\ \{0,1\}^n$; $\mathtt{E}[k, x] \leftarrow y$
Ret x

The simulator associates queries by the adversary to $\mathcal{S}_E$ and $\mathcal{S}_D$ with queries to f_{geval}. Note that C^{PRE} and $C^{\text{POST}}(v, m, \cdot)$ are bijections (the latter for any v, m), and their inverses via $C^{-\text{PRE}}$ and $C^{-\text{POST}}(v, m, \cdot)$ (see Section 5.1). The simulator uses these functions to map between inputs and outputs of the block cipher and inputs and outputs of f.

We utilize a sequence of games $G_0 \longrightarrow \cdots \longrightarrow G_6$ to bound A's advantage relative to $\mathcal{S}$. See Figures 10 and 11. Some of these games utilize as subroutines those shown in Figure 9, which together implement a modified version of an ideal cipher in which sampling with replacement is done (instead of the usual sampling without replacement). (The parameter s is used

[3] Note that [15] point out that these four just type-I PGV functions are also not suitable compression functions for building PROs via prefix-free encoding messages and then iteration.

procedure $\mathcal{O}_0(v,m)$: Game G_0

If $v \notin \{IV\} \cup \mathcal{V}$ then Ret $\perp$
$(k,x) \leftarrow C^{\text{PRE}}(v,m)$
$y \leftarrow$ Choose-$E(0,k,x)$
$w \leftarrow C^{\text{POST}}(v,m,y)$
$\mathcal{V} \xleftarrow{\cup} w$
Ret w

procedure $\mathcal{O}_1(k,x)$:

$y \leftarrow$ Choose-$E(0,k,x)$
Ret y

procedure $\mathcal{O}_2(k,y)$:

$x \leftarrow$ Choose-$D(0,k,y)$
Ret x

procedure $\mathcal{O}_0(v,m)$: Game G_1

If $v \notin \{IV\} \cup \mathcal{V}$ then Ret $\perp$
$(k,x) \leftarrow C^{\text{PRE}}(v,m)$
$y \leftarrow$ Choose-$E(0,k,x)$
$w \leftarrow C^{\text{POST}}(v,m,y)$
$i \leftarrow i+1$; $((v^i,m^i),w^i) \leftarrow ((v,m),w)$
$\mathcal{V} \xleftarrow{\cup} w$
Ret w

procedure $\mathcal{O}_1(k,x)$:

$y \leftarrow$ Choose-$E(0,k,x)$
$w \leftarrow C^{\text{POST}}(C^{-\text{PRE}}(k,x),y)$
$i \leftarrow i+1$; $((v^i,m^i),w^i) \leftarrow (C^{-\text{PRE}}(k,x),w)$
Ret y

procedure $\mathcal{O}_2(k,y)$:

For $j=1$ to i do
    $(k^i,x^i) \leftarrow C^{\text{PRE}}(v^i,m^i)$
    $y^i \leftarrow C^{-\text{POST}}(v^i,m^i,w^i)$
    If $k=k^i$ and $y=y^i$ then Ret x^i
$x \leftarrow$ Choose-$D(0,k,y)$
Ret x

procedure $\mathcal{O}_0(v,m)$: Games $\boxed{G_2}$ G_3

If $v \notin \{IV\} \cup \mathcal{V}$ then Ret $\perp$
$(k,x) \leftarrow C^{\text{PRE}}(v,m)$
$s \leftarrow 0$; If $\mathtt{E}[1,k,x] \neq \perp$ then bad $\leftarrow$ true $\boxed{;\ s \leftarrow 1}$
$y \leftarrow$ Choose-$E(s,k,x)$
$w \leftarrow C^{\text{POST}}(v,m,y)$
$i \leftarrow i+1$; $((v^i,m^i),w^i) \leftarrow ((v,m),w)$
$\mathcal{V} \xleftarrow{\cup} w$
Ret w

procedure $\mathcal{O}_1(k,x)$:

If $\mathtt{E}[1,k,x] \neq \perp$ then Ret Choose-$E(1,m,c)$
$y \leftarrow$ Choose-$E(0,k,x)$
$w \leftarrow C^{\text{POST}}(C^{-\text{PRE}}(k,x),y)$
$i \leftarrow i+1$; $((v^i,m^i),w^i) \leftarrow (C^{-\text{PRE}}(k,x),w)$
Ret y

procedure $\mathcal{O}_2(k,y)$:

For $j=1$ to i do
    $(k^i,x^i) \leftarrow C^{\text{PRE}}(v^i,m^i)$
    $y^i \leftarrow C^{-\text{POST}}(v^i,m^i,w^i)$
    If $k=k^i$ and $y=y^i$ then Ret x^i
$s \leftarrow 1$; If $\mathtt{D}[0,k,y] \neq \perp$ then bad $\leftarrow$ true $\boxed{;\ s \leftarrow 0}$
$x \leftarrow$ Choose-$D(s,k,y)$
Ret x

procedure $\mathcal{O}_0(v,m)$: Game G_4

If $v \notin \{IV\} \cup \mathcal{V}$ then Ret $\perp$
$(k,x) \leftarrow C^{\text{PRE}}(v,m)$
If $\mathtt{E}[1,m,c] \neq \perp$ then bad $\leftarrow$ true
$y \leftarrow f(k,x)$; $w \leftarrow C^{\text{POST}}(v,m,y)$
$\mathtt{D}[0,k,y] \leftarrow x$
$i \leftarrow i+1$; $((v^i,m^i),w^i) \leftarrow ((c,m),w)$
$\mathcal{V} \xleftarrow{\cup} w$
Ret w

procedure $\mathcal{O}_1(k,x)$:

If $\mathtt{E}[1,k,x] \neq \perp$ then Ret $\mathtt{E}[1,k,x]$
$y \leftarrow f(k,x)$; $w \leftarrow C^{\text{POST}}(C^{-\text{PRE}}(k,x),y)$
$\mathtt{D}[0,k,y] \leftarrow x$
$i \leftarrow i+1$; $((v^i,m^i),w^i) \leftarrow (C^{-\text{PRE}}(k,x),w)$
Ret y

procedure $\mathcal{O}_2(k,y)$:

For $j=1$ to i do
    $(k^i,x^i) \leftarrow C^{\text{PRE}}(v^i,m^i)$
    $y^i \leftarrow C^{-\text{POST}}(v^i,m^i,w^i)$
    If $k=k^i$ and $y=y^i$ then Ret x^i
If $\mathtt{D}[0,k,y] \neq \perp$ then bad $\leftarrow$ true
$x \leftarrow_{\$} \{0,1\}^n$; $\mathtt{E}[1,k,x] \leftarrow y$
Ret x

Figure 10: Games used in proof of Theorem 7.2. Subroutines Choose-E and Choose-D are detailed in Figure 9. Game G_4 uses a random function f mapping from $\{0,1\}^d \times \{0,1\}^n$ to $\{0,1\}^n$.

procedure $\mathcal{O}_0(v, m)$: Game G_5

If $v \notin \{IV\} \cup \mathcal{V}$ then Ret $\perp$
$(k, x) \leftarrow C^{\mathrm{PRE}}(v, m)$
If $\mathtt{E}[1, k, x] \neq \perp$ then $\mathsf{bad} \leftarrow \mathsf{true}$
$w \leftarrow f(C^{-\mathrm{PRE}}(k, x))$; $y \leftarrow C^{-\mathrm{POST}}(v, m, w)$
$\mathtt{D}[0, k, y] \leftarrow x$
$i \leftarrow i + 1$; $((v^i, m^i), w^i) \leftarrow ((v, m), w)$
$\mathcal{V} \xleftarrow{\cup} w$
Ret w

procedure $\mathcal{O}_1(k, x)$:

If $\mathtt{E}[1, m, c] \neq \perp$ then Ret $\mathtt{E}[1, m, c]$
$w \leftarrow f(C^{-\mathrm{PRE}}(k, x))$; $y \leftarrow C^{-\mathrm{POST}}(C^{-\mathrm{PRE}}(k, x), w)$
$\mathtt{D}[0, k, y] \leftarrow x$
$i \leftarrow i + 1$; $((v^i, m^i), w^i) \leftarrow (C^{-\mathrm{PRE}}(k, x), w)$
Ret y

procedure $\mathcal{O}_2(k, y)$:

For $j = 1$ to i do
    $(k^i, x^i) \leftarrow C^{\mathrm{PRE}}(v^i, m^i)$
    $y^i \leftarrow C^{-\mathrm{POST}}(v^i, m^i, w^i)$
    If $k = k^i$ and $y = y^i$ then Ret x^i
If $\mathtt{D}[0, k, y] \neq \perp$ then $\mathsf{bad} \leftarrow \mathsf{true}$
$x \leftarrow_{\$} \{0,1\}^n$; $\mathtt{E}[1, k, x] \leftarrow y$
Ret x

procedure $\mathcal{O}_0(v, m)$: Game G_6

If $v \notin \{IV\} \cup \mathcal{V}$ then Ret $\perp$
$w \leftarrow f(v, m)$
$i \leftarrow i + 1$; $((v^i, m^i), w^i) \leftarrow ((v, m), w)$
$\mathcal{V} \xleftarrow{\cup} w$
Ret w

procedure $\mathcal{O}_1(k, x)$:

If $\mathtt{E}[1, m, c] \neq \perp$ then Ret $\mathtt{E}[1, m, c]$
$w \leftarrow f(C^{-\mathrm{PRE}}(k, x))$
$i \leftarrow i + 1$; $((v^i, m^i), w^i) \leftarrow (C^{-\mathrm{PRE}}(k, x), w)$
Ret y

procedure $\mathcal{O}_2(k, y)$:

For $j = 1$ to i do
    $(k^i, x^i) \leftarrow C^{\mathrm{PRE}}(v^i, m^i)$
    $y^i \leftarrow C^{-\mathrm{POST}}(v^i, m^i, w^i)$
    If $k = k^i$ and $y = y^i$ then Ret x^i
$x \leftarrow_{\$} \{0,1\}^n$; $\mathtt{E}[1, k, x] \leftarrow y$
Ret x

Figure 11: Game used in the proof of Theorem 7.2. Games G_5 and G_6 use a random function f mapping from $\{0,1\}^n \times \{0,1\}^d$ to $\{0,1\}^n$.

to allow multiple instances of the objects. Namely, different s indicate distinct tables $\mathtt{E}$ and $\mathtt{D}$.) We will justify that

$$
\begin{aligned}
\Pr\left[\, \mathbf{Exp}^{\text{indiff-1}}_{\mathsf{DM},E,A} \Rightarrow 1 \,\right] &\leq \Pr\left[\, A^{\mathrm{G}_0} \Rightarrow 1 \,\right] + \frac{(q_0+q_1+q_2)^2}{2^n} && (18)\\
&= \Pr\left[\, A^{\mathrm{G}_1} \Rightarrow 1 \,\right] + \frac{(q_0+q_1+q_2)^2}{2^n} && (19)\\
&= \Pr\left[\, A^{\mathrm{G}_2} \Rightarrow 1 \,\right] + \frac{(q_0+q_1+q_2)^2}{2^n} && (20)\\
&\leq \Pr\left[\, A^{\mathrm{G}_3} \Rightarrow 1 \,\right] + \frac{(q_0+q_1+q_2)^2}{2^n} + \Pr\left[\, A^{\mathrm{G}_3} \text{ sets bad} \,\right] && (21)\\
&= \Pr\left[\, A^{\mathrm{G}_4} \Rightarrow 1 \,\right] + \frac{(q_0+q_1+q_2)^2}{2^n} + \Pr\left[\, A^{\mathrm{G}_4} \text{ sets bad} \,\right] && (22)\\
&= \Pr\left[\, A^{\mathrm{G}_5} \Rightarrow 1 \,\right] + \frac{(q_0+q_1+q_2)^2}{2^n} + \Pr\left[\, A^{\mathrm{G}_5} \text{ sets bad} \,\right] && (23)\\
&= \Pr\left[\, A^{\mathrm{G}_6} \Rightarrow 1 \,\right] + \frac{(q_0+q_1+q_2)^2}{2^n} + \Pr\left[\, A^{\mathrm{G}_5} \text{ sets bad} \,\right] && (24)\\
&= \Pr\left[\, \mathbf{Exp}^{\text{indiff-0}}_{f,\mathcal{S},A} \Rightarrow 1 \,\right] + \frac{(q_0+q_1+q_2)^2}{2^n} + \Pr\left[\, A^{\mathrm{G}_5} \text{ sets bad} \,\right] && (25)
\end{aligned}
$$

and then conclude by bounding the probability of bad being set in game G_4.

(Game G_0) Game G_0 implements the oracles $\bar{h}^E, E, D$ but where ideal cipher E (with inverse D) is implemented using the subroutines of Figure 9. A straightforward birthday argument justifies (18).

(Game G_1) Game G_1 is the same as G_0 except that book-keeping code is added to procedures $\mathcal{O}_0$ and $\mathcal{O}_1$ which is then used in $\mathcal{O}_2$. The new $\mathcal{O}_2$ functionality checks explicitly for previous queries to $\mathcal{O}_1$ or $\mathcal{O}_2$ that have already set $\mathtt{D}[0,k,y]$ (i.e. due to an execution of Choose-$E(0,k,x)$ that sampled range point y). To show that this is in fact the case, we need to show that anytime x^i is returned in $\mathcal{O}_2$ on line 123, the same value would have been returned on line 125. Let k, y be a pair queried to $\mathcal{O}_2$. Suppose there exists an i such that $k^i = k$ and $y^i = y$ where $(k^i, x^i) = C^{\text{PRE}}(v^i, m^i)$ and $y^i = C^{-\text{PRE}}(v^i, m^i, w^i)$. Then consider if v^i, m^i, w^i were assigned values on line 104. This means that Choose-$E(0, C^{\text{PRE}}(v^i, m^i)) =$ Choose-$E(0, k^i, x^i)$ was executed and it returned a value y', meaning $\mathtt{D}[0, k^i, y'] = x^i$. Since C^{PRE} and $C^{\text{POST}}(v^i, m^i, \cdot)$ are bijections we have that $y' = y^i = y$. Thus, $\mathtt{D}[0,k,y] = x^i$. Consider now if v^i, m^i, w^i were instead assigned values on line 112. Then again Choose-$E(0, C^{\text{PRE}}(v^i, m^i)) =$ Choose-$E(0, k^i, x^i)$ was executed and returned a value y', meaning $\mathtt{D}[0, k^i, y'] = x^i$. But now $w^i = C^{\text{POST}}(C^{-\text{PRE}}(k^i, x^i), y')$ and the bijectivity of C^{POST} and C^{PRE} gives that $y' = y^i = y$. Thus, $\mathtt{D}[0,k,y] = x^i$. We have justified (19).

(Game G_2, boxed statements included) Game G_2 is a modification of G_1 in that two pairs of tables (corresponding to $s = 0$ and $s = 1$) are used to track points defined by the oracles. The checks on lines 201, 210, and 222 ensure, however, that the two tables are used in a manner that mimics exactly the behavior of the single pair of tables in G_1. This justifies (20).

(Game G_3, boxed statements excluded) Game G_3 and G_2 are identical-until-bad, and the fundamental lemma of game-playing [2] justifies (21).

(Game G_4) In game G_4 f is a random oracle mapping points k, x to random values from $\{0,1\}^n$. It is used in conjunction with the other code on lines 403/404, 411/412, and 425, to completely remove use of Choose-E and Choose-D in the game. This simplification of the code of game G_3 implements identical functionality, justifying (22).

(Game G_5) In game G_5 we make two pairs of changes. First, we apply the random oracle f to $C^{-\text{PRE}}(k,x)$ as oppoesd to k, x on lines 503 and 511. (Technically, f is now a map on domain $\{0,1\}^n \times \{0,1\}^d$. Before it had a domain of $\{0,1\}^d \times \{0,1\}^n$.) Since C^{PRE} is bijective this change does not affect the distribution of the outputs of f. Second, we swap the order of assignment for y and w on lines 503 and 511, now assigning y to the output of f and then setting z as a function of y and v, m (or, equivalently, k, x). The change to line 503 is justified by the fact that $C^{\text{POST}}(v, m, \cdot)$ is a bijection for any v, m. The change to line 511 is justified since $C^{\text{POST}}(C^{-\text{PRE}}(k,x), \cdot)$ is a bijection for any k, x (since C^{PRE} is also a bijection). Thus the variables involved maintain the same (joint) distribution, justifying (23).

(Game G_6) In the final game we drop code that handled the setting of bad in G_5. Now we see that this final game G_6 is exactly implementing the oracles $(f_{geval}, \mathcal{S}_E^f, \mathcal{S}_D^f)$, justifying (24).

All that remains is to bound the probability of bad being set in game G_5. We proceed via case analysis.

The flag bad might be set due to line 502 or line 524. For line 524, however, the loop and conditional of 520 and 521 ensure that 524 will never be executed when $\mathtt{D}[0,k,y] \neq \bot$. To see this, note that $\mathtt{D}[0,k,y]$ is assigned a value only if one of lines 505 or 513 is also executed. Thus, if $\mathtt{D}[0,k,y] \neq \bot$ during a $\mathcal{O}_2(k,y)$ query, then necessarily there exists a $j \in [1 \mathinner{..} i]$ such that $k = k^j$ and $y = y^j$ where $(k^j, x^j) = C^{\text{PRE}}(v^j, m^j)$ and $y^j = C^{\text{POST}}(v^i, m^i, w^i)$. This is so because C^{PRE} and $C^{\text{POST}}(v^i, m^i, \cdot)$ are bijections. Thus, bad will never get set on line 524.

We turn to bounding the probability that line 502 sets bad. For bad to be set here, it must be that some query $\mathcal{O}_0(v,m)$ was made for which $v \in \{IV\} \cup \mathcal{V}$ at the time of the query and $\mathtt{E}[1,k,x] \neq \bot$. But $\mathtt{E}[1,k,x]$ can only be set to a non-bottom value if line 525 previously executed and here x was uniformly selected. Let $\mathcal{V}^*$ be the set $\mathcal{V}$ at the end of A's execution. Let x be a value sampled due to execution of line 523 for a query $\mathcal{O}_2(k,y)$. Then we want to assess the probability that A can make a later query v, m to $\mathcal{O}_0$ for which $C^{\text{PRE}}(v,m) = (k,x)$. Re-writing this last equation we have $(v,m) = C^{-\text{PRE}}(k,x)$ and thus $v = C_1^{-\text{PRE}}(k,x)$. Then we will justify that

$$
\begin{aligned}
\Pr\left[\, C^{\text{PRE}}(v,m) = (k,x) \wedge (v = IV \vee v \in \mathcal{V}^*) \,\right] &\leq \Pr\left[\, C_1^{-\text{PRE}}(k,x) = IV \vee C_1^{-\text{PRE}}(k,x) \in \mathcal{V}^* \,\right] \\
&\leq \Pr\left[\, C_1^{-\text{PRE}}(k,x) = IV \,\right] + \Pr\left[\, C_1^{-\text{PRE}}(k,x) \in \mathcal{V}^* \,\right] \\
&= \frac{1}{2^n} + \frac{q_0 + q_1}{2^n}
\end{aligned}
$$

where the indicated events are defined in the natural way (over the coins used in executing A^{G_5}). We can bound the first term on the right as follows. Since x is a uniformly-chosen point, and by the fact that $C_1^{-\text{PRE}}(k,\cdot)$ is a bijection, we have the probability that $C_1^{-\text{PRE}}(k,x) = IV$ with probability at most 2^{-n}. We can bound the second term as follows. Each value w added to $\mathcal{V}$ in the course of the game is the output of the random function f on some pair v, m. These points are chosen independently of the value v and there are at most $q_0 + q_1$ outputs of f chosen in the course of the game. Thus the probability of any one of these independent points being equal to $C_1^{-\text{PRE}}(k,x)$ is at most $(q_0 + q_1)/2^n$. Together, we see that for the probability of setting bad due to $\mathtt{E}[1,k,x]$ being set for any particular $\mathcal{O}_2$ query is at most $(1+q_0+q_1)/2^n$, justifying the equations above.

Finally, since there are at most q_2 queries to $\mathcal{O}_2$, and consequently at most q_2 values x sampled due to execution of line 525 we have via a straightforward union bound that

$$\Pr\left[\, A^{\mathrm{G}_5} \text{ sets } \mathsf{bad} \,\right] \le \frac{q_2(1+q_0+q_1)}{2^n} \, .$$

This completes the proof. ▌

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] D. Chang, S. Lee, M. Nandi, and M. Yung, Indifferentiable Security Analysis of Popular Hash Functions with Prefix-Free Padding. *Advances in Cryptology – ASIACRYPT '06*, LNCS vol. 4284, Springer, pp. 283–298, 2006.

[16] 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.

[17] J.S. Coron, Y. Dodis, C. Malinaud, and P. Puniya. Merkle-Damgård Revisited: How to Construct a Hash Function. *Advances in Cryptology – CRYPTO '05*, LNCS vol. 3621, Springer, pp. 21–39, 2005.

[18] I. Damgård. A design principle for hash functions. *Advances in Cryptology – CRYPTO '89*, LNCS vol. 435, Springer, pp. 416–427, 1989.

[19] 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.

[20] D. Davies and W. Price. The Application of Digital Signatures Based on Public Key Cryptosystems. *Proc. Fifth Intl. Computer Communications Conference*, pp. 525–530, October 1980.

[21] 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.

[22] 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.

[23] Y. Dodis, T. Ristenpart, and T. Shrimpton. Salvaging Merkle-Damgård for Practical Applications (full version of this paper). IACR ePrint Archive, 2009.

[24] 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.

[25] 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.

[26] S. Hirose. Some Plausible Constructions of Double-Length Hash Functions. *Fast Software Encryption – FSE '06*, LNCS vol. 4047, Springer, pp. 210–225, 2006.

[27] 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.

[28] 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.

[29] R. Merkle. One way hash functions and DES. *Advances in Cryptology – CRYPTO '89*, LNCS vol. 435, Springer, pp. 428–446, 1989.

[30] National Institute of Standards and Technology. FIPS PUB 180-1: Secure Hash Standard. (1995) Supersedes FIPS PUB 180 1993 May 11.

[31] 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.

[32] 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.

[33] 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.

[34] 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.

[35] 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.

[36] 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.

[37] M. Stam. Blockcipher Based Hashing Revisited. *Fast Software Encryption – FSE '09*, to appear, 2009.

[38] L. Wang. Personal correspondence. 2009.

[39] K. Yoneyama, S. Miyagawa, and K. Ohta. Leaky Random Oracle (Extended Abstract). Provable Security – ProvSec '08, LNCS vol. 5324, pp. 226–240, 2008.

A Proof of Theorem 3.2

Proof: We argue about WPrA security in the case of a single extraction query, and then apply Lemmas 3.3 and 3.4 to get the final result. We begin by defining the extractor $\mathcal{E}^+$; let it operate as follows:

algorithm $\mathcal{E}^+(z, \alpha)$:
Parse $(x_1, y_1), \cdots, (x_k, y_k) \leftarrow \alpha$
For $i = 1$ to k do
  If $y_i = z$ then $\mathcal{X} \leftarrow x_i$
If $\mathcal{X} = \emptyset$ then Ret $\perp$
Ret $\mathcal{X}$

That is, $\mathcal{E}^+$ simply iterates over the query-response pairs provided in the advice string and, upon finding a response that matches z, outputs the corresponding domain point. There are two cases to consider. First assume A made a P-query on x before extraction query (z, α). Then $\mathcal{E}^+$ will extract x from the advice string. Thus this case cannot contribute to A's advantage. On the other hand, assume A makes extraction query $\mathsf{Ex}(z, \alpha)$ for which z has not been already returned by P. Any subsequent P-query will return z with probability at most $1/2^n$. Since A can query at most q times to P, this means that the probability of hitting one such z is at most $q/2^n$. ▮

B Preimage Awareness of Iteration without Strengthening

We formalize a variant of preimage resistance, following [22]. Let $h^P : Dom \to Rng$ be a hash function for an ideal primitive P. An inversion adversary A takes no inputs, has access to a primitive oracle P, and outputs a point $x \in Dom$. For fixed value $IV \in Rng$, we define the experiment $\mathbf{Exp}^{\mathrm{inv}}_{h,P,IV,A}$ by the following pseudocode

$$x \leftarrow\!\!\$\; A^P \; ; \; \text{Ret } h^P(x) = IV$$

We associate to inv-adversary A, hash function H^P, and constant $IV \in Rng$ the advantage relation

$$\mathbf{Adv}^{\mathrm{inv}}_{h,P,IV}(A) = \Pr\left[\, \mathbf{Exp}^{\mathrm{inv}}_{h,P,IV,A} \Rightarrow \mathsf{true} \,\right] \; .$$

where the probability is taken over the coins used to execute the inv experiment. It is easy to verify that for the ideal-primitive-based compression functions we consider, the advantage in this game is low for any IV and any adversary. For example, for $f = \mathsf{RF}_{d+n,n}$, we have that $\mathbf{Adv}^{\mathrm{inv}}_{f,IV}(A) \leq q_p/2^n$ for any adversary A making at most q_p queries. We have the following result.

Theorem B.1 [Itr achieves PrA] Fix $n, d > 0$ and let P be an ideal primitive. Let $h^P : \{0,1\}^{n+d} \to \{0,1\}^n$ be a compression function, and let $H = \mathsf{Itr}[h^P]$ using some constant $IV \in \{0,1\}^n$. Let $\mathcal{E}_h$ be an arbitrary extractor for the PA-experiment involving h. Then there exists an extractor $\mathcal{E}_H$ such that for all pra-adversaries A making at most q_p primitive queries and q_e extraction queries and outputing a message of at most $\ell_{max} \geq 1$ blocks there exists an pra-adversary B and an inv-adversary C such that

$$\mathbf{Adv}^{\mathrm{pra}}_{H,P,\mathcal{E}_H}(A) \leq \mathbf{Adv}^{\mathrm{pra}}_{h,P,\mathcal{E}_h}(B) + \mathbf{Adv}^{\mathrm{inv}}_{h,P,IV}(C) \; .$$

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

The proof can be adapted easily from that used for SMD. Namely, the probability of the last case (where suffix-freeness is invoked) of the case analysis occurring can be shown to imply the existance of a natural inversion adversary.

C Group-2 PGV schemes are PrA in the iteration

Consider a generalized rate-1 blockcipher-based compression function, which operates as follows on input (v, m): $(k, x) \leftarrow C^{\text{PRE}}(v, m)$, $y \leftarrow\!\!\$\, E(k, x)$, $w \leftarrow C^{\text{POST}}(v, m, y)$, output w. We recall that a blockcipher-based compression function is Stam Type-II if: 1) C^{PRE} is bijective, 2) for all v, m the postprocessing $C^{\text{POST}}(v, m, \cdot)$ is bijective, and 3) for all k, the inverse map $C_1^{-\text{PRE}}(k, \cdot)$ is bijective. Here the map $C_1^{-\text{PRE}}\colon \{0,1\}^d \times \{0,1\}^n \to \{0,1\}^n$ is defined by $C_1^{-\text{PRE}}(k, m) = v$ where $(v, m) = C^{-\text{PRE}}(k, x)$. (This is simply the projection of $C^{-\text{PRE}}$ to its left output.) Stam has shown that the Group-2 PGV schemes from [7] are also Type-II.

Theorem C.1 [The Group-2/Type-II PGV schemes are PrA in the iteration.] Fix $\kappa, n > 0$, let $E \leftarrow\!\!\$\, \mathsf{BC}(\kappa, n)$. Let P be an ideal primitive providing an interfacs to E and E^{-1}. Let h^P be a Type-II blockcipher-based compression function, and let $H = \mathsf{Itr}[h^P]$ for some constant $IV \in \{0,1\}^n$. 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

$$\mathbf{Adv}^{\text{pra}}_{H,\mathcal{E}}(A) \leq \frac{q_e q_p}{2^n - q_p} + \frac{1.5 q_p (q_p + 1)}{2^n - q_p}$$

where $\mathcal{E}$ runs in time at most $\mathcal{O}\left(q_e(q_p^2 + q_p(\mathsf{Time}(C^{-\text{PRE}}) + \mathsf{Time}(C^{\text{POST}}))\right)$ $\square$

Proof: Let C be the event that when the PrA-adversary A halts, the advice string α contains the queries required to evaluate $H(M)$ and $H(M')$ for $M \neq M'$ such that $H(M) = H(M')$, or to evaluate $H(M)$ for some non-trivial M such that $H(M) = IV$.

Now we condition:

$$\begin{aligned} \Pr\left[\mathbf{Exp}^{\text{pra}}_{H,\mathcal{E},A} \Rightarrow 1\right] &= \Pr\left[\left(\mathbf{Exp}^{\text{pra}}_{H,\mathcal{E},A} \Rightarrow 1\right) \wedge \mathsf{C}\right] + \Pr\left[\left(\mathbf{Exp}^{\text{pra}}_{H,\mathcal{E},A} \Rightarrow 1\right) \wedge \neg\mathsf{C}\right] \\ &\leq \Pr\left[\left(\mathbf{Exp}^{\text{pra}}_{H,\mathcal{E},A} \Rightarrow 1\right) \mid \neg\mathsf{C}\right] + \Pr\left[\mathsf{C}\right] \end{aligned} \quad (26)$$

The proof of collision resistance for the Type-II schemes given in Stam [37] bounds $\Pr[\mathsf{C}] \leq .5 q_p (q_p + 1)/(2^n - q_p)$. We continue then under the assumption that $\neg\mathsf{C}$ holds.

The extractor $\mathcal{E}$ on input (z, α) operates as follows. Informally, it will build from the blockcipher queries in α an IV-rooted tree with vertices that are labeled by chaining values and edges labeled by message blocks. It then looks to see if z appears as a vertex label in the tree. If so, it returns the appropriate preimage by reading edge labels from the IV to z; if not, it returns $\perp$.

Let us be more formal. Let $\mathcal{L}$ be an initially empty list. Let $\mathcal{E}(z, \alpha)$ first parse the advice string $(k_1, x_1, y_1), \ldots, (k_r, x_r, y_r)$. Let $\mathcal{E}$ build a graph $T = (V, E)$ with vertex and edge labels; the vertex labeld by the initial value will simply be refered to by IV. Initially, $V = \{IV\}$ and $E = \{\}$. For $i \in [1..r]$ it does the following: $(m_i, v_i) \leftarrow C^{-\text{PRE}}(k_i, x_i)$, $w_i \leftarrow C^{\text{POST}}(v_i, m_i, y_i)$; if $w_i = z$ then an (initially false) flag $\mathrm{possible}$ is set to true. If $v_i = IV$, $\mathcal{E}$ adds (v_i, m_i, w_i) to $\mathcal{L}$ with the annotation "$\text{USED}, 0$", and creates edge (IV, w_i) in T with label m_i; else $\mathcal{E}$ simply adds $(v_i, m_i w_i)$ to $\mathcal{L}$. If the flag $\mathrm{possible}$ was not set to true during this loop, the extractor halts with output $\perp$. We pause in our description to note the following. Since C^{PRE} and $C^{\text{POST}}(v, m, \cdot)$ are bijections, each blockcipher query (k, x, y) defines a unique (compression function) tuple (v, m, w). Combined with the assumption that $\neg\mathsf{C}$ holds, any tuple marked as USED will never again be added to the graph. (Note that $\neg\mathsf{C}$ also implies that $v \neq w$.) Continuing our description of $\mathcal{E}$, for each $(v, m, w) \in \mathcal{L}$ that is not marked as USED, it searches the vertices at distance one from the IV (corresponding exactly to the $\mathcal{L}$-tuples labeled "$\text{USED}, 0$") for a vertex labeled v; if it finds one, it creates a new edge (v, w) labeled by m, and marks the tuple (v, m, w) as "$\text{USED}, 1$". The extractor continues in this way, making repeated passes over $\mathcal{L}$ and attempting to add un-USED tuples to the tree. Notice that by our assumptions, $\mathcal{E}$ only needs to compare un-USED tuples to tuples marked "USED, ℓ" for the highest value of ℓ in the list. (This is easily implement by a counter that keeps track of what is the current distance from the IV.) When either all tuples in $\mathcal{L}$ are marked as USED, or when a pass completes without adding any new edges to T, the extractor stops building. It then searches the tree for a vertex labeled z. If it finds one (and there will be at most one) it reads the edge labels from IV to z and returns the corresponding messsage. Otherwise, it returns $\perp$.

It is clear that (for a single extraction query) the running time of the extractor is $\mathcal{O}(r^2 + r(\mathsf{Time}(C^{-\text{PRE}}) + \mathsf{Time}(C^{\text{POST}}))$.

Now, since $\neg\mathsf{C}$ holds, the adversary must win by finding a preimage for some z upon which the extractor never returned a non-$\perp$ value. (It is possible that some z is queried more than once to $\mathcal{E}$ with different advice strings.) We call these z *useful*. Without loss of generality, we can assume that the adversary outputs a preimage for a useful z as soon as one can be computed from the query list. Thus we can also assume that all extractor queries are made prior to the adversary finding its eventual preimage, and so there are at most q_e useful z. It remains then to bound the probability that the adversary manages to find a preimage for any of these z.

As the adversary runs, we imaging building a graph $G = (V, E)$ much as the one above. The vertex set $V = \{0,1\}^n$ and the edge set is initially empty. When the adversary learns (via a P query) a triple (k, x, y), we compute $(v, m) \leftarrow C^{-\text{PRE}}(k, x)$ and $w \leftarrow C^{\text{POST}}(v, m, y)$, and place an edge between vertices v and w. Consider this graph at the time of the final extraction query. In particular, consider the subgraph G' that consists of the component containing the IV vertex and any component containing a vertex labeled with a useful z. Let $S \subseteq V$ be the set of vertices in this subgraph. We claim that if k queries to P have been made at the time of the last extraction query, then $|S| \leq 1 + q_e + k$. To see this, notice that S contains the IV vertex and the (at most) q_e vertices labeled by the useful z. Moreover, as components are by definition connected subgraphs, each of the k edges placed in G would have added at most one new vertex to G'.

Now, as A continues to make P queries after the final extraction query, each adds at most one new vertex to S. A necessary condition for A to win is for it to create a new edge between vertices already in S, in particular because the IV and all of the useful z are in S. (This is certainly not a sufficient condition.) Say this occurs on query $k + i$, for $i \in [1..q_p - k]$. If this is a forward blockcipher query (k, x) yielding y, then by the bijectivity of C^{PRE} and $C^{\text{POST}}(v, m, \cdot)$, the probability that $w = C^{\text{POST}}(v, m, y)$ is the label of a vertex already in S is at most $(1 + q_e + k + (i-1))/(2^n - (k + (i-1)))$. Similarly, if the necessary edge is created by an inverse blockcipher query (k, y) yielding x, then the bijectivity of $C_1^{-\text{PRE}}(k, \cdot)$ insures again that the probability of creating the edge is at most $(1 + q_e + k + (i-1))/(2^n - (k + (i-1)))$.

Thus, by a union bound

$$
\begin{aligned}
\Pr\left[\left(\mathbf{Exp}^{\text{pra}}_{H,\mathcal{E},A} \Rightarrow 1\right) \mid \neg\mathsf{C}\right] &\leq \sum_{i=1}^{q_p - k} \frac{q_e + k + i}{2^n - (k + (i-1))} \\
&\leq \frac{1}{2^n - q_p} \sum_{i=1}^{q_p - k} (q_e + k + i) \\
&= \frac{1}{2^n - q_p} \left((q_p - k)(q_e + k) + \frac{(q_p - k)(q_p - k + 1)}{2} \right) \\
&\leq \frac{q_p - k}{2^n - q_p} \left(\frac{2(q_e + k) + (q_p - k + 1)}{2} \right) \\
&\leq \frac{q_p - k}{2^n - q_p} \left(\frac{2q_e + q_p + k + 1}{2} \right) \\
&\leq \frac{q_p}{2^n - q_p} (q_e + q_p + 1)
\end{aligned}
$$

where the final line is overly conservative because it sets $k = 0$ outside of the parentheses, and $k = q_p$ inside. (We choose to use this looser bound for presentation purposes, and anyway it suffices.) Altogether then, (26) becomes

$$
\Pr\left[\mathbf{Exp}^{\text{pra}}_{H,\mathcal{E},A} \Rightarrow 1\right] \leq \frac{q_p(q_e + q_p + 1)}{2^n - q_p} + \frac{.5(q_p)(q_p + 1)}{2^n - q_p} = \frac{q_e q_p}{2^n - q_p} + \frac{1.5(q_p)(q_p + 1)}{2^n - q_p}
$$

completing the proof.

D Alternative Formulation for Preimage Awareness

A two-stage pa-adversary $A = (A_1, A_2)$ is a pair of algorithms. The challenge selection algorithm A_1 runs on no input and has access to a primitive oracle P. It outputs a triple (z, α, st) where $z \in \mathit{Rng}$, $\alpha \in \{0,1\}^*$ is an advice string, and $st \in \{0,1\}^*$ is a string representing arbitrary state information. The preimage selection algorithm A_2 runs on input z, st, has oracle access to P, and outputs a preimage $x' \in Dom$. Then the 1-PrA experiment $\mathbf{Exp}^{\text{1-pra}}_{H,\mathcal{E},A}$ is defined by the pseudocode

$$(z, \alpha, st) \leftarrow\!\!\$\; A_1^P \; ; \; x \leftarrow \mathcal{E}(z, \alpha) \; ; \; x' \leftarrow\!\!\$\; A_2^P(z, st) \; ; \; \text{Ret } (x \neq x' \wedge H^P(x') = z)$$

To H, $\mathcal{E}$, and A we associate the advantage relation

$$\mathbf{Adv}^{\text{1-pra}}_{H,\mathcal{E}}(A) = \Pr\left[\, \mathbf{Exp}^{\text{1-pra}}_{H,\mathcal{E},A} \Rightarrow \mathsf{true} \,\right]$$

where the probability is taken over the coins used in executing the experiment. The next theorem captures the simple hybrid that 1-PrA security implies full PrA security, but with a factor q_e (the number of extraction queries) loss in concrete security. The proof (omitted) is by a simple hybrid argument.

Theorem D.1 **[**1-PrA $\Rightarrow$ PrA**]** Let $H^P : Dom \rightarrow Rng$ be a hash function. Let $\mathcal{E}$ be an extractor. Then there exists an extractor $\mathcal{E}$ such that for any pra-adversary A making q_e extraction queries there exists a two-stage pra-adversary $B = (B_1, B_2)$ such that

$$\mathbf{Adv}^{\text{pra}}_{H,\mathcal{E}}(A) \leq q_e \cdot \mathbf{Adv}^{\text{1-pra}}_{H,\mathcal{E}}(B) \; .$$

B runs in time that of A plus $q_e \cdot \mathsf{Time}(\mathcal{E})$. $\square$