How (Not) to Efficiently Dither Blockcipher-Based Hash Functions?

Jean-Philippe Aumasson[1]* and Raphael C.-W. Phan[2]†

[1] FHNW, 5210 Windisch, Switzerland
[2] Loughborough Uni, LE11 3TU Leics, UK

Abstract. In the context of iterated hash functions, "dithering" designates the technique of adding an iteration-dependent input to the compression function in order to defeat certain generic attacks. The purpose of this paper is to identify methods for dithering blockcipher-based hash functions that provide security bounds and efficiency, contrary to the previous proposals. We considered 56 different constructions, based on the 12 secure PGV schemes. Proofs are given in the blackbox model that 12 of them preserve the bounds on collision and inversion resistance given by Black et al. These 12 schemes avoid the need for short dither values, induce negligible extra-computation, and achieve security independent of the dither sequence used. We also identify 8 schemes that lead to strong compression functions but potentially insecure hash functions. Application of our results can be considered to popular hash functions like SHA-1 or Whirlpool.

1 Introduction

The idea of making hash functions out of blockciphers goes back to 1978, when Rabin [40] proposed to hash $(m_1, \ldots, m_\ell)$ as $\mathsf{DES}_{m_\ell}(\ldots(\mathsf{DES}_{m_1}(IV))\ldots)$. Subsequent works devised less straightforward schemes, with either one or two calls to the blockcipher within a compression function [28, 30, 33, 37, 39]. In 1993 research went a step further when Preneel et al. [38] conducted a systematic analysis of all 64 compression functions of the form $f(h, m) = E_K(P) \oplus F$, for $K, P, F \in \{m, h, m \oplus h, v\}$, where v is a constant. They showed that only 4 of these schemes resist all considered vulnerabilities, and 8 others just have the non-critical attribute of easily found fixed-points. A decade later, Black et al. [9] proved the security of hash functions based on these 12 PGV schemes in the blackbox model.

Like a majority of hash functions blockcipher-based hash functions follow the Merkle-Damgård (MD) paradigm [16, 31]. Recent generic attacks [17, 20, 22–24] that exploit its structure led to proposals to extend the basic MD construction.

*Supported by the Swiss National Science Foundation under project number 113329.

†Work done while the author was with the Security & Cryptography Lab (LASEC), EPFL, Lausanne, Switzerland.

These include the idea of *dithering*, i.e. adding an input (the *dither*) to the compression function, whose value depends on the iteration count. The goal is to defeat attacks based on message block repetitions (like [17, 24]).

Proposals of dither sequences came from Kelsey and Schneier [24] (using a counter), from Biham and Dunkelman [5, 6] (as part of the HAIFA framework, using the number of bits hashed so far), and from Rivest [41] (using an abelian square-free sequence). However, the method proposed [6, 41] for integrating the dither value into concrete hash functions is inefficient, in the sense that it increases the number of calls to the compression function. This method indeed consists in reducing the effective size of a message block to make way for the dither, i.e. filling the dither into the space freed up. This motivated Rivest's proposal to use short dithers (2-byte) encoding particular patterns over a small alphabet. Another drawback of this method is that system parameters have to be modified such that message chunks become, for example, 448 bits long instead of 512 with a 64-bit counter, or 496 with Rivest's method. It thus seems valuable to explore generic dithering methods that preserve efficiency and system parameters, and that are still simple to apply.

1.1 Contribution

We will be concerned with the problem of constructing dithered compression functions from blockcipher-based schemes, grounding our work on the 12 secure PGV schemes. We first introduce 56 dithered variants, along with security definitions adapted for dithered functions. Our blackbox analysis singles out 12 dithered schemes leading to hash functions as secure as the original (undithered) ones, as far as collision and inversion resistance are concerned. The bounds given are *independent* of the dither sequence use, contrary to 32 other constructions. A counter-intuitive fact is proven, that 8 out of 56 dithered schemes lead to hash functions which are *not collision resistant* when the dithering method of HAIFA [6] is used, despite having a collision-resistant compression function. This re-opens the suitability issue of the Merkle-Damgård theorem for dithered hash functions, and suggests that a careful revisit is required.

We emphasize that our results say nothing on the resistance to generic second-preimage attacks as [17, 24] that dithering aims at preventing. The resistance to these attacks depends on the dither sequence used, whereas our point is to show that previously known security bounds can hold as well when dithering is used, independently of the sequence chosen.

Apart from our formal security analysis, the interest of our constructions is twofold: Firstly, they are efficient, because the number of calls to the compression function is no longer increased by dithering; secondly, they allow dither values of arbitrary length (up to the size of the key of the blockcipher), with no performance penalty. As a result, a counter supporting large messages can now be used. More generally, it avoids the need for short dither value with non-trivial patterns like [41], which in addition provides fewer security guarantees than a counter (see [12]).

1.2 Related Work

After a very calm period during the 90s, the results of [9] seem to have triggered a regain of interest for blockcipher-based hashing: In 2005 Black et al. [8] proved that a compression function of the form $f_2\left(h_{i-1}, m_i, E_K(f_1(h_{i-1}, m_i))\right)$ cannot be provably secure with respect to E_K. This result has been recently extended by Rogaway and Steinberger [43], who proved generic upper bounds on the security of permutation-based hash functions.

Along the same lines, combinations of fixed permutations were previously studied by Shrimpton and Stam [45]. Another impossibility result is due to Boneh and Boyen [11] for hash functions combiners, later generalized by Pietrzak [35]. In [29], Lee et al. extend the [9] results to 22 other constructions, using similar blackbox proofs, and in [46] Stam simplifies the [9] proofs. In [26], Knudsen and Rijmen study known-key distinguishers for blockciphers; though unrealistic for attacking encryption primitives, this scenario can be relevant for blockcipher-based hashing (they show near-collisions for Matyas-Meyer-Oseas, see Appendix B).

More concretely, the recent hash functions Maelstrom [19] and Grindahl [27] are based on AES, and blockcipher-based designs remain a promising alternative for several researchers (e.g. [25]). The NIST hash competition may also mark a revival of hash functions built on blockciphers.

Stream-cipher-based hash function attracted less attention. They offer a less confortable framework because (1) they are generally not defined to operate over "blocks", (2) until now they have been less reliable than blockciphers, and (3) they often have a slow initialization. A counter-example is Bernstein's compression function Rumba [3] is based on the stream cipher Salsa20. We can also cite [14], based on RC4.

Fewer works have been produced about dithering. We can cite Shoup's construction [44] for universal one-way hash functions, which can be seen as a kind of dithering (a sequence of values called the "schedule" is input through iterations, see also [32]). More recently, Bouillaguet et al. [12] presented another generic second-preimage attack, slower than [24] in general, but performing slightly better when certain dither sequences are used. For etymological issues, see Appendix C.

1.3 Notations

We adopt the notations of [9], with only minor changes: A *blockcipher* is a map $E : \{0,1\}^\kappa \times \{0,1\}^\mu \mapsto \{0,1\}^\mu$, such that $E_k(\cdot) = E(k,\cdot)$ is a permutation on $\{0,1\}^\mu$ for all $k \in \{0,1\}^\kappa$, and its inverse is written E^{-1}. The set of all blockciphers with κ-bit key and μ-bit messages is denoted $\mathsf{Bloc}(\kappa,\mu)$. A *blockcipher-based hash function* is a map $H : \mathsf{Bloc}(\kappa,\mu) \times D \mapsto R$, where $D \subseteq \{0,1\}^\star$ and $R = \{0,1\}^n$, defined iteratively by a compression function $f : \mathsf{Bloc}(\kappa,\mu) \times \{0,1\}^{n_1} \times \{0,1\}^{n_2} \mapsto \{0,1\}^{n_2}$, where n_1 is the size of a message block, and n_2 the size of chaining values. In the remainder of the paper, we assume $\mu = n_1 = n_2 = n$. We write f_E (resp. H_E) to denote the compression (resp.

hash) function instantiated with a particular E. Eventually, an *adversary* is an algorithm with oracle-access to E and E^{-1}—working within this setting is also known as analysis in the blackbox model, along with the assumption that each E_k is a random permutation. Furthermore, we set $h^+ = f(h, m)$ the output of a compression function. In the context of iterated hashing, blocks and chaining values are indexed as follows: h_0 is the IV, $\bar{m} = (m_1, \ldots, m_\ell)$ is the (encoded) message, $h_1 = f(h_0, m_1)$, and so on until $h_\ell = f(h_{\ell-1}, m_\ell) = H(\bar{m})$.

2 Dithering the PGV Schemes

The 12 PGV schemes $f_1, \ldots, f_{12}$ are depicted in Fig. 1 and 2: f_1 is the Matyas-Meyer-Oseas [30] construction (MMO), and one of the simplest schemes; f_3 is the Miyaguchi-Preneel construction, notably employed in Whirlpool [2], with as blockcipher a variant of Rijndael; f_5 is the Davies-Meyer construction, somehow the dual of MMO: Its structure is similar, except that the inputs of h and m play reversed roles. However, it has the undesirable attribute of easily found fixed-points; indeed, for an arbitrary m, choosing $h = {E^{-1}}_m(0)$ implies $f_5(h, m) = h$. The Davies-Meyer construction is used by the hash function Maelstrom-0 [19] (a variant of Whirlpool), and implicitly by some dedicated hash functions like MD5 and SHA-1.

We consider dithered versions obtained with a single input of the dither value d through an xor operation (in practice, it might be replaced by any easy-to-invert mapping which is a permutation for one of its inputs fixed). We suppose d non-null, and of convenient length (that is, not larger than its input slots in the blockcipher). The dithered PGV (dPGV) schemes are then classified into five subsets, describing the possible points for the dither d, see Table 1.

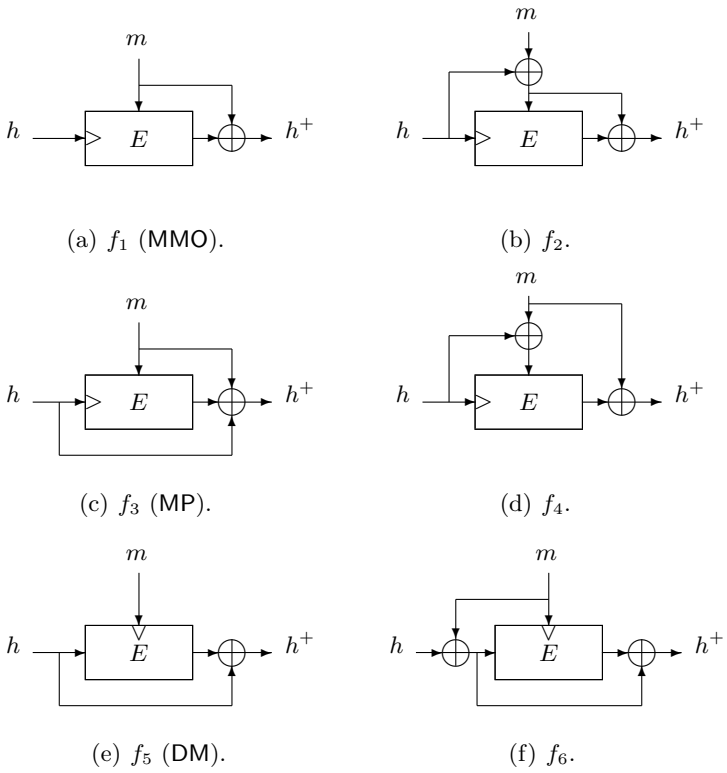
Table 1. Subsets of dithered PGV schemes.

Subset	Input point	Modification
$\mathcal{C}_1$	chaining value	$h \leftarrow h \oplus d$
$\mathcal{C}_2$	message block	$m \leftarrow m \oplus d$
$\mathcal{C}_3$	output	$h^+ \leftarrow h^+ \oplus d$
$\mathcal{C}_4$	key	$E_k \leftarrow E_{k \oplus d}$
$\mathcal{C}_5$	plaintext	$E_k(\cdot) \leftarrow E_k(\cdot \oplus d)$

We write $f_{i,j}$ for the dithered scheme obtained by applying the j-th transform to f_i; thus $f_{i,j} \in \mathcal{C}_j$, for all $(i, j) \in \{1, \ldots, 12\} \times \{1, \ldots, 5\}$. Clearly, $|\mathcal{C}_i| = 12$ for $i = 1, \ldots, 5$, but there are only 56 distinct dithered schemes, rather than 60, because $f_{1,1} \equiv f_{1,4}$, $f_{5,2} \equiv f_{5,4}$, $f_{9,1} \equiv f_{9,4}$, and $f_{10,2} \equiv f_{10,5}$.

A crucial observation is that almost all schemes of $\mathcal{C}_4$ and $\mathcal{C}_5$ are formally equivalent to a $\mathcal{C}_3$ scheme, up to variable renaming, e.g.

$$f_{4,4}(h, m, d) = E_{h \oplus d}(h \oplus m) \oplus m = E_{h'}(h' \oplus m') \oplus m' \oplus d = f_{4,3}(h', m', d),$$
$$f_{9,5}(h, m, d) = E_{h \oplus m}(m \oplus d) \oplus m = E_{h' \oplus m'}(m') \oplus m' \oplus d = f_{9,3}(h', m', d),$$

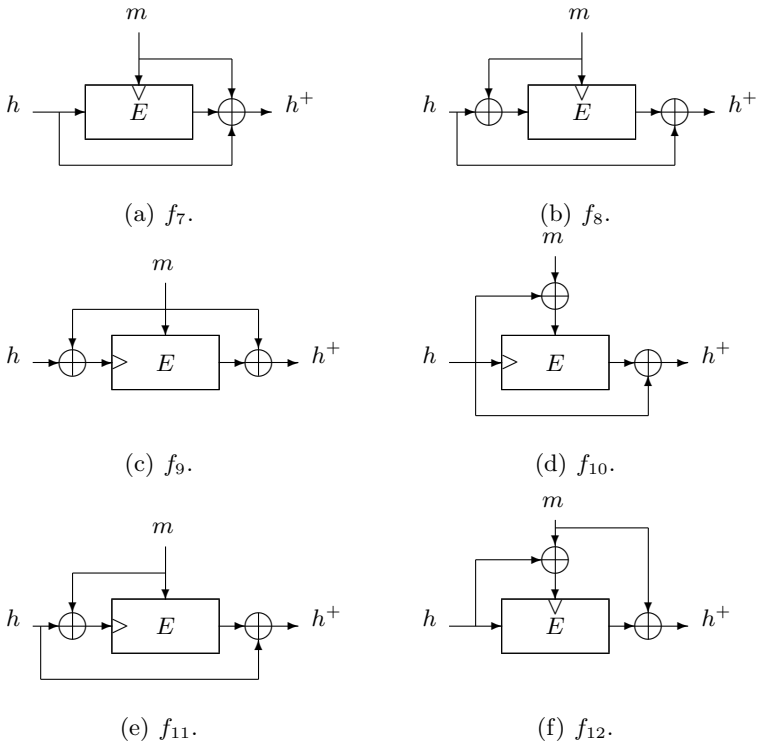


(a) f_1 (MMO). (b) f_2.

(c) f_3 (MP). (d) f_4.

(e) f_5 (DM). (f) f_6.

Fig. 1. PGV schemes f_1 to f_6, where a hatch marks the key input (we assume keys and message blocks of same size, cf. §1.3).



(a) f_7. (b) f_8.

(c) f_9. (d) f_{10}.

(e) f_{11}. (f) f_{12}.

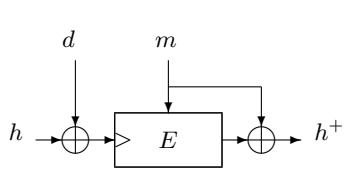
Fig. 2. PGV schemes f_7 to f_{10}.

for $h' = h \oplus d$, $m' = m \oplus d$. We denote $\mathcal{C}_3^+$ the set of schemes that can be expressed in $\mathcal{C}_3$ form. Only four members of $\mathcal{C}_4 \cup \mathcal{C}_5$ do not admit such rewriting, namely the ones equivalent to a scheme of $\mathcal{C}_1$ or $\mathcal{C}_2$. We thus have

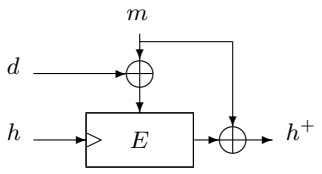
$$\mathcal{C}_3^+ = (\mathcal{C}_3 \cup \mathcal{C}_4 \cup \mathcal{C}_5) \setminus \{f_{1,4}, f_{5,4}, f_{9,4}, f_{10,5}\}.$$

This set is used later for simplifying security proofs (we shall exploit the $\mathcal{C}_3$ structure for proving security bounds on $\mathcal{C}_3^+$ schemes). To summarize, we have $|\mathcal{C}_1 \cup \mathcal{C}_2| = 24$, $|\mathcal{C}_3^+| = 32$, and $(\mathcal{C}_1 \cup \mathcal{C}_2) \cap \mathcal{C}_3^+ = \emptyset$.

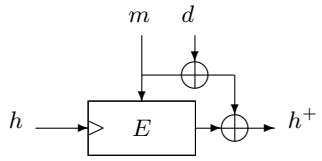
Note that if f_i admits easily found fixed-points (as do 8 of the 12 PGV schemes), then any dithered variant also possesses the property. For instance, $f_{5,5}$ admits the fixed-point ${E^{-1}}_m(0) \oplus d$, for any choice of m. It follows that exactly 37 among the 56 dithered schemes have trivial fixed-points.



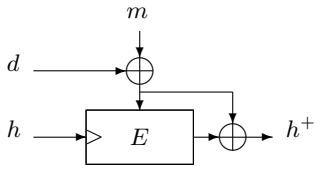
(a) dMMO_1: $h^+ = E_{h \oplus d}(m) \oplus m$.



(b) dMMO_2: $h^+ = E_h(m \oplus d) \oplus m$.



(c) dMMO_3: $h^+ = E_h(m) \oplus m \oplus d$.



(d) dMMO_4: $h^+ = E_h(m \oplus d) \oplus m \oplus d$.

Fig. 3. Dithered versions of the Matyas-Meyer-Oseas scheme.

To illustrate our constructions, Figure 3 depicts the dithered variants of MMO (f_1): For a given d, dMMO_1 is similar to MMO up to a reordering of the permutation indexes (more precisely, the h-th permutation takes $(h \oplus d)$ as new index). dMMO_2 simulates the undithered MMO for a blockcipher $E'(\cdot) = E(\cdot \oplus d)$, while dMMO_3 has simply the output of MMO xored with d, as in Shoup's method [44]. The structure of dMMO_4 is somewhat similar to the RMX transform for randomized hashing [21].

3 Security Definitions for Dithered Functions

We build on the formal definitions of [9] (recalled in Appendix A), extending them to the case of dithered functions: A collision for dithered compression functions where dithers are distinct is termed a Δ-collision—in essence, this is somewhat analogous to a free-start collision for hash functions, in the sense that here the dithers are distinct and public. Such a collision does not trivially translate into a collision for the derived hash function, mainly due to the MD-strengthening padding. We reserve the term *collision* to the case where a pair of inputs map to the same image with *same dither values*. This is in order to maintain the usefulness of the MD paradigm ported over to dithered hash functions.

In the following definitions, $\mathcal{A}$ is an adversary that has access to E and E^{-1}, and f is a dithered blockcipher-based compression function, $f : \mathsf{Bloc}(\kappa, n) \times \{0,1\}^n \times \{0,1\}^n \times \{0,1\}^k \mapsto \{0,1\}^n$, for some fixed $k > 0$. The IV of the hash function is an arbitrary constant h_0, introduced for considering collisions with the empty string. Furthermore, we introduce the following definition:

Definition 1 (Dither sequence). *A* dither sequence *is defined by a triplet $(\mathcal{I}, \mathcal{D}, d)$, where $\mathcal{I} \subseteq \mathbb{N}$ is the set of iteration indexes, $\mathcal{D} \subseteq \{0,1\}^k$ is the set of valid dither values, and d is a function $\mathcal{I} \mapsto \mathcal{D}$ returning the dither value corresponding to an iteration index.*

This definition is independent of the particular input method, and is relevant for any iterated hash function. In the remainder, we let $\delta = |\mathcal{D}| \leq 2^k$, and write d for a dither value[3].

Definition 2 (Collision for Dithered Compression Function). *The advantage of $\mathcal{A}$ in finding a collision in f is*

$$\mathbf{Adv}_f^{\mathsf{col}}(\mathcal{A}) = \Pr\left[\begin{array}{l|l} E \xleftarrow{\$} \mathsf{Bloc}(\kappa,\mu), & (h,m,d) \neq (h',m',d),\ d \in \mathcal{D}, \\ (h,m,d,h',m') \xleftarrow{\$} \mathcal{A} & \big[f^E(h,m,d) = f^E(h',m',d) \\ & \textit{or } f^E(h,m,d) = h_0\big] \end{array}\right].$$

This notion of collision can be viewed as a variant of target-collision resistance [34], where the key indexing the function is chosen by the attacker.

Definition 3 (Δ-Collision for Dithered Compression Functions). *The advantage of $\mathcal{A}$ in finding a Δ-collision in f is*

$$\mathbf{Adv}_f^{\Delta\mathsf{col}}(\mathcal{A}) = \Pr\left[\begin{array}{l|l} E \xleftarrow{\$} \mathsf{Bloc}(\kappa,\mu), & (h,m,d) \neq (h',m',d'),\ (d,d') \in \mathcal{D}^2, \\ (h,m,d,h',m',d') \xleftarrow{\$} \mathcal{A} & \big[f^E(h,m,d) = f^E(h',m',d') \\ & \textit{or } f^E(h,m,d) = h_0\big] \end{array}\right].$$

[3]The notation $\Pr[\alpha|\beta]$ stands here for the probability of the event β *after* the experiment α. This should not be confused with the notation of conditional probabilities.

The notion capturing one-wayness is termed as "inversion" rather than "preimage", merely because of the different sampling rule for the challenge image (see [9, Ap. B] for a discussion).

Definition 4 (Inversion for Dithered Compression Function). *The advantage of $\mathcal{A}$ in inverting f is*

$$\mathbf{Adv}^{\mathsf{inv}}_{f}(\mathcal{A}) = \Pr\left[\begin{array}{l|l} E \xleftarrow{\$} \mathsf{Bloc}(\kappa,\mu),\ h^{+} \xleftarrow{\$} \mathsf{Range}(f^{E}), & d \in \mathcal{D}, \\ (h,m,d) \xleftarrow{\$} \mathcal{A} & f(h,m,d) = h^{+} \end{array}\right].$$

Let **Adv** be any of the advantages defined above. For $q \geq 0$, we write $\mathbf{Adv}(q) = \max_{\mathcal{A}}(\mathbf{Adv}(\mathcal{A}))$, where the maximum is taken over all adversaries making at most q oracle queries. The definitions for hash functions apply as well for dithered hash functions, where a random blockcipher is used, and a given dither sequence is considered.

4 Collision Resistance

4.1 Blackbox Bounds

Theorem 1 (Collision Resistance of dPGV Hash Functions). *Let H be a hash function built on a dithered PGV scheme $f \notin \mathcal{C}_2$, where MD-strengthening is applied. Then the best advantage for a q-bounded adversary in finding collisions is*

$$\mathbf{Adv}^{\mathsf{col}}_{H}(q) \leq \frac{q(q+1)}{2^{n}},\ \textit{for } f \in \mathcal{C}_1,$$
$$\mathbf{Adv}^{\mathsf{col}}_{H}(q) \leq \frac{(\delta^2+\delta)(q^2+q)}{2^{n+1}},\ \textit{for } f \in \mathcal{C}_3^{+},$$

where $\delta = |\mathcal{D}|$ is the number of valid dither values.

This gives for $\mathcal{C}_1$ schemes a bound on collision resistance independent on the dither sequence used. But for $\mathcal{C}_3$ schemes the bound depends on the size of the dither domain $\mathcal{D}$: Clearly, when $\mathcal{D}$ is large (e.g. when $\delta = |\mathcal{D}| = 2^n$) this bound is not relevant. However, it makes sense for example for Rivest's dithering proposal, for which $\delta \leq 2^{15}$.

We prove Theorem 1 by first upper bounding $\mathbf{Adv}^{\mathsf{col}}_{H}$ by a collision-finding advantage for the compression function (see Lemma 1), then bounding this advantage in the blackbox model (Propositions 1 and 2).

Lemma 1 (Dithered Extension of MD Theorem). *Let H be a hash function built on a dithered PGV scheme $f \in \mathcal{C}_1 \cup \mathcal{C}_3^{+}$ and using MD-strengthening. Then $\mathbf{Adv}^{\mathsf{col}}_{H}(q) \leq \mathbf{Adv}^{\Delta\mathsf{col}}_{f}(q)$. Furthermore, if $f \in \mathcal{C}_1$, then $\mathbf{Adv}^{\mathsf{col}}_{H}(q) \leq \mathbf{Adv}^{\mathsf{col}}_{f}(q)$.*

This lemma states that the security of the hash function built on a dPGV scheme can be reduced to the security of its compression function, except for $\mathcal{C}_2$ functions. We show later a counter-example of $\mathcal{C}_2$-based hash functions which are not collision resistant, despite having a collision-resistant compression function.

Proof. Assume given an arbitrary colliding pair $(\bar{m}, \bar{m}')$ for a H^E with random $E \in \mathsf{Bloc}(n, n)$, and set $\ell = |\bar{m}|$, $\ell' = |\bar{m}'|$ (in blocks). We distinguish two cases:

1. $\ell = \ell'$: If $m_\ell \neq m'_\ell$ or $h_{\ell-1} \neq h'_{\ell-1}$, then we get a collision on f with the distinct tuples $(h_{\ell-1}, m_\ell, d_\ell)$ and $(h'_{\ell-1}, m'_\ell, d_\ell)$; otherwise, $h_{\ell-1} = h'_{\ell-1}$ and $m_\ell = m'_\ell$; we then work inductively with the same argument backwards until a collision is found, which necessarily exists, because $\bar{m} \neq \bar{m}'$ by hypothesis. Therefore, $\mathbf{Adv}^{\mathsf{col}}_H(q) \leq \mathbf{Adv}^{\mathsf{col}}_f(q)$ for messages of same length.
2. $\ell \neq \ell'$: Since MD-strengthening is applied, we have $m_\ell \neq m'_{\ell'}$, that necessarily leads to distinct Δ-colliding tuples for f with distinct message block, thus $\mathbf{Adv}^{\mathsf{col}}_H(q) \leq \mathbf{Adv}^{\Delta\mathsf{col}}_f(q)$. If $d_\ell = d_{\ell'}$, we even get a collision on f (with same dither value). Furthermore, for $\mathcal{C}_1$ functions, the pairs $(h_{\ell-1} \oplus d_\ell, m_\ell)$ and $(h_{\ell'-1} \oplus d_{\ell'}, m_{\ell'})$ form a collision for the original (undithered) scheme, hence $\mathbf{Adv}^{\mathsf{col}}_H(q) \leq \mathbf{Adv}^{\mathsf{col}}_f(q)$ in this case.

This covers all possible cases, showing reductions to the security of the dithered compression function f, which completes the proof. □

For functions of $\mathcal{C}_2$, the advantage cannot be bounded by $\mathbf{Adv}^{\mathsf{col}}_f(q)$ since the case $m_i \oplus d_i = m'_j \oplus d'_j$ may occur, for $d_i \neq d'_j$, which does not necessarily lead to a collision on the original undithered scheme. To prove such inequality, one should add the assumption that the dither and the message length padded in the last block do not overlap; e.g. consider the dither coded on the $n/2$ first bits of the blocks, while at most $n/2$ bits are dedicated to encoding the message length.

Proposition 1 (Collision Resistance of dPGV Schemes). *Let f be a dithered PGV scheme. Then the best advantage of a q-bounded adversary in finding collisions in f is $\mathbf{Adv}^{\mathsf{col}}_f(q) \leq q(q+1)/2^n$.*

Proof. For ease of exposition, consider the dithered MMO schemes, instantiated with a random E: From arbitrary colliding inputs (h, m, d) and (h', m', d) with image h^+, we can construct colliding inputs $(h_\star, m_\star)$ and $(h'_\star, m'_\star)$ for the original undithered scheme as follows:

	$h_\star$	$h'_\star$	$m_\star$	$m'_\star$	$h^+_\star$
dMMO_1	$h \oplus d$	$h' \oplus d$	m	m'	h^+
dMMO_2	h	h'	$m \oplus d$	$m' \oplus d$	$h^+ \oplus d$
dMMO_3	h	h'	m	m'	$h^+ \oplus d$
dMMO_4	h	h'	$m \oplus d$	$m' \oplus d$	h^+

A similar method applies for all dithered PGV schemes. The proposition now follows from the bound $q(q+1)/2$ given in Lemma 3.3 of [9]. □

Proposition 2 (Δ-Collision Resistance of dPGV Schemes). *Let f be a dithered PGV scheme. If $f \in \mathcal{C}_1 \cup \mathcal{C}_2$, then the best advantage of a q-bounded adversary in finding Δ-collisions in f is $\mathbf{Adv}_f^{\Delta\mathsf{col}}(q) = 1$. If $f \in \mathcal{C}_3^+$, then $\mathbf{Adv}_f^{\Delta\mathsf{col}}(q) \leq (\delta^2 + \delta)(q^2 + q)/2^{n+1}$.*

The idea of the proof of Proposition 2 for $f \in \mathcal{C}_3^+$ is similar to the one of the proof of [9, Lemma 3.3]. In short, we first show that any collision for a $\mathcal{C}_3^+$ scheme can be used to find values $(x_r, k_r, E_{k_r}(x_r))$ and $(x_s, k_s, E_{k_s}(x_s))$ satisfying a particular relationship, then we bound the cost of finding such values. The proof strategy is fairly standard. The simulator used for E and E^{-1} is described in [9, Fig. 4].

Proof. For $f \in \mathcal{C}_1 \cup \mathcal{C}_2$, we simply show how to construct a collision: For $f \in \mathcal{C}_1$, pick an arbitrary triplet (h, m, d) such that $d \in \mathcal{D}$. Then construct (h', m', d') by choosing an arbitrary $d' \in \mathcal{D}$ distinct from d, and setting $h' = h \oplus d \oplus d'$, $m' = m$. For $f \in \mathcal{C}_2$, a similar method can be applied with $h' = h$, and $m' = m \oplus d \oplus d'$. In both cases the constructed pairs map to the same image.

For $f \in \mathcal{C}_3^+$, we just give the proof for MMO dithered variants (a similar one can easily be derived for any $\mathcal{C}_3^+$ scheme): First, observe that dMMO_2 ($\in \mathcal{C}_5$) and dMMO_3 ($\in \mathcal{C}_3$) are in $\mathcal{C}_3^+$, while $\mathsf{dMMO}_1, \mathsf{dMMO}_4, \mathsf{dMMO}_5$ are not in $\mathcal{C}_3^+$. Hence, the proof considers only dMMO_2 and dMMO_3.

Then, observe that for both dMMO_2 and dMMO_3 finding a Δ-Collision is *equivalent* to finding a tuple $(h, h', m, m', \tilde{d})$ such that $(E_h(m) \oplus E_{h'}(m') \oplus m \oplus m') \in \mathcal{D}_\oplus$, for

$$\mathcal{D}_\oplus = \left\{ \tilde{d} = d \oplus d', (d, d') \in \mathcal{D}^2 \right\}, \text{ and } \delta_\oplus = |\mathcal{D}_\oplus| .$$

Indeed, for such a tuple $(h, h', m, m', \tilde{d} = d \oplus d')$, we have that

- (h, m, d) and (h', m', d') form a collision for dMMO_3, because
$$E_h(m) \oplus m \oplus d = E_{h'}(m') \oplus m \oplus d'$$
- $(h, m \oplus d, d)$ and $(h', m' \oplus d', d')$ form a collision for dMMO_2, because
$$E_h((m \oplus d) \oplus d) \oplus (m \oplus d) = E_{h'}((m' \oplus d') \oplus d') \oplus (m' \oplus d')$$

We have thus shown that for any Δ-collision for dMMO_2 (or dMMO_3), one can return two triplets (x_r, k_r, y_r) and (x_s, k_s, y_s) such that $x_r \oplus x_s \oplus y_r \oplus y_s \in \mathcal{D}_\oplus$ and $y_r = E_{k_r}(x_r)$, $y_s = E_{k_s}(x_s)$. Using arguments similar to [9, Lemma 3.3 proof], we will show that this event is unlikely.

As preliminaries, consider an adversary $\mathcal{A}$ making q queries (to E or E^{-1}) and who gets q triplets (x_i, k_i, y_i), such that $y_i = E_{k_i}(x_i)$, $i = 1, \ldots, q$. These triplets are constructed by the simulator described in [9, Fig. 4]. Following these notations, $\mathcal{A}$ succeeds only if there exists distinct r, s such that $(x_r \oplus x_s \oplus y_r \oplus y_s) \in \mathcal{D}_\oplus$, or $x_r \oplus y_r = h_0$.

Now, in the process of simulating E (and E^{-1}) we let C_i stand for the event "$x_i \oplus y_i = h_0$ or there exists $j < i$ such that $(x_i \oplus x_j \oplus y_i \oplus y_j) \in \mathcal{D}_\oplus$"; in other words, this is the event "$\mathcal{A}$ succeeds".

The probabilistic argument is that, depending on the oracle queried, either y_i or x_i was (uniformly) randomly selected from a set of size $\geq 2^n - (i-1)$ (see the definition of the simulator in [9, Fig. 4]). Hence $\Pr[\mathsf{C}_i] \leq i \cdot \delta_\oplus / (2^n - (i-1))$, because there are $\delta_\oplus = |\mathcal{D}_\oplus|$ values of $(x_i \oplus x_j \oplus y_i \oplus y_j)$ for which $\mathcal{A}$ succeeds.

It follows that for a number of queries $q \leq 2^{n-1}$,

$$\mathbf{Adv}_f^{\Delta\mathsf{col}}(q) \leq \Pr[\mathsf{C}_1 \vee \cdots \vee \mathsf{C}_q] \leq \sum_{0<i\leq q} \frac{i \cdot \delta_\oplus}{(2^n - i + 1)} \leq \frac{\delta_\oplus}{2^n - 2^{n-1}} \sum_{0<i\leq q} i,$$

that is, $\mathbf{Adv}_f^{\Delta\mathsf{col}}(q) \leq \delta_\oplus \cdot q(q+1)/2^n \leq (\delta^2 + \delta)(q^2 + q)/2^{n+1}$.

We have proven the bound when f is dMMO_2 or dMMO_3. As suggested in §2, a similar proof can be given for any $f \in \mathcal{C}_3^+$: the only difference will be in the conversion of the tuple $(h, h', m, m', \tilde{d} = d \oplus d')$ for which $(E_h(m) \oplus E_{h'}(m') \oplus m \oplus m') \in \mathcal{D}_\oplus$ to a collision for f. For instance, consider $f_{2,5}(h, m, d) = E_h(m \oplus h \oplus d) \oplus m \oplus h$; from the tuple above we can construct the collision

$$f_{2,5}(h, m \oplus h \oplus d) = E_h(m) \oplus m \oplus d = f_{2,5}(h', m' \oplus h' \oplus d', d').$$

Similar conversion can be given for the other $\mathcal{C}_3^+$ schemes. The rest of the proof is then independent of the scheme considered, hence apply as well to any $f \in \mathcal{C}_3^+$.

□

Proof (Theorem 1). The result follows directly from Lemma 1 and the bounds given in Propositions 1 and 2. □

4.2 Finding Collisions for $\mathcal{C}_2$ Hash Functions

We describe an attack for 8 of the 12 schemes of $\mathcal{C}_2$ (namely $f_{5,2}, \ldots, f_{12,2}$), when the dither sequence scheme is the one of HAIFA [6], i.e. where d_i is the number of message bits hashed so far, and when MD-strengthening is applied. The attack exploits the structure of the compression function, and computes a pair of message colliding for any choice of a blockcipher.

The method is inspired from slide attacks on blockciphers [7]: Consider an arbitrary message $\bar{m} = (m_1, \ldots, m_\ell)$, split into ℓ blocks, with $\{d_i\}_{0<i\leq\ell}$ the dither sequence. Compute the fixed-point $h_0 = h_1$ corresponding to m_1, and construct the message $\bar{m}' = (m'_1, \ldots, m'_{\ell-1})$ by setting $m'_i = m_{i+1} \oplus d_{i+1} \oplus d'_i$, for $i = 1, \ldots, \ell - 2$. The last message blocks m_ℓ and $m'_{\ell-1}$ have to follow the MD-strengthening rule, that is, having the number of bits of the message coded in their least significant bits. Since we consider a dither sequence coding the number of message bits hashed so far, the padded values will be equal to d_ℓ and $d_{\ell-1}$, respectively. Therefore $m'_{\ell-1} = m_\ell \oplus d_\ell \oplus d_{\ell-1}$ is a valid last block, and we end up with $h_\ell = h'_{\ell-1}$, giving a collision with $\bar{m}$. Note that no call to the compression is needed, nor to the blockcipher.

When MD-strengthening is not used, this technique can be applied for any dither sequence, for the 8 schemes $f_{5,2}, \ldots, f_{12,2}$. This concerns for instance

Rivest's dithering, that uses a special dither for the last block instead of MD-strengthening. The fact that a secure compression function (i.e. a provably secure PGV scheme) can lead to a weak hash function contrasts with the result of Black et al. where certain weak compression functions (namely non-preimage-resistant) are shown to provide collision-resistant hash functions.

5 Inversion Resistance

Theorem 2 below holds for inverting a random range point, rather than the image of a random domain point (the latter problem being refered as "preimage"). Quoting [9, Ap. B], though these measures *"can, in general, be far apart, it is natural to guess that they coincide for 'reasonable' hash functions"*.

Theorem 2 (Inversion Resistance of dPGV Hash Functions). *Let H be a hash function built on a dithered PGV scheme f, where MD-strengthening is applied. Then the best advantage of a q-bounded adversary in inverting H is*

$$\mathbf{Adv}^{\mathsf{inv}}_{H}(q) \leq \frac{q}{2^{n-1}}, \text{ for } f \in \mathcal{C}_1 \cup \mathcal{C}_2,$$
$$\mathbf{Adv}^{\mathsf{inv}}_{H}(q) \leq \frac{\delta \cdot q}{2^{n-1}}, \text{ for } f \in \mathcal{C}_3^{+}.$$

Proposition 3 (Inversion Resistance of dPGV Schemes). *Let f be a dithered PGV scheme. Then the best advantage of a q-bounded adversary in inverting f is $\mathbf{Adv}^{\mathsf{inv}}_{f}(q) \leq \delta \cdot q/2^{n-1}$. Furthermore, if $f \in \mathcal{C}_1 \cup \mathcal{C}_2$, then $\mathbf{Adv}^{\mathsf{inv}}_{f}(q) \leq q/2^{n-1}$.*

Proof. For $\mathcal{C}_1$ and $\mathcal{C}_2$, just observe that a preimage oracle for the $\mathcal{C}_1$ or $\mathcal{C}_2$ version of a PGV scheme can be used to solve the preimage problem for the original scheme, whose bound is $q/2^{n-1}$, from [9].

For $\mathcal{C}_3^{+}$, the problem is equivalent to finding h and m such that $(F(h,m) \oplus h^{+}) \in \mathcal{D}$, with F the original (undithered) scheme, for a fixed h^{+}. This equation is satisfied for a random permutation and arbitrary h, m, h^{+} with probability $\delta/2^{n}$. We can use the same strategy as for proving Proposition 1: e.g. for $\mathsf{dMMO}_3 \in \mathcal{C}_3$, let C_i be the event "the i-th query (x_i, k_i, y_i) satisfies $(y_i \oplus x_i \oplus h^{+}) \in \mathcal{D}$", $i \in \{1, \ldots, q\}$; then we have $\Pr[\mathsf{C}_i] \leq \delta/(2^{n} - (i-1))$. By the union bound, we get

$$\mathbf{Adv}^{\mathsf{inv}}_{f}(q) \leq \Pr[\mathsf{C}_1 \vee \cdots \vee \mathsf{C}_q] \leq \frac{\delta \cdot q}{2^{n-1}}.$$

□

Proof (Theorem 2). An oracle inverting H can be trivially used for inverting its dithered compression function. The result of the theorem then follows from Proposition 3. □

6 Conclusions

Among the 56 dPGV schemes studied,

- 12 inherit the bounds on collision and inversion resistance of the the original (undithered) constructions, independently of the dither sequence considered (these are of the form $f_{i,1}$, $i \in [1, 12]$)
- 37 have the "fixed-point" attribute ($f_{i,j}$, $i \in [5, 12]$)
- 8 lead to weak hash functions for HAIFA's dithering ($f_{i,2}$, $i \in [5, 12]$)

It appears that the most reliable schemes have the dither value simply xored with the initial chaining value h (subset $\mathcal{C}_1$). Nevertheless, the schemes of $\mathcal{C}_2$ fail to achieve similar security just because the overlap of dither and padding might allow collisions, for particular dither sequence. This problem can be easily avoided in practice, e.g. by encoding the dither in big-endian, and the message-dependent padding in little-endian, such that the two values do not overlap. In this case, $\mathcal{C}_2$ becomes as secure as $\mathcal{C}_1$, with the added benefit that it requires no change in the implementation of the hash function (whereas all other $\mathcal{C}_i$'s do).

Another desirable property concerns all 56 schemes considered: The fact that efficiency is no longer affected by the length of dither values allows to use a large counter, which provides better protection against attacks as [17, 20, 24] than schemes with short dithers [12]. An additional feature which can be derived from our constructions is *randomized hashing*, e.g. by choosing a random starting point for the counter. This would avoid extra changes to the compression function, like the RMX transform [21].

Eventually, we stress that dithering not only protects against generic short-cut attacks for second-preimage—which we might live with, since they remain much slower than collision search—but also provides a safety net against more elaborate attacks, and is expected to complicate some existing dedicated attacks.

Further work may consider the existence of generic second-preimage attacks for dPGV schemes instantiated with particular dither sequences, as well as refinement of our proofs at the light of Stam's recent improvements [46].

Acknowledgments

Particular thanks to Africacrypt anonymous referee #4 for his/her constructive criticism, and to referees #1, #5 and #6 for their pointing out several ways to improve readability.

References

1. Elena Andreeva, Charles Bouillaguet, Pierre-Alain Fouque, Jonathan Hoch, John Kelsey, Adi Shamir, and Sebastien Zimmer. Second preimage attacks on dithered hash functions. In Nigel Smart, editor, *EUROCRYPT*, LNCS. Springer, 2008. To appear.
2. Paulo Barreto and Vincent Rijmen. The Whirlpool hashing function. First Open NESSIE Workshop, 2000.

3. Daniel J. Bernstein. The Rumba20 compression function. `http://cr.yp.to/rumba20.html`. Function introduced in [4].
4. Daniel J. Bernstein. What output size resists collisions in a xor of independent expansions? ECRYPT Workshop on Hash Functions, 2007. See `http://cr.yp.to/rumba20.html#expandxor`.
5. Eli Biham. Recent advances in hash functions - the way to go, 2005. Presented at the ECRYPT Hash Function Workshop, 2005.
6. Eli Biham and Orr Dunkelman. A framework for iterative hash functions - HAIFA. Cryptology ePrint Archive, Report 2007/278, 2007. Previously presented at the second NIST Hash Function Workshop, 2006.
7. Alex Biryukov and David Wagner. Slide attacks. In Lars R. Knudsen, editor, *FSE*, volume 1636 of *LNCS*, pages 245–259. Springer, 1999.
8. John Black, Martin Cochran, and Thomas Shrimpton. On the impossibility of highly-efficient blockcipher-based hash functions. In Cramer [15], pages 526–541.
9. John Black, Phillip Rogaway, and Thomas Shrimpton. Black-box analysis of the block-cipher-based hash-function constructions from PGV. Cryptology ePrint Archive, Report 2002/066, 2002. Full version of [10].
10. John Black, Phillip Rogaway, and Thomas Shrimpton. Black-box analysis of the block-cipher-based hash-function constructions from PGV. In Moti Yung, editor, *CRYPTO*, volume 2442 of *LNCS*, pages 330–335. Springer, 2002.
11. Dan Boneh and Xavier Boyen. On the impossibility of efficiently combining collision resistant hash functions. In Dwork [18], pages 570–583.
12. Charles Bouillaguet, Pierre-Alain Fouque, Adi Shamir, and Sebastien Zimmer. Second preimage attacks on dithered hash functions. Cryptology ePrint Archive, Report 2007/395, 2007. See also [1].
13. Gilles Brassard, editor. *Advances in Cryptology - CRYPTO '89, 9th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 1989, Proceedings*, volume 435 of *LNCS*. Springer, 1990.
14. Donghoon Chang, Kishan Chand Gupta, and Mridul Nandi. RC4-Hash: A new hash function based on RC4. In Rana Barua and Tanja Lange, editors, *INDOCRYPT*, volume 4329 of *LNCS*, pages 80–94. Springer, 2006.
15. Ronald Cramer, editor. *Advances in Cryptology - EUROCRYPT 2005, 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22-26, 2005, Proceedings*, volume 3494 of *LNCS*. Springer, 2005.
16. Ivan Damgård. A design principle for hash functions. In Brassard [13], pages 416–427.
17. Richard Drews Dean. *Formal Aspects of Mobile Code Security.* PhD thesis, Princeton University, 1999.
18. Cynthia Dwork, editor. *Advances in Cryptology - CRYPTO 2006, 26th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 2006, Proceedings*, volume 4117 of *LNCS*. Springer, 2006.
19. Decio Gazzoni Filho, Paulo Barreto, and Vincent Rijmen. The Maelstrom-0 hash function. In *6th Brazilian Symposium on Information and Computer Security*, 2006.
20. Praveen Gauravaram and John Kelsey. Cryptanalysis of a class of cryptographic hash functions. Cryptology ePrint Archive, Report 2007/277, 2007.
21. Shai Halevi and Hugo Krawczyk. Strengthening digital signatures via randomized hashing. In Dwork [18], pages 41–59.

22. Antoine Joux. Multicollisions in iterated hash functions. application to cascaded constructions. In Matthew K. Franklin, editor, *CRYPTO*, volume 3152 of *LNCS*, pages 306–316. Springer, 2004.
23. John Kelsey and Tadoyoshi Kohno. Herding hash functions and the Nostradamus attack. First NIST Cryptographic Hash Function Workshop, 2005.
24. John Kelsey and Bruce Schneier. Second preimages on n-bit hash functions for much less than 2^n work. In Cramer [15], pages 474–490.
25. Lars Knudsen. Hash functions and SHA-3. Invited talk at FSE 2008.
26. Lars Knudsen and Vincent Rijmen. Known-key distinguishers for some block ciphers. In Kaoru Kurosawa, editor, *ASIACRYPT*, volume 4833 of *LNCS*, pages 315–324. Springer, 2007.
27. Lars R. Knudsen, Christian Rechberger, and Søren S. Thomsen. The Grindahl hash functions. In Alex Biryukov, editor, *FSE*, volume 4593 of *LNCS*, pages 39–57. Springer, 2007.
28. Xuejia Lai and James Massey. Hash function based on block ciphers. In Rainer A. Rueppel, editor, *EUROCRYPT*, volume 658 of *LNCS*, pages 55–70. Springer, 1992.
29. Wonil Lee, Mridul Nandi, Palash Sarkar, Donghoon Chang, Sangjin Lee, and Kouichi Sakurai. PGV-style block-cipher-based hash families and black-box analysis. *IEICE Transactions*, 88-A(1):39–48, 2005.
30. Stephen Matyas, Carl Meyer, and Jonathan Oseas. Generating strong one-way functions with cryptographic algorithm. *IBM Technical Disclosure Bulletin*, 27(10A):5658–5659, 1985.
31. Ralph C. Merkle. One way hash functions and DES. In Brassard [13], pages 428–446.
32. Ilya Mironov. Hash functions: From Merkle-Damgård to Shoup. In Birgit Pfitzmann, editor, *EUROCRYPT*, volume 2045 of *LNCS*, pages 166–181. Springer, 2001.
33. Shoji Miyaguchi, Kazuo Ohta, and Masahiko Iwata. New 128-bit hash function. In *4th International Joint Workshop on Computer Communications*, pages 279–288, 1989.
34. Moni Naor and Moti Yung. Universal one-way hash functions and their cryptographic applications. In *STOC*, pages 33–43. ACM, 1989.
35. Krzysztof Pietrzak. Non-trivial black-box combiners for collision-resistant hash-functions don't exist. In Moni Naor, editor, *EUROCRYPT*, volume 4515 of *LNCS*, pages 23–33. Springer, 2007.
36. Ken Pohlmann. *Principles of Digital Audio*. McGraw-Hill, fourth edition edition, 2005.
37. Bart Preneel, Antoon Bosselaers, René Govaerts, and Joos Vandewalle. Collision-free hash functions based on block cipher algorithms. In *Carnahan Conference on Security Technology*, pages 203–210, 1989.
38. Bart Preneel, René Govaerts, and Joos Vandewalle. Hash functions based on block ciphers: A synthetic approach. In Douglas R. Stinson, editor, *CRYPTO*, volume 773 of *LNCS*, pages 368–378. Springer, 1993.
39. Jean-Jacques Quisquater and Marc Girault. 2n-bit hash-functions using n-bit symmetric block cipher algorithms. In *EUROCRYPT*, pages 102–109, 1989.
40. Michael Rabin. Digitalized signatures. In Richard Lipton and Richard DeMillo, editors, *Foundations of Secure Computation*, pages 155–166. Academic Press, 1978.
41. Ronald Rivest. Abelian square-free dithering for iterated hash functions. ECRYPT Workshop on Hash Functions, 2005. Also presented in [42].
42. Ronald Rivest. Abelian square-free dithering for iterated hash functions. NIST Hash Function Workshop, 2005.

43. Philip Rogaway and John Steinberger. Security/efficiency tradeoffs for permutation-based hashing. In Nigel Smart, editor, *EUROCRYPT*, LNCS. Springer, 2008. To appear.
44. Victor Shoup. A composition theorem for universal one-way hash functions. In Bart Preneel, editor, *EUROCRYPT*, volume 1807 of *LNCS*, pages 445–452. Springer, 2000.
45. Thomas Shrimpton and Martijn Stam. Building a collision-resistant compression function from non-compressing primitives. Cryptology ePrint Archive, Report 2007/409, 2007.
46. Martijn Stam. Another glance at blockcipher based hashing. Cryptology ePrint Archive, Report 2008/071, 2008.
47. Wikipedia. Dither — Wikipedia, The Free Encyclopedia, 2007. Accessed 22-November-2007.

A Definitions

Definition 5 (Collision for Hash Functions). *Let H be a blockcipher-based hash function. The advantage of $\mathcal{A}$ in finding collisions in H is*

$$\mathbf{Adv}^{\mathsf{col}}_{H}(\mathcal{A}) = \Pr\left[\begin{array}{l|l} E \xleftarrow{\$} \mathsf{Bloc}(\kappa, n), & \bar{m} \neq \bar{m}', \\ (\bar{m}, \bar{m}') \xleftarrow{\$} \mathcal{A} & H^E(\bar{m}) = H^E(\bar{m}') \end{array}\right].$$

Definition 6 (Collision for Compression Functions). *Let f be a blockcipher-based compression function. The advantage of $\mathcal{A}$ in finding collisions in f is*

$$\mathbf{Adv}^{\mathsf{col}}_{f}(\mathcal{A}) = \Pr\left[\begin{array}{l|l} E \xleftarrow{\$} \mathsf{Bloc}(\kappa, \mu), & (h, m) \neq (h', m'), \\ (h, m, h', m') \xleftarrow{\$} \mathcal{A} & \big[f^E(h, m) = f^E(h', m') \\ & \textit{or } f^E(h, m) = h_0\big] \end{array}\right].$$

B Near-Collisions for dPGV and PGV Schemes

For the 32 schemes of C_3^+, which can be rewritten as $f(h, m, d) = F(h, m) \oplus d$, near-collisions might be easily found, depending on the structure of $\mathcal{D}$: suppose there exists $d, d' \in \mathcal{D}$ such that $d \oplus d'$ has weight w. Then, $f(h, m, d) \oplus f(h, m, d') = d \oplus d'$ and has weight w as well. This trivial property seems not to imply any weakness on the hash functions, since an adversary has no freedom on choosing the dither value for a given iteration count.

In the "known-key" scenario for blockciphers, Knudsen and Rijmen [26] presents a distinguisher for 7-round Feistel blockciphers based on the finding messages m, m' such that $E_k(m) \oplus m \oplus E_k(m') \oplus m' = 0 \ldots 0 \| x$, where x is a random $n/2$-bit value. As they observe, it can be applied to find "half-collisions" on MMO (f_1) instantiated with a similar blockcipher; indeed, MMO sets $f(h, m) = E_h(m) \oplus m$, thus one can choose a h which shall play the role of the "known-key" (note that in [26] the key cannot be chosen). We observe that a similar method can be applied to the other PGV schemes $f_2, \ldots, f_8$ (for some of them, by conveniently choosing the null value for h or m).

C Origins of Dithering

The use of the term "dither" in the context of hash functions finds its origin in signal processing, which itself borrowed it from engineers, who adapted the ancient word "didder" to a mechanical problem. The three quotes below give a bit more details about this story.

Quoting Rivest [41]: *"The word 'dithering' derives from image-processing, where a variety of gray or colored values can be represented by mixing together pixels of a small number of basic shades or colors; this is done in a random or pseudo-random manner to prevent simple visual patterns from being visible. We adapt the term dithering here to refer to the process of adding an additional 'dithering' input to a sequence of processing steps, to prevent an adversary from causing and exploiting simple repetitive patterns in the input."*

Quoting Wikipedia [47]: *"Dither is an intentionally applied form of noise, used to randomize quantization error, thereby preventing large-scale patterns such as contouring that are more objectionable than uncorrelated noise. (...) Dither most often surfaces in the fields of digital audio and video, where it is applied to rate conversions and (usually optionally) to bit-depth transitions; it is utilized in many different fields where digital processing and analysis is used—especially waveform analysis."*

Quoting Pohlman [36]: *"one of the earliest [applications] of dither came in World War II. Airplane bombers used mechanical computers to perform navigation and bomb trajectory calculations. Curiously, these computers (boxes filled with hundreds of gears and cogs) performed more accurately when flying on board the aircraft, and less well on ground. Engineers realized that the vibration from the aircraft reduced the error from sticky moving parts. Instead of moving in short jerks, they moved more continuously. Small vibrating motors were built into the computers, and their vibration was called 'dither' from the Middle English verb 'didderen,' meaning 'to tremble.' Today, when you tap a mechanical meter to increase its accuracy, you are applying dither, and modern dictionaries define 'dither' as 'a highly nervous, confused, or agitated state.' In minute quantities, dither successfully makes a digitization system a little more analog in the good sense of the word."*