On Security Arguments of the Second Round SHA-3 Candidates

Elena Andreeva · Andrey Bogdanov · Bart Mennink · Bart Preneel · Christian Rechberger

March 19, 2012

Abstract In 2007, the US National Institute for Standards and Technology (NIST) announced a call for the design of a new cryptographic hash algorithm in response to vulnerabilities like differential attacks identified in existing hash functions, such as MD5 and SHA-1. NIST received many submissions, 51 of which got accepted to the first round. 14 candidates were left in the second round, out of which 5 candidates have been recently chosen for the final round. An important criterion in the selection process is the SHA-3 hash function security. We identify two important classes of security arguments for the new designs: (1) the possible reductions of the hash function security to the security of its underlying building blocks, and (2) arguments against differential attack on building blocks. In this paper, we compare the state of the art provable security reductions for the second round candidates, and review arguments and bounds against classes of differential attacks. We discuss all the SHA-3 candidates at a high functional level, analyze and summarize the security reduction results and bounds against differential attacks. Additionally, we generalize the well-known proof of collision resistance preservation, such that all SHA-3 candidates with a suffix-free padding are covered.

Keywords SHA-3 competition · hash functions · classification · security reductions · differential attacks

A preliminary version of this paper appeared at ISC 2010 [2].

E. Andreeva · A. Bogdanov · B. Mennink · B. Preneel
Dept. Electrical Engineering, ESAT/COSIC and IBBT
Katholieke Universiteit Leuven, Belgium
Tel.: +32-16-321800
Fax: +32-16-321969
E-mail: bart.mennink@esat.kuleuven.be

C. Rechberger
Institut for Matematik
Danmarks Tekniske Universitet, Denmark

1 Introduction

Hash functions serve a crucial foundation of numerous cryptographic applications, ranging from key derivation functions over various security protocols to digital signatures. In 2004, a series of differential attacks by Wang et al. [81, 82] have exposed security vulnerabilities in the design of the most widely adopted and deployed SHA-1 hash function. As a result, the US National Institute for Standards and Technology (NIST) recommended the replacement of SHA-1 by the SHA-2 hash function family and announced a call for the design of a new SHA-3 hashing algorithm. The SHA-3 hash function must allow for message digests of length $224, 256, 384$ and 512 bits, it should be efficient, and most importantly it should provide an adequate level of security. In the second round, 14 candidate hash functions were considered, 5 candidates remaining in the race for the selection of the SHA-3 hash function in the current third round.

As a result of the performed comparative analysis, several classifications of the SHA-3 candidates, mostly focused on hardware performance, appeared in the literature [43, 46, 59, 79]. A classification based on the NIST-specified security criteria is however still due.

NIST Security Requirements. NIST specifies a number of security requirements [70] to be satisfied by the future SHA-3 function: `(i)` at least one variant of the hash function must securely support HMAC and randomized hashing. Furthermore, for all n-bit digest values, the hash function must provide `(ii)` preimage resistance of approximately n bits, `(iii)` second preimage resistance of approximately $n-L$ bits, where the first preimage is of length at most 2^L blocks, `(iv)` collision resistance of approximately $n/2$ bits, and `(v)` all variants must be resistant to the length-extension attack. Finally, `(vi)` for any $m \leq n$, the hash function specified by

taking a fixed subset of m bits of the function's output is required to satisfy properties (ii)-(v) with n replaced by m.

Our Contribution. In this work we provide a survey of the 14 second round SHA-3 candidates including the 5 finalists, in which we compare their security reductions, and their resistance against differential attacks. Regarding security reductions, we consider preimage, second preimage and collision resistance (security requirements (ii)-(iv)) for the $n = 256$ and $n = 512$ variants. Most of this security analysis is realized in the ideal model, where one or more of the underlying integral building blocks (e.g., the underlying block cipher or permutation(s)) are assumed to be ideal, i.e. random primitives. To argue collision resistance we extend the standard proof of Merkle-Damgård collision resistance [38, 68] to cover *all* second round SHA-3 candidate hash function with a suffix-free padding (App. B). Notice that the basic Merkle-Damgård proof does not suffice in the presence of a final transformation and/or a chopping. Additionally, we consider the indifferentiability of the candidates. Informally, indifferentiability guarantees that a design has no structural design flaws [32], and in particular (as formally proven in App. A) an indifferentiability result renders upper bounds on the advantage of finding preimages, second preimages and collisions. The provable security analysis in this contribution extends to the findings of Andreeva et al. [2].

Recent substantial progress in the differential attacks against SHA-1 [81] is the main motivation for the SHA-3 competition [29], and many of the first round candidates fell victim to differential attacks as well, due to their well-elaborated and advanced toolbox. Hence we survey known bounds against classes of these attacks. Note that, to the best of current knowledge, informative bounds exist only for a subset of relevant differential properties. Also only a subset of all second round SHA-3 candidates allow for such bounds by design, which excludes e.g. all ARX-based constructions from the analysis. Therefore, in this work, we will be mainly considering upper bounds on the expected differential trail probability (EDTP) of hash function building blocks such as underlying permutations and block ciphers which are based on substitution-permutation networks. Although EDTP proved to be highly informative in predicting the power of differential cryptanalysis for keyed block ciphers, for most attacks on hash functions all inputs to the function's building blocks are known and can often even be chosen by the adversary. However, at some point in most major attack techniques against hash functions, the adversary cannot control the difference propagation any more and relies on the differential trails, EDTP becoming again a good predictor of the attack complexity. Thus, due to the lack of a better universal evaluation tool with respect to differential attacks on the one hand, and because of the ability of the differential trails to reflect at least some important features of such attacks on the other, we have opted to use upper bounds on EDTP in our comparative analysis.

Section 2 briefly covers the notation, and the basic principles of hash function design. In Sect. 3, we consider all candidates, both from a provable security point of view, and from a cryptanalysis view. We give a high level algorithmic description of each hash function, and discuss the existing security results. All results are summarized in Table 1 and Table 2. We conclude the paper with Sect. 4 and give some final remarks on the security comparison.

2 Preliminaries

For a positive integer value $n \in \mathbb{N}$, we denote by $\mathbb{Z}_2^n$ the set of bit strings of length n, and by $(\mathbb{Z}_2^n)^*$ the set of strings of length a positive multiple of n bits. We denote by $\mathbb{Z}_2^*$ the set of bit strings of arbitrary length. If x, y are two bit strings, their concatenation is denoted by $x\|y$. By $|x|$ we denote the length of a bit string x, and for $m, n \in \mathbb{N}$ we denote by $\langle m \rangle_n$ the encoding of m as an n-bit string. The function $\mathsf{chop}_n(x)$ chops off the n rightmost bits of a bit string x.

Throughout, we use a unified notation for all candidates. The value n denotes the output size of the hash function, l the size of the chaining value, and m the number of message bits compressed in one iteration of the compression function. A padded message is always parsed as a sequence of $k \geq 1$ message blocks of length m bits: $(M_1, \ldots, M_k)$.

2.1 Reductionist Security Notions

In this section we investigate the reductionist security of hash functions in the 'ideal model' and the more classical 'generic' security.

Security in the ideal model. In the ideal model, a *compressing* function F (either on fixed or arbitrary input lengths) that uses one or more underlying building blocks is viewed insecure if there exists a successful information-theoretic adversary that has only query access to the idealized underlying primitives of F. The complexity of the attack is measured by the number of queries q to the primitive made by the adversary. In this work it is clear from the context which of the underlying primitives is assumed to be ideal. The three main security properties required from the SHA-3 hash function are preimage, second preimage and collision resistance. For each of these three notions, with $\mathbf{Adv}_F^{\mathrm{atk}}$, where $\mathrm{atk} \in \{\mathrm{pre}, \mathrm{sec}, \mathrm{col}\}$, we denote the maximum advantage of an adversary to break the function F under the security notion atk. The advantage is the probability function taken over all random choices of the underlying primitives, and the

maximum is taken over all adversaries that make at most q queries to their oracles.

Additionally, we consider the indifferentiability of the SHA-3 candidates. The indifferentiability framework introduced by Maurer et al. [67] is an extension of the classical notion of indistinguishability, and ensures that a hash function has no structural defects. We denote the indifferentiability security of a hash function $\mathcal{H}$ by $\mathbf{Adv}^{\text{pro}}_{\mathcal{H}}$, maximized over all distinguishers making at most q queries of maximal length $K \geq 0$ message blocks to their oracles. We refer to [32] for a formal definition. An indifferentiability bound guarantees security of the hash function against specific attacks. In particular, one can obtain a bound on $\mathbf{Adv}^{\text{atk}}_{\mathcal{H}}$, for any security notion atk: $\mathbf{Adv}^{\text{atk}}_{\mathcal{H}} \leq \mathbf{Pr}^{\text{atk}}_{RO} + \mathbf{Adv}^{\text{pro}}_{\mathcal{H}}$, where $\mathbf{Pr}^{\text{atk}}_{RO}$ denotes the success probability of a generic attack against $\mathcal{H}$ under atk. This bound is proven in Thm. 1 (App. A).

Generic security. The *generic collision resistance security* in the context of this work deals with analyzing the collision resistance of hash functions in the standard model. A hash function $\mathcal{H}$ is called generically (t,ε) collision resistant if no adversary running in time at most t can find two different messages M, M' such that $\mathcal{H}(M) = \mathcal{H}(M')$ with advantage more than ε. We denote by $\mathbf{Adv}^{\text{gcol}}_{\mathcal{H}}$ the generic collision resistance security of the function $\mathcal{H}$, maximized over all 'efficient' adversaries. We refer the reader to [4,73,74] for a more formal discussion.

To argue generic collision resistance security of the hash function $\mathcal{H}$ (as domain extenders of fixed input length compression functions) we use the composition result of Merkle and
Damgård [38,68] and extend it to a wider class of suffix-free hash functions (App. B). This result concludes the collision resistance of the hash function $\mathcal{H}$ assuming collision resistance security guarantees from the underlying compression functions. We then translate ideal model collision resistance security results on the compression functions via the latter composition to ideal model collision results on the hash function (expressed by $\mathbf{Adv}^{\text{col}}_{\mathcal{H}}$). A generic collision result, generally speaking, applies to a wider class of schemes for which no bounds on the collision resistance security of the underlying compression functions is known, e.g. for BLAKE and BMW.

If a compressing function F outputs a bit string of length n, one expects to find collisions with high probability after approximately $2^{n/2}$ queries (due to the birthday attack). Similarly, (second) preimages can be found with high probability after approximately 2^n queries[1]. Moreover, finding second preimages is provably harder than finding collisions, and similar for preimages (depending on the specification of F) [74]. Formally, we have $\Omega(q^2/2^n) = \mathbf{Adv}^{\text{col}}_F = O(1)$, $\Omega(q/2^n) = \mathbf{Adv}^{\text{sec}}_F \leq \mathbf{Adv}^{\text{col}}_F$, and $\Omega(q/2^n) = \mathbf{Adv}^{\text{pre}}_F \leq \mathbf{Adv}^{\text{col}}_F + \varepsilon$, where ε is negligible if F is a variable input length compressing function. In the remainder, we will consider these bounds for granted, and only include security results that improve either of these bounds. A bound is called *tight* if the lower and upper bound are the same up to a constant factor, and *optimal* if the bound is tight with respect to the original lower bound.

2.2 Compression Function Design Strategies

A common way to build compression functions is to base it on a block cipher [25,72,77], or on a (limited number of) permutation(s) [24,75,76]. Preneel et al. [72] analyzed and categorized 64 block cipher based compression functions. Twelve of them were formally proven secure by Black et al. [25]. These results have been recently generalized by Stam [77]. Interestingly, the latter result implies security bounds for some compression functions that do not fit in the PGV-model, like ECHO, Hamsi and SIMD. In the ideal model, everywhere second preimage resistance of the compression function can be proven similar as the preimage resistance, up to a constant (the security analysis differs only in that we give the adversary one query for free). Throughout, by 'PGVx' we denote the x^{th} type compression function of [72]. We note that PGV1, PGV3 and PGV5 are better known as the Matyas-Meyer-Oseas, the Miyaguchi-Preneel and the Davies-Meyer compression functions, respectively.

In the context of permutation based compression functions, Black et al. [24] analyzed $2l$- to l-bit compression functions based on *one* l-bit permutation, and proved them insecure. This result has been generalized by Rogaway and Steinberger [75], Stam [76] and Steinberger [78] to compression functions with arbitrary input and output sizes, and an arbitrary number of underlying permutations. Their bounds indicate the number of queries required to find collisions or preimages for permutation based compression functions.

2.3 Hash Function Design Strategies

In order to allow the hashing of arbitrarily long strings, all SHA-3 candidates employ a specific mode of operation. Central to all designs is the *iterated hash function principle* [62]: on input of an initialization vector IV, the iterated hash

[1] Kelsey and Schneier [56] describe a second preimage attack on the Merkle-Damgård hash function that requires at most approximately 2^{n-L} queries, where the first preimage is of length at most 2^L blocks. This attack does, however, not apply to all SHA-3 candidates. In particular, the wide-pipe SHA-3 candidates remain mostly unaffected due to their increased internal state.

function $\mathcal{H}^f$ based on the compression function f proceeds a padded message $(M_1,\ldots,M_k)$ as follows:

$$\mathcal{H}^f(\mathsf{IV};M_1,\ldots,M_k) = h_k, \text{ where:}$$
$$h_0 = \mathsf{IV},$$
$$h_i = f(h_{i-1},M_i) \text{ for } i = 1,\ldots,k.$$

This principle is also called the plain Merkle-Damgård (MD) design [38,68]. Each of the 14 remaining candidates is based on this design, possibly followed by a final transformation (FT), and/or a chop-function[2].

The padding function $\mathsf{pad} : \mathbb{Z}_2^* \to (\mathbb{Z}_2^m)^*$ is an injective mapping that transforms a message of arbitrary length to a message of length a multiple of m bits (the number of message bits compressed in one compression function iteration). Most of the candidates employ a sufficiently strong padding rule (cf. Fig. 2). Additionally, in some of the designs the message blocks are compressed along with specific counters or tweaks, which may strengthen the padding rule. We distinguish between 'prefix-free' and/or 'suffix-free' padding.

A padding rule is called **suffix-free**, if for any distinct M,M', there exists no bit string X such that $\mathsf{pad}(M') = X\|\mathsf{pad}(M)$. The plain MD design with any suffix-free padding (also called MD-strengthening [62]) preserves collision resistance [38,68]. We generalize this result in Thm. 2 (App. B): informally, this preservation result also holds if the iteration is finalized by a distinct compression function and/or the chop-function. Other security properties, like preimage resistance, are however not preserved in the MD design [4]. It is also proven that the MD design with a suffix-free padding need not necessarily be indifferentiable [32]. However, the MD construction *is* indifferentiable if it ends with a chopping function or a final transformation, both when the underlying compression function is ideal or when the hash function is based on a PGV compression function [32,52,66].

A padding rule is called **prefix-free**, if for any distinct M,M', there exists no bit string X such that $\mathsf{pad}(M') = \mathsf{pad}(M)\|X$. It has been proved that the MD design, based on ideal compression function or ideal PGV construction, with prefix-free padding is indifferentiable from a random oracle [30,32,52,66]. Security notions like collision-resistance, are however not preserved in the MD design with prefix-free only padding.

HAIFA design. A concrete design based on the MD principle is the HAIFA construction [21]. In HAIFA the message is padded in a specific way so as to solve some deficiencies of the original MD construction: in the iteration, each message block is accompanied with a fixed (optional) salt of s bits and a (mandatory) counter C_i of t bits. The counter C_i keeps track of the number of message bits hashed so far, and equals 0 by definition if the i^{th} block does not contain any message bits. Partially due to the properties of this counter, the HAIFA padding rule is suffix- and prefix-free. As a consequence, the construction preserves collision resistance (cf. Thm. 2) and the indifferentiability results of [32] carry over. For the HAIFA design, these indifferentiability results are improved in [19]. Furthermore, the HAIFA construction is proven secure against second preimage attacks if the underlying compression function is assumed to behave like an ideal primitive [27].

Wide-pipe design. In the wide-pipe design [65], the iterated state size is significantly larger than the final hash output: at the end of the iteration, a fraction of the output of a construction is discarded. As proved in [32], the MD construction with a distinct final transformation and/or chopping at the end is indifferentiable from a random oracle.

Sponge functions. We do not explicitly consider sponge functions [16] or their generalization [3] as a specific type of construction: all SHA-3 candidates known to be sponge(-like) functions, CubeHash, Fugue, JH, Keccak and Luffa, can be described in terms of the chop-MD construction (possibly with a final transformation before or instead of the chopping).

2.4 Concrete Security of Hash Functions and Differential Cryptanalysis

All security notions and design approaches mentioned so far substantially assume the underlying primitives of hash functions (such as compression functions, permutations or block ciphers) to behave in an idealized way, where the primitive is randomly drawn from the corresponding class of primitives. However, any practical setting requires the primitives to be efficiently implementable, their representation being compact. As a matter of fact, this does not comply to the random procedure of choice assumed, since it is extremely improbable to select a compactly implementable primitive at random. Thus, once the rule of domain extension has been proven sound assuming the idealness of the underlying primitive, the problem of evaluating the concrete primitive with respect to real-world attacks arises.

In this paper, we choose to employ the toolbox of differential cryptanalysis [23] to address the latter problem, particularly because this analysis approach is also responsible for the attacks on MD5 and SHA-1, that are the main motivation for the SHA-3 competition.

The security of hash functions with respect to such central requirements as (second) preimage and collision resis-

[2] A function g is a final transformation if it differs from f, and is applied to the final state, possibly with the injection of an additional message block. The chop-function is not considered to be (a part of) a final transformation.

tance can be reformulated in terms of the input and output differences of the underlying primitives. A large proportion, though not all, of second-round SHA-3 candidates follow design approaches allowing one to efficiently argue the resistance of their primitives (compression functions, permutations, ciphers) to differential cryptanalysis. These are mainly those candidates relying on the substitution-diffusion networks (often similar to AES [44] or even reusing its components) which make proofs of some relevant properties in the domain of differential cryptanalysis possible [34].

Differentials, DP, EDP. Strictly speaking, for some primitive ϕ mapping to n bits, we do not want the differential probability (DP) of any non-trivial differential (Δ, ∇) over ϕ to significantly deviate from 2^{-n} (see [35] for a comprehensive statistical study of this parameter for idealized permutations and functions). The differential (Δ, ∇) for primitive ϕ consists of input difference Δ and output difference ∇. Once the parameters of ϕ are fixed (keys, salts, initial vectors, etc.), one speaks about the differential probability DP as the probability for (Δ, ∇) to hold averaged over all inputs. The expected DP (EDP) is the DP averaged over all sets of parameters for ϕ.

Differential trails, DTP, EDTP. For most practical constructions, however, it is often impossible to derive any tight (and, thus, informative) upper bounds on DP or even EDP. That is why, to simplify the analysis, one frequently has to revert to differential trails and their probabilities for the evaluation of designs with respect to differential cryptanalysis [23,34]. A differential can be seen as the set of all difference propagation paths from Δ to ∇ through intermediate differences corresponding to the iterations of an iterative construction. Each of these paths is called a differential trail. The probability that a differential trail holds is referred to as differential trail probability (DTP). Similarly to differentials, the expected DTP averaged over all parameters will be denoted as EDTP.

For substitution-diffusion networks, a common belief is that keeping the number of active S-boxes (those involved into the difference propagation) high is correlated with lower values of EDTPs and EDPs which gave rise to the wide trail design strategy [37] and resulted in the successful design of AES [34]. However, formally speaking, the number of active S-boxes and their local DPs do not directly translate to EDP over several rounds of primitives, being only adequate for deriving bounds on single-round EDP. Attempts have been made to compute informative upper bounds on EDP over several rounds of AES: though it has been possible to compute the exact maximum EDP for two rounds of AES [55], the problem persists for 4 rounds [33]. However, one can still prove some upper bounds on EDP over more than two rounds, which has been done e.g. to evaluate the security of SHAvite-3 and ECHO by their respective designers. As applied to SHA-3 candidates, we will talk about EDTP whenever the results deal with more rounds than one could cover by a tight upper bound on EDP.

Depending on a concrete hash function among the second round SHA-3 candidates, the respective designers deal with distinct definitions of a differential trail to derive a bound on EDTP for the building blocks. Such definitions range from the S-box level (e.g. Fugue, Grøstl, and JH) to several AES rounds (e.g. ECHO and SHAvite-3). As a rule, going from the S-box level to the level of several rounds results in bounds on EDTP closer to EDP.

Throughout the paper, we refer to the upper bounds on EDP and EDTP (attained or not attained) as MEDP and MEDTP, respectively.

3 SHA-3 Hash Function Candidates

In this section, we analyze the security of the 14 remaining SHA-3 candidates in more detail, from a provable security perspective as well as with respect to the security against classes of differential attacks. Each paragraph contains an informal discussion for each candidate, its reductionist security results, and its security against differential attacks. The mathematical descriptions of the (abstracted) designs are given in Fig. 1, and the candidates' padding functions are summarized in Fig. 2.

With respect to the reductionist security, for simplicity we only consider the proposals of the SHA-3 candidates that output digests of 256 or 512 bits. Observe that in many candidate SHA-3 hash function families, the algorithms outputting 224 or 384 bits are the same as the 256- or 512-bits algorithms, except for an additional chopping at the end. Particularly, the results of [32] and Thm. 2 carry over in most of the cases. The same remark applies to requirement `(vi)` of NIST.

Requirement `(i)`. All designers claim that their proposal can safely be used in HMAC mode [12] or for randomized hashing [54], and we do not discuss it here;

Requirements `(ii)-(iv)`. Preimage, second preimage and collision resistance of each hash function are discussed in this section. Additionally, we also consider the indifferentiability of the candidates;

Requirement `(v)`. All hash function candidates are secure against the length extension attack, and thus we do not discuss it further.

The reductionist security results for all current candidate hash functions are summarized in Table 1.

With respect to the security against classes of differential attacks, the primitives of all candidate hash functions are analyzed. These security results are summarized in Table 2.

BLAKE:
$(n,l,m,s,t) \in \{(256,256,512,128,64),$
$(512,512,1024,256,128)\}$
$E : \mathbb{Z}_2^{2l} \times \mathbb{Z}_2^m \to \mathbb{Z}_2^{2l}$ a block cipher
$L : \mathbb{Z}_2^{l+s+t} \to \mathbb{Z}_2^{2l}$, $L' : \mathbb{Z}_2^{2l} \to \mathbb{Z}_2^l$ linear functions
$f(h,M,S,C) = L'(E_M(L(h,S,C))) \oplus h \oplus (S\|S)$

BLAKE$(M) = h_k$, where:
$(M_1,\ldots,M_k) \leftarrow \mathsf{pad}_1(M)$; $h_0 \leftarrow \mathsf{IV}$
$S \in \mathbb{Z}_2^s$; $(C_i)_{i=1}^k$ HAIFA-counter
$h_i \leftarrow f(h_{i-1},M_i,S,C_i)$ for $i = 1,\ldots,k$

BMW:
$(n,l,m) \in \{(256,512,512),(512,1024,1024)\}$
$E : \mathbb{Z}_2^m \times \mathbb{Z}_2^l \to \mathbb{Z}_2^m$ a block cipher
$L : \mathbb{Z}_2^{l+m+l} \to \mathbb{Z}_2^l$ a compressing function
$f(h,M) = L(h,M,E_h(M))$
$g(h) = f(\mathsf{IV}',h)$

BMW$(M) = h$, where:
$(M_1,\ldots,M_k) \leftarrow \mathsf{pad}_2(M)$; $h_0 \leftarrow \mathsf{IV}$
$h_i \leftarrow f(h_{i-1},M_i)$ for $i = 1,\ldots,k$
$h \leftarrow \mathsf{chop}_{l-n}[g(h_k)]$

CubeHash:
$(n,l,m) \in \{(256,1024,256),(512,1024,256)\}$
$P : \mathbb{Z}_2^l \to \mathbb{Z}_2^l$ a permutation
$f(h,M) = P(h \oplus (M\|0^{l-m}))$
$g(h) = P^{10}(h \oplus (0^{992}\|1\|0^{31}))$

CubeHash$(M) = h$, where:
$(M_1,\ldots,M_k) \leftarrow \mathsf{pad}_3(M)$; $h_0 \leftarrow \mathsf{IV}$
$h_i \leftarrow f(h_{i-1},M_i)$ for $i = 1,\ldots,k$
$h \leftarrow \mathsf{chop}_{l-n}[g(h_k)]$

ECHO:
$(n,l,m,s,t) \in \{(256,512,1536,128,64/128),$
$(512,1024,1024,256,64/128)\}$
$E : \mathbb{Z}_2^{2048} \times \mathbb{Z}_2^{s+t} \to \mathbb{Z}_2^{2048}$ a block cipher
$L : \mathbb{Z}_2^{2048} \to \mathbb{Z}_2^l$ a linear function
$f(h,M,S,C) = L(E_{S,C}(h\|M) \oplus (h\|M))$

ECHO$(M) = h$, where:
$(M_1,\ldots,M_k) \leftarrow \mathsf{pad}_4(M)$; $h_0 \leftarrow \mathsf{IV}$
$S \in \mathbb{Z}_2^s$; $(C_i)_{i=1}^k$ HAIFA-counter
$h_i \leftarrow f(h_{i-1},M_i,S,C_i)$ for $i = 1,\ldots,k$
$h \leftarrow \mathsf{chop}_{l-n}[h_k]$

Fugue:
$(n,l,m) \in \{(256,960,32),(512,1152,32)\}$
$P,\tilde{P} : \mathbb{Z}_2^l \to \mathbb{Z}_2^l$ permutations
$L : \mathbb{Z}_2^l \times \mathbb{Z}_2^m \to \mathbb{Z}_2^l$ a linear function
$f(h,M) = P(L(h,M))$

Fugue$(M) = h$, where:
$(M_1,\ldots,M_k) \leftarrow \mathsf{pad}_5(M)$; $h_0 \leftarrow \mathsf{IV}$
$h_i \leftarrow f(h_{i-1},M_i)$ for $i = 1,\ldots,k$
$h \leftarrow \mathsf{chop}_{l-n}[\tilde{P}(h_k)]$

Grøstl:
$(n,l,m) \in \{(256,512,512),(512,1024,1024)\}$
$P,Q : \mathbb{Z}_2^l \to \mathbb{Z}_2^l$ permutations
$f(h,M) = P(h \oplus M) \oplus Q(M) \oplus h$
$g(h) = P(h) \oplus h$

Grøstl$(M) = h$, where:
$(M_1,\ldots,M_k) \leftarrow \mathsf{pad}_6(M)$; $h_0 \leftarrow \mathsf{IV}$
$h_i \leftarrow f(h_{i-1},M_i)$ for $i = 1,\ldots,k$
$h \leftarrow \mathsf{chop}_{l-n}[g(h_k)]$

Hamsi:
$(n,l,m) \in \{(256,256,32),(512,512,64)\}$
$P,\tilde{P} : \mathbb{Z}_2^{2n} \to \mathbb{Z}_2^{2n}$ permutations
$\mathsf{Exp} : \mathbb{Z}_2^m \to \mathbb{Z}_2^n$ a linear code
$f(h,M) = h \oplus \mathsf{chop}_n[P(\mathsf{Exp}(M)\|h)]$
$g(h,M) = h \oplus \mathsf{chop}_n[\tilde{P}(\mathsf{Exp}(M)\|h)]$

Hamsi$(M) = h$, where:
$(M_1,\ldots,M_k) \leftarrow \mathsf{pad}_7(M)$; $h_0 \leftarrow \mathsf{IV}$
$h_i \leftarrow f(h_{i-1},M_i)$ for $i = 1,\ldots,k-1$
$h \leftarrow g(h_{k-1},M_k)$

JH:
$(n,l,m) \in \{(256,1024,512),(512,1024,512)\}$
$P : \mathbb{Z}_2^l \to \mathbb{Z}_2^l$ a permutation
$f(h,M) = P(h \oplus (0^{l-m}\|M)) \oplus (M\|0^{l-m})$

JH$(M) = h$, where:
$(M_1,\ldots,M_k) \leftarrow \mathsf{pad}_8(M)$; $h_0 \leftarrow \mathsf{IV}$
$h_i \leftarrow f(h_{i-1},M_i)$ for $i = 1,\ldots,k$
$h \leftarrow \mathsf{chop}_{l-n}[h_k]$

Keccak:
$(n,l,m) \in \{(256,1600,1088),(512,1600,576)\}$
$P : \mathbb{Z}_2^l \to \mathbb{Z}_2^l$ a permutation
$f(h,M) = P(h \oplus (M\|0^{l-m}))$

Keccak$(M) = h$, where:
$(M_1,\ldots,M_k) \leftarrow \mathsf{pad}_9(M)$; $h_0 \leftarrow \mathsf{IV}$
$h_i \leftarrow f(h_{i-1},M_i)$ for $i = 1,\ldots,k$
$h \leftarrow \mathsf{chop}_{l-n}[h_k]$

Luffa:
$(n,l,m,w) \in \{(256,768,256,3),(512,1278,256,5)\}$
$P_i : \mathbb{Z}_2^m \to \mathbb{Z}_2^m$ $(i = 1,\ldots,w)$ permutations
$L : \mathbb{Z}_2^{wm+m} \to \mathbb{Z}_2^{wm}$, $L' : \mathbb{Z}_2^{wm} \to \mathbb{Z}_2^m$ linear functions
$f(h,M) = (P_1(h'_1)\|\cdots\|P_w(h'_w))$
where $(h'_1,\ldots,h'_w) = L(h,M)$
$g(h) = (L'(h)\|L'(f(h,0^m)))$

Luffa$(M) = h$, where:
$(M_1,\ldots,M_k) \leftarrow \mathsf{pad}_{10}(M)$; $h_0 \leftarrow \mathsf{IV}$
$h_i \leftarrow f(h_{i-1},M_i)$ for $i = 1,\ldots,k$
$h \leftarrow \mathsf{chop}_{512-n}[g(h_k)]$

Shabal:
$(n,l,m) \in \{(256,1408,512),(512,1408,512)\}$
$E : \mathbb{Z}_2^{896} \times \mathbb{Z}_2^{1024} \to \mathbb{Z}_2^{896}$ a block cipher
$f(h,C,M) = (y_1,h_3 - M,y_3)$
where $h \in \mathbb{Z}_2^l \to h = (h_1,h_2,h_3) \in \mathbb{Z}_2^{384+m+m}$
and $(y_1,y_3) = E_{M,h_3}(h_1 \oplus (0^{320}\|C),h_2 + M)$

Shabal$(M) = h$, where:
$(M_1,\ldots,M_k) \leftarrow \mathsf{pad}_{11}(M)$; $h_0 \leftarrow \mathsf{IV}$
$h_i \leftarrow f(h_{i-1},\langle i\rangle_{64},M_i)$ for $i = 1,\ldots,k$
$h_{k+i} \leftarrow f(h_{k+i-1},\langle k\rangle_{64},M_k)$ for $i = 1,\ldots,3$
$h \leftarrow \mathsf{chop}_{l-n}[h_{k+3}]$

SHAvite-3:
$(n,l,m,s,t) \in \{(256,256,512,256,64),$
$(512,512,1024,512,128)\}$
$E : \mathbb{Z}_2^l \times \mathbb{Z}_2^{m+s+t} \to \mathbb{Z}_2^l$ a block cipher
$f(h,M,S,C) = E_{M,S,C}(h) \oplus h$

SHAvite-3$(M) = h_k$, where:
$(M_1,\ldots,M_k) \leftarrow \mathsf{pad}_{12}(M)$; $h_0 \leftarrow \mathsf{IV}$
$S \in \mathbb{Z}_2^s$; $(C_i)_{i=1}^k$ HAIFA-counter
$h_i \leftarrow f(h_{i-1},M_i,S,C_i)$ for $i = 1,\ldots,k$

SIMD:
$(n,l,m) \in \{(256,512,512),(512,1024,1024)\}$
$E,\tilde{E} : \mathbb{Z}_2^l \times \mathbb{Z}_2^m \to \mathbb{Z}_2^l$ block ciphers
$f(h,M) = L(h,E_M(h \oplus M))$
$g(h,M) = L(h,\tilde{E}_M(h \oplus M))$

SIMD$(M) = h$, where:
$(M_1,\ldots,M_k) \leftarrow \mathsf{pad}_{13}(M)$; $h_0 \leftarrow \mathsf{IV}$
$h_i \leftarrow f(h_{i-1},M_i)$ for $i = 1,\ldots,k-1$
$h_k \leftarrow g(h_{k-1},M_k)$
$h \leftarrow \mathsf{chop}_{l-n}[h_k]$

Skein:
$(n,l,m) \in \{(256,256,256),(512,512,512)\}$
$E : \mathbb{Z}_2^m \times \mathbb{Z}_2^{128} \times \mathbb{Z}_2^l \to \mathbb{Z}_2^m$ a tweakable block cipher
$f(h,T,M) = E_{h,T}(M) \oplus M$

Skein$(M) = h$, where:
$(M_1,\ldots,M_k) \leftarrow \mathsf{pad}_{14}(M)$; $h_0 \leftarrow \mathsf{IV}$
$(T_i)_{i=1}^k$ round-specific tweaks
$h_i \leftarrow f(h_{i-1},T_i,M_i)$ for $i = 1,\ldots,k$
$h \leftarrow \mathsf{chop}_{l-n}[h_k]$

SHA-2:
$(n,l,m) \in \{(256,256,512),(512,512,1024)\}$
$E : \mathbb{Z}_2^l \times \mathbb{Z}_2^m \to \mathbb{Z}_2^l$ a block cipher
$f(h,M) = E_M(h) + h$

SHA-2$(M) = h_k$, where:
$(M_1,\ldots,M_k) \leftarrow \mathsf{pad}_{15}(M)$; $h_0 \leftarrow \mathsf{IV}$
$h_i \leftarrow f(h_{i-1},M_i)$ for $i = 1,\ldots,k$

Fig. 1 The padding rules employed by the functions are summarized in Fig. 2. In all algorithm descriptions, IV denotes an initialization vector, h denotes state values, M denotes message blocks, S denotes a (fixed) salt, C denotes a counter and T denotes a tweak. The functions L, L', Exp underlying BLAKE, BMW, ECHO, Fugue, Hamsi and Luffa, are explained in the corresponding section.

$$
\begin{aligned}
&\text{BLAKE}: && \mathsf{pad}_1(M) = M\|1\|0^{-|M|-t-2 \bmod m}\|1\|\langle|M|\rangle_t,\\
&\text{BMW}: && \mathsf{pad}_2(M) = M\|1\|0^{-|M|-65 \bmod m}\|\langle|M|\rangle_{64},\\
&\text{CubeHash}: && \mathsf{pad}_3(M) = M\|1\|0^{-|M|-1 \bmod m},\\
&\text{ECHO}: && \mathsf{pad}_4(M) = M\|1\|0^{m-1-(|M|+144 \bmod m)}\|\langle n\rangle_{16}\|\langle|M|\rangle_{128},\\
&\text{Fugue}: && \mathsf{pad}_5(M) = M\|0^{-|M| \bmod m}\|\langle|M|\rangle_{64},\\
&\text{Grøstl}: && \mathsf{pad}_6(M) = M\|1\|0^{-|M|-65 \bmod l}\|\langle\lceil(|M|+65)/l\rceil\rangle_{64},\\
&\text{Hamsi}: && \mathsf{pad}_7(M) = M\|1\|0^{-|M|-1 \bmod m}\|\langle|M|\rangle_{64},\\
&\text{JH}: && \mathsf{pad}_8(M) = M\|1\|0^{383+(-|M| \bmod m)}\|\langle|M|\rangle_{128},\\
&\text{Keccak}: && \mathsf{pad}_9(M) = M\|1\|0^{-|M|-2 \bmod m}1,\\
&\text{Luffa}: && \mathsf{pad}_{10}(M) = M\|1\|0^{(-|M|-1 \bmod m)+256},\\
&\text{Shabal}: && \mathsf{pad}_{11}(M) = M\|1\|0^{-|M|-1 \bmod m},\\
&\text{SHAvite-3}: && \mathsf{pad}_{12}(M) = M\|1\|0^{-|M|-t-17 \bmod m}\|\langle|M|\rangle_t\|\langle n\rangle_{16},\\
&\text{SIMD}: && \mathsf{pad}_{13}(M) = M\|0^{-|M| \bmod m}\|\langle|M|\rangle_m,\\
&\text{Skein}: && \mathsf{pad}_{14}(M) = M'\|0^{(-|M'| \bmod m)+m}, \text{where } M' = \begin{cases} M \text{ if } |M| \equiv 0 \bmod 8,\\ M\|1\|0^{-|M|-1 \bmod 8} \text{ otherwise.}\end{cases}\\
&\text{SHA-2}: && \mathsf{pad}_{15}(M) = \begin{cases} M\|1\|0^{-|M|-65 \bmod m}\|\langle|M|\rangle_{64}, \text{ for } n = 224, 256,\\ M\|1\|0^{-|M|-129 \bmod m}\|\langle|M|\rangle_{128}, \text{ for } n = 384, 512.\end{cases}
\end{aligned}
$$

Fig. 2 The padding rules of all SHA-3 hash function candidates and SHA-2 are summarized. All padding functions output bit strings parsed as sequences of m-bit blocks, where m is the message block length of the corresponding function. For the hash functions BLAKE, ECHO, Shabal, SHAvite-3 and Skein, the complete padding rule of the corresponding hash function is additionally defined by a counter or tweak (as explained in Sect. 3). Particularly, all hash functions employ an injective padding rule.

3.1. The **BLAKE** hash function [8] is a HAIFA construction. The message blocks are accompanied with a HAIFA-counter, and more generally, the function employs a suffix- and prefix-free padding rule. The compression function f is block cipher based[3]. It moreover employs an injective linear function L, and a linear function L' that XORs the first and second halves of the input.

Reductionist security of BLAKE. The compression function of BLAKE is proven optimally collision, second preimage and preimage resistant [5]. As the mode of operation of BLAKE is based on the HAIFA structure, all security properties regarding this type (cf. Sect. 2.3) hold [21], provided the compression function is assumed to be ideal. However, as independently shown by Andreeva et al. [5] and Chang et al. [31], the BLAKE compression function shows non-random behavior: it is differentiable from a random compression function in about $2^{n/4}$ queries, making the above-mentioned security properties invalid. Still, Thm. 2 applies to BLAKE, and as a consequence we obtain $\mathbf{Adv}^{\mathrm{col}}_{\mathcal{H}} = \Theta(q^2/2^n)$. Additionally, a preimage for BLAKE implies a preimage for its final transformation, and we obtain $\mathbf{Adv}^{\mathrm{pre}}_{\mathcal{H}} = \Theta(q/2^n)$. The hash function is moreover proven optimally second preimage resistant in the ideal cipher model by Andreeva et al. [5], which gives $\mathbf{Adv}^{\mathrm{sec}}_{\mathcal{H}} = \Theta(q/2^n)$. Furthermore, the BLAKE hash function is proven indifferentiable from a random oracle if the underlying block cipher is ideal [5, 31].

Bounds on DTP for BLAKE. No informative bounds on DTP are known for BLAKE.

3.2. The **Blue Midnight Wish** (BMW) hash function [51] is a chop-MD construction, with a final transformation before chopping. The hash function employs a suffix-free padding rule. The compression function f is block cipher based[4], and the final transformation g consists of the same compression function with the chaining value processed as a message, and with an initial value as chaining input. The compression function employs a function L which consists of two compression functions with specific properties as specified in [51].

Reductionist security of BMW. The compression function of BMW shows similarities with the PGV3 compression function [25], but no security results are known for this variant. Theorem 2 applies to BMW, where the final transformation has no message block as input, and as a consequence we obtain $\mathbf{Adv}^{\mathrm{col}}_{\mathcal{H}} = \Theta(q^2/2^n)$ (if f is assumed ideal). Additionally, a preimage for BMW implies a preimage for its final transformation, and we obtain $\mathbf{Adv}^{\mathrm{pre}}_{\mathcal{H}} = \Theta(q/2^n)$. Furthermore, albeit no indifferentiability proof for the BMW hash function is known, we note that BMW can be seen as a combination of the HMAC- and the chop-construction, both proven indifferentiable from a random oracle [32]. We remark that a distinguisher for the compression function of BMW is derived in [7].

[3] As observed in [8, Sect. 5], the core part of the compression function can be seen as a permutation keyed by the message, which we view here as a block cipher.

[4] As observed in [51], the compression function can be seen as a generalized PGV3 construction, where the function f_0 of [51] defines the block cipher keyed with the chaining value.

Bounds on DTP for BMW. No informative bounds on DTP are known for BMW.

3.3. The **CubeHash** hash function [15] is a chop-MD construction, with a final transformation before chopping. The compression function f is permutation based, and the final transformation g consists of flipping a certain bit in the state and applying 10 more compression function rounds on zero-messages.

Reductionist security of CubeHash. The compression function of CubeHash is based on one permutation[5], and collisions and preimages for the compression function can be found in one query to the permutation [24]. The CubeHash hash function is as parazoa design proven indifferentiable from a random oracle if the underlying permutation is assumed to be ideal [3]. Using Thm. 1, this indifferentiability bound additionally renders an optimal collision resistance bound for CubeHash, $\mathbf{Adv}^{\mathrm{col}}_{\mathcal{H}} = \Theta(q^2/2^n)$, as well as an improved upper bound $O\left(\frac{q}{2^n} + \frac{q^2}{2^{l-n}}\right)$ on the preimage and second preimage resistance. Note that these bounds are optimal for the $n = 256$ variant.

Bounds on DTP for CubeHash. No informative bounds on DTP are known for CubeHash.

3.4. The **ECHO** hash function [14] is a chop-HAIFA construction. The message blocks are accompanied with a HAIFA-counter, and more generally, the function employs a suffix- and prefix-free padding rule. The compression function f is block cipher based[6]. It moreover employs a linear function L that chops the state in blocks of length l bits, and XORs these.

Reductionist security of ECHO. The compression function of ECHO is a 'chopped single call Type-I' compression function in the categorization of [77]. Therefore, the results of [77, Thm. 15] carry over, yielding optimal security bounds for the compression function. Observe that these results can easily be adjusted to obtain bound $\mathbf{Adv}^{\mathrm{col}}_{\mathsf{chop}\circ f} = \Theta(q^2/2^n)$. ECHO is a combination of HAIFA and chop-MD, but it is unclear whether all HAIFA security properties hold after chopping. Still, Thm. 2 applies to ECHO, and as a consequence we obtain $\mathbf{Adv}^{\mathrm{col}}_{\mathcal{H}} = \Theta(q^2/2^n)$. Additionally, a preimage for ECHO implies a preimage for its last compression function, and we obtain $\mathbf{Adv}^{\mathrm{pre}}_{\mathcal{H}} = \Theta(q/2^n)$. Furthermore, the ECHO hash function would be indifferentiable from a random oracle if the underlying compression function is assumed to be ideal, due to the chopping function at the end [32]. However, the compression function of ECHO is easily differentiable from a random oracle [48], and we cannot directly apply the results of [32].

Bounds on DTP for ECHO. The primitive for all versions of ECHO is ECHO.AES with a 2048-bit block size, having 8 and 10 rounds for hash sizes up to 256 and 512 bit, respectively. One underlying function of ECHO.AES is given by the application of two AES rounds. The exact value of the MEDP for two rounds of AES is about $2^{-28.27}$. One round of ECHO.AES provides one active function. Two rounds provide an upper bound on MEDP of about $2^{-4\cdot 28.27}$, an equivalent of four active functions. Similarly, the bound on the MEDP over 4 rounds yields an equivalent of 16 active functions. MEDP for ECHO is considered averaged over salt and counter. Combining the above results, one obtains upper bounds on the differential trail probability over up to 10 rounds, outlined in Table 2. Note that a differential trail in this case is defined as a concatenation of either 1, 2 or 4 rounds of ECHO.AES. Correspondingly, the MEDTP is computed under the consideration of the above MEDP values for 1, 2 or 4 rounds of ECHO.AES.

3.5. The **Fugue** hash function [53] is a chop-MD construction, with a final transformation before chopping. The hash function employs a suffix-free padding rule. The compression function f is permutation based, and the final transformation consists of a permutation $\tilde{P}$ which differs from P in the parametrization. The compression function employs a linear function L for message injection (TIX of [53]).

Reductionist security of Fugue. The compression function of Fugue is based on one permutation, and collisions and preimages for the compression function can be found in one query to the permutation [24]. As a consequence, the result of Thm. 2 is irrelevant, even though the padding rule of Fugue is suffix-free. The Fugue hash function is as parazoa design proven indifferentiable from a random oracle if the underlying permutations P and $\tilde{P}$ are assumed to be ideal [3]. Using Thm. 1, this indifferentiability bound additionally renders an optimal collision resistance bound for Fugue, $\mathbf{Adv}^{\mathrm{col}}_{\mathcal{H}} = \Theta(q^2/2^n)$, as well as an improved upper bound $O\left(\frac{q}{2^n} + \frac{q^2}{2^{l-m-n}}\right)$ on the preimage and second preimage resistance. Note that these bounds are optimal for the $n = 256$ variant. We remark that a distinguisher for the final round of Fugue is derived in [9,49].

Bounds on DTP for Fugue. For Fugue a number of arguments against differential attacks exist that are, compared to all other bounds for candidates discussed in this paper, rather different in nature. The main statement for Fugue is a theorem, that assumes that the adversary is given a random pair of states satisfying any difference of his choice. It then investigates whether the adversary would be able to find a pair of message inputs that converts these states into a full internal collision after *exactly* four round transformations. The theorem proves, under some independence assumption, that

[5] Effectively, this permutation consists of one simpler permutation executed 16 times iteratively.

[6] As observed in [14], the core part of the compression function can be seen as a permutation keyed by the salt and counter, which we view here as a block cipher. This cipher is AES-based.

the probability that such message inputs exist is bounded by 2^{-168}. For a lower number of rounds, the probability will be higher though, while it is not clear how the probability of a higher number of rounds will be.

3.6. The **Grøstl** hash function [50] is a chop-MD construction, with a final transformation before chopping. The hash function employs a suffix-free padding rule. The compression function f is permutation based, and the final transformation g is defined as $g(h) = P(h) \oplus h$.

Reductionist security of Grøstl. The compression function of Grøstl is permutation based, and the results of [75, 76] apply. Furthermore, the preimage resistance of the compression function is analyzed in [47], and an upper bound for collision resistance can be obtained easily. As a consequence, we obtain tight security bounds on the compression function, $\mathbf{Adv}^{\mathrm{pre}}_{f} = \Theta(q^2/2^l)$ and $\mathbf{Adv}^{\mathrm{col}}_{f} = \Theta(q^4/2^l)$. In the ideal model, everywhere second preimage resistance of the compression function can be proven similar as the preimage resistance, up to a constant (the security analysis differs only in that we give the adversary one query for free). Theorem 2 applies to Grøstl, where the final transformation has no message block as input. Observe that we also have $\mathbf{Adv}^{\mathrm{col}}_{\mathsf{chop}\circ g} = \Theta(q^2/2^n)$, and as a consequence we obtain $\mathbf{Adv}^{\mathrm{col}}_{\mathcal{H}} = \Theta(q^2/2^n)$. Additionally, a preimage for Grøstl implies a preimage for its final transformation, and we obtain $\mathbf{Adv}^{\mathrm{pre}}_{\mathcal{H}} = \Theta(q/2^n)$. The hash function is moreover proven optimally second preimage resistant in the ideal permutations model [6], which gives $\mathbf{Adv}^{\mathrm{sec}}_{\mathcal{H}} = \Theta(q/2^{n-L})$, where the first preimage is of length at most 2^L blocks. Furthermore, the Grøstl hash function is proven indifferentiable from a random oracle if the underlying permutations are ideal [1].

Bounds on DTP for Grøstl. The compression function of Grøstl is based on two permutations of size 512 for Grøstl-224 and Grøstl-256, and is based on two permutations of size 1024 for Grøstl-384 and Grøstl-512. Those permutations are designed according to the wide trail design strategy [37], hence known bounds on the MEDTP carry over as follows. For 2 rounds, at least 9 S-boxes, for 4 rounds, at least 81 S-boxes are active. A lower bound on the DP of the employed AES S-box is 2^{-6}. From this, the 4-round MEDTP is 2^{-486}. For the 10-round 512-bit permutation the MEDTP is hence 2^{-1026}, and for the 14-round 1024-bit permutation the MEDTP is hence 2^{-1512}.

3.7. The **Hamsi** hash function [60] is a MD construction, with a final transformation before chopping. The hash function employs a suffix-free padding rule. The compression function f is permutation based, but the last round is executed with a compression function g based on a permutation $\tilde{P}$ which differs from P in the parametrization. The compression functions employ a linear code Exp for message injection [60].

Reductionist security of Hamsi. The compression function of Hamsi is a 'chopped single call Type-I' compression function in the categorization of [77]. Therefore, the results of [77, Thm. 15] carry over, yielding optimal security bounds for the compression function. Observe that these bounds also apply to the function g. Theorem 2 applies to Hamsi, and as a consequence we obtain $\mathbf{Adv}^{\mathrm{col}}_{\mathcal{H}} = \Theta(q^2/2^n)$. Additionally, a preimage for Hamsi implies a preimage for its last compression function, and we obtain $\mathbf{Adv}^{\mathrm{pre}}_{\mathcal{H}} = \Theta(q/2^n)$. Furthermore, the Hamsi hash function is proven indifferentiable from a random oracle if the underlying permutations are ideal [61].

Bounds on DTP for Hamsi. No informative bounds on DTP are known for Hamsi.

3.8. The **JH** hash function [83] is a chop-MD construction. The hash function employs a suffix-free padding rule. The compression function f is permutation based.

Reductionist security of JH. The compression function of JH is based on one permutation, and collisions and preimages for the compression function can be found in one query to the permutation [24]. As a consequence, the result of Thm. 2 is irrelevant, even though the padding rule of JH is suffix-free. The JH hash function is proven optimally collision resistant [63], and proven preimage and second preimage resistant up to bound $O\left(\frac{q}{2^n} + \frac{q^2}{2^{l-m}}\right)$ [6]. Furthermore, the JH hash function is proven indifferentiable from a random oracle if the underlying permutation is ideal [20].

Bounds on DTP for JH. The JH permutation consists of 42 rounds of a generalized AES-like structure. Note the number of rounds has been increased in the 3rd round submission documents for JH. The maximum DP of the 4-bit S-box used is 2^{-2}. The specification document suggests that 17 rounds of the underlying permutation yield at least 296 active S-boxes. Since no formal treatment of the differential effect has been conducted, we consider the differential trails on the S-box level in this work. So the MEDTP for the $2 \cdot 17 = 34$ rounds is at most 2^{-1184}, which is also the MEDTP value we take for all 42 rounds, as no lower bounds are provided on the number of active S-boxes for less than 17 rounds.

3.9. The **Keccak** hash function [18] is a chop-MD construction. The compression function f is permutation based. The hash function output is obtained by chopping off $l - n$ bits of the state. Notice that the parameters of Keccak satisfy $l = 2n + m$.

Reductionist security of Keccak. The compression function of Keccak is based on one permutation, and collisions and preimages for the compression function can be found in one query to the permutation [24]. The Keccak hash function is proven indifferentiable from a random oracle if the underlying permutation is assumed to be ideal [17]. Using Thm. 1, this indifferentiability bound additionally renders an optimal

collision resistance bound for Keccak, $\mathbf{Adv}^{\mathrm{col}}_{\mathcal{H}} = \Theta(q^2/2^n)$, as well as an optimal preimage second preimage resistance bound $\Theta(q/2^n)$.

Bounds on DTP for Keccak. The Keccak permutation consists of 24 rounds. In [36] for the MEDTP a value of 2^{-32} is proven for 3 rounds, 2^{-74} for 6 rounds, and 2^{-296} for the full 24 rounds.

3.10. The **Luffa** hash function [39] is a chop-MD construction, with a final transformation before chopping. The compression function f is permutation based, and the final transformation g is built on this compression function and a linear function L' that chops the state in blocks of length m bits, and XORs these. The compression function employs a linear function L for message injection (MI of [39])[7]. Notice that the state size of Luffa satisfies $l = w \cdot m$.

Reductionist security of Luffa. The compression function of Luffa is based on w permutations executed independently. As a consequence, collisions and preimages for the compression function can be found in at most 5 queries to the permutations [24]. The Luffa hash function borrows characteristics from the sponge design and is similar to the parazoa design, if the permutation P consisting of the w permutations P_i is considered ideal, and ideas from the indifferentiability proofs of [3,17] may carry over. However, for the case of w different permutations P_i this is not immediately clear. We remark that a distinguisher for the permutation of Luffa is derived in [57].

Bounds on DTP for Luffa. The Luffa permutations consist of 8 rounds. Using an exhaustive search, designers report a minimal number of active S-boxes for 4 rounds of 31. A lower bound on the DP of the employed AES S-box is 2^{-2}, hence the MEDTP for the full 8-round permutation is 2^{-124}.

3.11. The **Shabal** hash function [28] is a chop-MD construction. The message blocks are accompanied with a counter, and the last block is iterated three times. In particular, the function employs a suffix- and prefix-free padding rule. The compression function f is block cipher based. Notice that the parameters of Shabal satisfy $l = 384 + 2m$.

Reductionist security of Shabal. A bound on the collision resistance of the compression function of Shabal is derived in [28]. Concretely, it is proven that the Shabal compression function is collision resistant up to $q = 2^{(l-m)/2}$ queries. Theorem 2 applies to Shabal. Collision and preimage resistance of Shabal are studied in [28], yielding optimal bounds $\mathbf{Adv}^{\mathrm{pre}}_{\mathcal{H}} = \Theta(q/2^n)$ and $\mathbf{Adv}^{\mathrm{col}}_{\mathcal{H}} = \Theta(q^2/2^n)$. Furthermore, the same authors prove the Shabal hash function to be indifferentiable from a random oracle if the underlying block cipher is assumed to be ideal [28]. Using Thm. 1, this indifferentiability bound additionally renders an improved upper bound $O\left(\frac{q}{2^n} + \frac{q^2}{2^{l-m}}\right)$ on the second preimage resistance. Note that this bound is optimal for the $n = 256$ variant. We remark that a distinguisher for the block cipher of Shabal is derived in [10,11,58,71,80].

Bounds on DTP for Shabal. No informative bounds on DTP are known for Shabal.

3.12. The **SHAvite-3** hash function [22] is a HAIFA construction. The message blocks are accompanied with a HAIFA-counter, and more generally, the function employs a suffix- and prefix-free padding rule. The compression function f is block cipher based.

Reductionist security of SHAvite-3. The compression function of SHAvite-3 is the PGV5 compression function, and the security results of [25] carry over. As a consequence, we obtain optimal security bounds on the compression function. The mode of operation of SHAvite-3 is based on the HAIFA structure, and as a consequence all security properties regarding this type hold [21]. In particular, the design preserves collision resistance, and as a consequence we obtain $\mathbf{Adv}^{\mathrm{col}}_{\mathcal{H}} = \Theta(q^2/2^n)$. Also, the design is secure against second preimage attacks. Additionally, a preimage for SHAvite-3 implies a preimage for its last compression function, and we obtain $\mathbf{Adv}^{\mathrm{pre}}_{\mathcal{H}} = \Theta(q/2^n)$. Finally, the SHAvite-3 hash function is indifferentiable from a random oracle if the underlying block cipher is assumed to be ideal, due to the prefix-free padding [32]. This result has been improved under the assumption that the underlying compression function is ideal [19]. However, the compression function of SHAvite-3 is easily differentiable from a random oracle due to the presence of fixed-points.

Bounds on DTP for SHAvite-3. The underlying cipher for SHAvite-3_{224} and SHAvite-3_{256} is E^{256} which operates on two block halves of 128 bit each using the balanced Feistel construction. Every two or three rounds of balanced Feistel add at least one or two active functions, respectively. The differential probability of one function (3 AES rounds) in E^{256} is shown to be upper bounded by 2^{-49} [22]. This leads to at least 8 functions active within 12 rounds and a differential trail probability of at most 2^{-392} after 12 rounds of E^{256}. While the lower bounds of [22] on the number of differentially active functions appear to be tight for E^{256}, this is not the case for E^{512}, the block cipher underlying SHAvite-3_{384} and SHAvite-3_{512} which is based on a 4-line type-II generalized Feistel network structure. One can prove that 2, 3, 4, 5 and 6 rounds of E^{512} yield at least 1, 2, 3, 4 and 6 differentially active functions, respectively. This result combined with the proven upper bound of 2^{-113} on the differential probability of one function (4 AES rounds) [22] gives the upper bound of 2^{-1356} on the differential trail probability

[7] We defined the output transformation in a slightly more complicated but unified way. Essentially, Luffa$_{256}$ simply outputs $L'(h)$. Observe that we implicitly captured the extra compression function call in the adjusted padding.

over 14 rounds of E^{512}. Note that we consider differential trails on the round level of SHAvite-3 here.

3.13. The **SIMD** hash function [64] is a chop-MD construction, with a final transformation before chopping. The hash function employs a suffix-free padding rule. The compression function f is block cipher based, but the last round is executed with a compression function g based on a block cipher $\tilde{E}$ which differs from E in the parametrization. These function employ a quasi-group operation[8] L [64].

Reductionist security of SIMD. The compression function of SIMD is a 'rate-1 Type-I' compression function in the categorization of [77]. Therefore, the results of [77, Thm. 6] carry over, yielding optimal security bounds for the compression function. Observe that these bounds also apply to the function g. Observe moreover that these results can easily be adjusted to obtain bound $\mathbf{Adv}^{\mathrm{col}}_{\mathsf{chop}\circ g} = \Theta(q^2/2^n)$. Theorem 2 applies to SIMD, and as a consequence we obtain $\mathbf{Adv}^{\mathrm{col}}_{\mathcal{H}} = \Theta(q^2/2^n)$. Additionally, a preimage for SIMD implies a preimage for its last compression function, and we obtain $\mathbf{Adv}^{\mathrm{pre}}_{\mathcal{H}} = \Theta(q/2^n)$. Furthermore, the SIMD hash function would be indifferentiable from a random oracle if the underlying compression functions are assumed to be ideal, due to the chopping function at the end [32]. However, the compression functions of SIMD are easily differentiable from a random oracle [26], and we cannot directly apply the results of [32].

Bounds on DTP for SIMD. A number of arguments against differential attacks are made [26] that do not directly fit into the framework considered in this paper as they are based on some (mild) conditions rather than applying unconditionally. One assumption is that the attacker has to use differences on the message input, which rules out certain attacks on building blocks like the block cipher or compression function. Using a minimum distance of 520 for SIMD-256 and 1032 for SIMD-512, and a number of assumptions some of which are in favor of an attacker and some of the design, authors give a lower bound of at least 132 conditions for any differential trail in SIMD-256, and a lower bound of at least 253 conditions for any differential trail in SIMD-512.

3.14. The **Skein** hash function [42] is a chop-MD construction. The message blocks are accompanied with a round-specific tweak[9], and more generally, the function employs a suffix- and prefix-free padding rule. The compression function f is based on a tweakable block cipher.

Reductionist security of Skein. The compression function of Skein is the PGV1 compression function, with a difference that a tweak is involved. As claimed in [13], the results of [25] carry over, which in turn results in optimal security bounds on the compression function. Theorem 2 applies to Skein, and as a consequence we obtain $\mathbf{Adv}^{\mathrm{col}}_{\mathcal{H}} = \Theta(q^2/2^n)$. Additionally, a preimage for Skein implies a preimage for its last compression function, and we obtain $\mathbf{Adv}^{\mathrm{pre}}_{\mathcal{H}} = \Theta(q/2^n)$. The hash function is moreover proven optimally second preimage resistant in the ideal cipher model [6], which gives $\mathbf{Adv}^{\mathrm{sec}}_{\mathcal{H}} = \Theta(q/2^n)$. Furthermore, the Skein hash function is proven indifferentiable from a random oracle if the underlying tweakable block cipher is assumed to be ideal [13]. This proof is based on the preimage-awareness approach [41].

Bounds on DTP for Skein. No informative bounds on DTP are known for Skein.

3.15. The **SHA-2** hash function [45] is an MD construction. The hash function employs a suffix-free padding. The compression function f is block cipher based.

Reductionist security of SHA-2. The compression function of SHA-2 is the PGV5 compression function (with minor modification), and the security results of [25] carry over. As a consequence, we obtain optimal security bounds on the compression function. Theorem 2 applies to SHA-2, and as a consequence we obtain $\mathbf{Adv}^{\mathrm{col}}_{\mathcal{H}} = \Theta(q^2/2^n)$. Additionally, a preimage for SHA-2 implies a preimage for its last compression function, and we obtain $\mathbf{Adv}^{\mathrm{pre}}_{\mathcal{H}} = \Theta(q/2^n)$. The hash function is moreover proven optimally second preimage resistant in the ideal function model [27], which gives $\mathbf{Adv}^{\mathrm{sec}}_{\mathcal{H}} = \Theta(q/2^{n-L})$, where the first preimage is of length at most 2^L blocks. Furthermore, the SHA-2 hash function is differentiable from a random oracle due to an attack described in [32].

Bounds on DTP for SHA-2. No informative bounds on DTP are known for SHA-2.

4 Summary and Conclusions

In this survey, we compared the security achieved by the round 2 SHA-3 hash function candidates. The main contribution of this paper are the summary of the security reductions for the hash function candidates in Table 1, and a comparison of the resistance of the designs against classes of differential attacks in Table 2. Before giving an interpretation of these results, we first make some remarks on the provided classification.

– Assuming ideality of the underlying primitives (permutations or block ciphers) is not realistic. In particular, none of the candidates' primitives is ideal, and some

[8] For any of the variables fixed, the function L is a permutation.

[9] More formally, the design is based on the UBI (unique block identifier) chaining mode which queries its underlying tweakable block cipher on additional tweaks, that differ in each iteration. The general description of Skein involves a specific final transformation. In the primary proposal of the hash function, however, this final transformation consists of another execution of the compression function, with an output-specific tweak and with message 0^m. As we included this final message block in the padding, the given description of Skein suffices.

Table 1 A schematic summary of reductionist results on second round SHA-3 candidates and SHA-2. The *first* column describes the hash function construction, and the *second* and *third* column show which hash functions have a suffix-free (sf) or prefix-free (pf) padding. The *fourth* column summarizes the main parameters n, l, m, which denote the hash function output size, the chaining value size and the message input size, respectively. In the remaining columns, the security bounds are summarized together with the underlying assumptions. All other notions and notations and further explained in Sect. 2.

	type	sf	pf	(n,l,m)	$\mathbf{Adv}^{\mathrm{pre}}_{f}$	$\mathbf{Adv}^{\mathrm{sec}}_{f}$	$\mathbf{Adv}^{\mathrm{col}}_{f}$	$\mathbf{Adv}^{\mathrm{pre}}_{\mathcal{H}}$	$\mathbf{Adv}^{\mathrm{sec}}_{\mathcal{H}}$	$\mathbf{Adv}^{\mathrm{gcol}}_{\mathcal{H}}$	$\mathbf{Adv}^{\mathrm{col}}_{\mathcal{H}}$	$\mathbf{Adv}^{\mathrm{pro}}_{\mathcal{H}}$
BLAKE	HAIFA	✓	✓	$(256,256,512)$ or $(512,512,1024)$	$\Theta(q/2^n)$ E ideal	$\Theta(q/2^n)$ E ideal	$\Theta(q^2/2^n)$ E ideal	$\Theta(q/2^n)$ E ideal	$\Theta(q/2^n)$ E ideal	$\leq \mathbf{Adv}^{\mathrm{gcol}}_{f}$	$\Theta(q^2/2^n)$ E ideal (proof by preservation)	$\Theta((Kq)^2/2^n)$ E ideal
BMW	chop-(MD+FT)	✓	✗	$(256,512,512)$ or $(512,1024,1024)$	PGV3-like E ideal	PGV3-like E ideal	PGV3-like E ideal	$\Theta(q/2^n)$ f ideal		$\leq \mathbf{Adv}^{\mathrm{gcol}}_{f}$ $+\mathbf{Adv}^{\mathrm{gcol}}_{\mathsf{chop}\circ g}$	$\Theta(q^2/2^n)$ f ideal (proof by preservation)	chopHMAC-like f ideal
CubeHash	chop-(MD+FT)	✗	✗	$(256,1024,256)$ or $(512,1024,256)$	$\Theta(1)$ P ideal	$\Theta(1)$ P ideal	$\Theta(1)$ P ideal	$O\left(\frac{q}{2^n}+\frac{q^2}{2^{l-n}}\right)$ P ideal	$O\left(\frac{q}{2^n}+\frac{q^2}{2^{l-n}}\right)$ P ideal	(no preservation)	$\Theta(q^2/2^n)$ P ideal	$O((Kq)^2/2^{l-n})$ P ideal
ECHO	chop-HAIFA	✓	✓	$(256,512,1536)$ or $(512,1024,1024)$	$\Theta(q/2^l)$ E ideal	$\Theta(q/2^l)$ E ideal	$\Theta(q^2/2^l)$ E ideal	$\Theta(q/2^n)$ E ideal	chop-HAIFA f ideal	$\leq \mathbf{Adv}^{\mathrm{gcol}}_{\mathsf{chop}\circ f}$	$\Theta(q^2/2^n)$ E ideal (proof by preservation)	chopMD construction
Fugue	chop-(MD+FT)	✓	✗	$(256,960,32)$ or $(512,1152,32)$	$\Theta(1)$ P ideal	$\Theta(1)$ P ideal	$\Theta(1)$ P ideal	$O\left(\frac{q}{2^n}+\frac{q^2}{2^{l-m-n}}\right)$ $P,\bar{P}$ ideal	$O\left(\frac{q}{2^n}+\frac{q^2}{2^{l-m-n}}\right)$ $P,\bar{P}$ ideal	(no preservation)	$\Theta(q^2/2^n)$ $P,\bar{P}$ ideal	$O((Kq)^2/2^{l-m-n})$ $P,\bar{P}$ ideal
Grøstl	chop-(MD+FT)	✓	✗	$(256,512,512)$ or $(512,1024,1024)$	$\Theta(q^2/2^l)$ P,Q ideal	$\Theta(q^2/2^l)$ P,Q ideal	$\Theta(q^4/2^l)$ P,Q ideal	$\Theta(q/2^n)$ P ideal	$\Theta(q/2^{n-L})$ P,Q ideal	$\leq \mathbf{Adv}^{\mathrm{gcol}}_{f}$ $+\mathbf{Adv}^{\mathrm{gcol}}_{\mathsf{chop}\circ g}$	$\Theta(q^2/2^n)$ P,Q ideal (proof by preservation)	$O((Kq)^4/2^l)$ P,Q ideal
Hamsi	MD+FT	✓	✗	$(256,256,32)$ or $(512,512,64)$	$\Theta(q/2^n)$ P ideal	$\Theta(q/2^n)$ P ideal	$\Theta(q^2/2^n)$ P ideal	$\Theta(q/2^n)$ $\tilde{P}$ ideal		$\leq \mathbf{Adv}^{\mathrm{gcol}}_{f}$ $+\mathbf{Adv}^{\mathrm{gcol}}_{g}$	$\Theta(q^2/2^n)$ $P,\tilde{P}$ ideal (proof by preservation)	$\Theta((Kq)^2/2^n)$ $P,\tilde{P}$ ideal
JH	chop-MD	✓	✗	$(256,1024,512)$ or $(512,1024,512)$	$\Theta(1)$ P ideal	$\Theta(1)$ P ideal	$\Theta(1)$ P ideal	$O\left(\frac{q}{2^n}+\frac{q^2}{2^{l-m}}\right)$ P ideal	$O\left(\frac{q}{2^n}+\frac{q^2}{2^{l-m}}\right)$ P ideal	(no preservation)	$\Theta(q^2/2^n)$ P ideal	$O\left(\frac{q^3}{2^{l-m}}+\frac{Kq^3}{2^{l-n}}\right)$ P ideal
Keccak	chop-MD	✗	✗	$(256,1600,1088)$ or $(512,1600,576)$	$\Theta(1)$ P ideal	$\Theta(1)$ P ideal	$\Theta(1)$ P ideal	$\Theta(q/2^n)$ P ideal	$\Theta(q/2^n)$ P ideal	(no preservation)	$\Theta(q^2/2^n)$ P ideal	$\Theta((Kq)^2/2^{l-m})$ P ideal
Luffa	chop-(MD+FT)	✗	✗	$(256,768,256)$ or $(512,1278,256)$	$\Theta(1)$ P_i ideal	$\Theta(1)$ P_i ideal	$\Theta(1)$ P_i ideal			(no preservation)		sponge-like P_i ideal
Shabal	chop-MD	✓	✓	$(256,1408,512)$ or $(512,1408,512)$			$O(q^2/2^{l-m})$ E ideal	$\Theta(q/2^n)$ E ideal	$O\left(\frac{q}{2^n}+\frac{q^2}{2^{l-m}}\right)$ E ideal	$\leq \mathbf{Adv}^{\mathrm{gcol}}_{\mathsf{chop}\circ f}$	$\Theta(q^2/2^n)$ E ideal	$O((Kq)^2/2^{l-m})$ E ideal
SHAvite-3	HAIFA	✓	✓	$(256,256,512)$ or $(512,512,1024)$	$\Theta(q/2^n)$ E ideal	$\Theta(q/2^n)$ E ideal	$\Theta(q^2/2^n)$ E ideal	$\Theta(q/2^n)$ E ideal	$\Theta(q/2^n)$ f ideal	$\leq \mathbf{Adv}^{\mathrm{gcol}}_{f}$	$\Theta(q^2/2^n)$ E ideal (proof by preservation)	$O((Kq)^2/2^n)$ E ideal
SIMD	chop-(MD+FT)	✓	✗	$(256,512,512)$ or $(512,1024,1024)$	$\Theta(q/2^l)$ E ideal	$\Theta(q/2^l)$ E ideal	$\Theta(q^2/2^l)$ E ideal	$\Theta(q/2^n)$ $\tilde{E}$ ideal		$\leq \mathbf{Adv}^{\mathrm{gcol}}_{f}$ $+\mathbf{Adv}^{\mathrm{gcol}}_{\mathsf{chop}\circ g}$	$\Theta(q^2/2^n)$ $E,\tilde{E}$ ideal (proof by preservation)	chopMD construction
Skein	chop-MD	✓	✓	$(256,512,512)$ or $(512,512,512)$	$\Theta(q/2^l)$ E ideal	$\Theta(q/2^l)$ E ideal	$\Theta(q^2/2^l)$ E ideal	$\Theta(q/2^n)$ E ideal	$\Theta(q/2^n)$ E ideal	$\leq \mathbf{Adv}^{\mathrm{gcol}}_{f}$	$\Theta(q^2/2^n)$ E ideal (proof by preservation)	$O((Kq)^2/2^l)$ E ideal
SHA-2	MD	✓	✗	$(256,256,512)$ or $(512,512,1024)$	$\Theta(q/2^n)$ E ideal	$\Theta(q/2^n)$ E ideal	$\Theta(q^2/2^n)$ E ideal	$\Theta(q/2^n)$ E ideal	$\Theta(q/2^{n-L})$ f ideal	$\leq \mathbf{Adv}^{\mathrm{gcol}}_{f}$	$\Theta(q^2/2^n)$ E ideal (proof by preservation)	$\Theta(1)$ f ideal

Table 2 Maximum EDTP (MEDTP) for the 14 second round SHA-3 candidates. Note that the presented bounds do not allow for direct comparisons, as proofs of bounds on EDTP for different hash functions work with distinct definitions of a differential trail. See the respective subsections of Sect. 3 for the differential trail definitions for each specific hash function under consideration. Note that the absence of MEDTP-related information (marked with -) does not mean that the respective design is insecure against differential cryptanalysis. Neither does it mean that there are no other attacks on the design.

	hash size n	rounds R	$\log_2$MEDTP for R	r for MEDTP $\approx 2^{-n}$	r/R for r with $\log_2$MEDTP $\approx -n$	$\log_2$MEDTP for $R/4$	$\log_2$ MEDTP for $R/3$	$\log_2$MEDTP for $R/2$
BLAKE	all	-	-	-	-	-	-	-
BMW	all	-	-	-	-	-	-	-
CubeHash	all	-	-	-	-	-	-	-
ECHO	224	8	-904.64	4	0.500	-113.08	-141.35	-452.32
	256	8	-904.64	4	0.500	-113.08	-141.35	-452.32
	384	10	-1017.72	4	0.400	-141.35	-452.32	-480.32
	512	10	-1017.72	6	0.600	-141.35	-452.32	-480.32
Fugue	all		Comes with bounds regarding differential attacks that do not fit into the framework we use here. See Sect. 3.5.					
Grøstl	224	10	-972	4	0.400	-54	-102	-534
	256	10	-972	4	0.400	-54	-102	-534
	384	14	-1458	4	0.286	-102	-486	-588
	512	14	-1458	5	0.357	-102	-486	-588
Hamsi	all	-	-	-	-	-	-	-
JH	224	42	< -1184	< 17	< 0.405	-	-	< -592
	256	42	< -1184	< 17	< 0.405	-	-	< -592
	384	42	< -1184	< 17	< 0.405	-	-	< -592
	512	42	< -1184	≈ 17	≈ 0.405	-	-	< -592
Keccak	224	24	-296	24	1	-74	-74	-148
	256	24	-296	24	1	-74	-74	-148
	384	24	-296	> 24	> 1	-74	-74	-148
	512	24	-296	> 24	> 1	-74	-74	-148
Luffa	224	8	-124	> 8	> 1	-	-	-
	256	8	-124	> 8	> 1	-	-	-
	384	8	-124	> 8	> 1	-	-	-
	512	8	-124	> 8	> 1	-	-	-
Shabal	all	-	-	-	-	-	-	-
SHAvite-3	224	12	-392	8	0.667	-98	-98	-196
	256	12	-392	9	0.750	-98	-98	-196
	384	14	-1469	5	0.357	-339	-452	-678
	512	14	-1469	6	0.423	-339	-452	-678
SIMD	all		Comes with bounds regarding differential attacks that do not fit into the framework we use here. See Sect. 3.13.					
Skein	all	-	-	-	-	-	-	-
SHA-2	all	-	-	-	-	-	-	-

even have identified weaknesses. However, assuming ideality of these primitives gives significantly more confidence in the security of the higher level structure and is the only way to get useful (and comparable) security bounds on the candidate hash functions;

– The fact that different hash functions have different bounds, does not directly imply that one of the functions offers a higher level of security: albeit the underlying structure of the basic primitives is abstracted away (see the previous item), still many differences among the schemes remain (chaining size, message input size, etc.). Moreover, not all bounds are tight, and different definitions of differential trails are employed.

In general, stronger assumptions as a rule imply stronger claims, however, results making less assumptions often offer higher correlation with the real-world security.

4.1 Reductionist Security

Security of the compression function. For the sponge(-like) hash functions, CubeHash, Fugue, JH, Keccak and Luffa, collisions and preimages for the compression function can be found in a constant number of queries. This does not have direct implications for the security of the hash function. In fact, the only consequence is that it becomes unreasonable to assume ideality of the compression function in order to prove security at a higher level. Most of the remaining nine candidates are provided with a tight bound for collision and/or preimage resistance of the compression function, merely due to the results of [25,77,5]. Single exception is BMW, for which the results of [77] are not directly applicable. With respect to second preimage resistance of the compression functions, we note that in the ideal model (everywhere) second preimage resistance of the compression function can be proven similar as the preimage resistance, up to a constant (the security analysis differs only in that we give the adversary one query for free);

Indifferentiability of the hash function. Ten of the candidates are proven indifferentiable from a random oracle, and four of the candidates have a similar constructions to ones proven indifferentiable. We note that there exist some differences among the bounds. For instance, for the hash function variant outputting $n = 512$ bits, the indifferentiability bounds vary between $O(Kq^3/2^{512})$ and $O((Kq)^2/2^{1024})$. These differences are mainly caused by the fact that the bounds are parameterized by the internal chaining value size l, rather than the output size n (as is the case for bounds on the collision resistance). As a consequence, a higher state size often results in a better indifferentiability bound. The indifferentiability results are powerful, as they render bounds on the (second) preimage and collision resistance of the design (Thm. 1);

(Second) preimage resistance of the hash function. Most of the hash functions are not provided with an optimal security bound on the second preimage resistance. Main cause for this is that the MD design does not preserve second preimage resistance [4]. Additionally, the second preimage bound that can be derived via the indifferentiability (Thm. 1) is not always sufficiently tight. Proving security against these attacks could be attempted either by making a different (possibly weaker) assumption on the compression function or by basing it directly on the ideality of the underlying block cipher or permutation(s). A fruitful direction might be the graph based approach followed in [5,6,27,28];

Collision resistance of the hash function. Except for the sponge(-like) functions, the collision resistance preservation result of Thm. 2 (App. B) applies to all candidates. This theorem results in a bound on the generic collision resistance of the hash function, which, intuitively, means that 'finding collisions for the hash function is at least as hard as finding collisions for (one of) the underlying function(s)'. Together with the collision resistance bounds on the compression functions in the ideal model, the preservation result allows for obtaining a collision resistance bound on the entire hash function. This leads to optimal bounds on the collision resistance for ECHO, Grøstl, Hamsi, SHAvite-3, SIMD and Skein. For BLAKE, CubeHash, Fugue, JH, Keccak and Shabal, the same optimal bound is obtained differently (e.g. based on the indifferentiability of the hash function). Again, the graph based approach may be suitable to prove collision resistance of the candidates for which no collision resistance bound is yet obtained.

4.2 Bounds on the Differential Properties

We presented a first approach to the problem of comparing the differential properties of diverse contemporary hash function designs at the example of the second round SHA-3 candidates. This research problem is not easy, nor is it close to being resolved, which is mainly for the following reasons:

- Many hash function primitives have been designed using non-SPN approaches, mostly relying on ARX operations, for which it does not appear trivial to compute an informative upper bound on any differential properties such as differential trail probability. Therefore, for approximately half of the SHA-3 candidates we do not know where the bounds are and we have to perform our comparative analysis among the remaining designs only;
- The bounds on differential properties are derived for single underlying primitives, while a hash function can invoke multiple primitives in each compression step: it is one invocation for ECHO, Fugue, JH, and SHAvite-3, two invocations for Grøstl, as well as 3 to 5 invocations for Luffa. This fact has to be taken into account before drawing conclusions about the security of a hash function with respect to differential attacks. Note also that different attacks can use different numbers of building blocks and relations among them;
- Strictly speaking, given a bound on the expected differential trail probability it is hard to say what it exactly implies for the expected differential probability, since there might be strong differential effects. Note that such MEDTP bounds as the ones for ECHO and SHAvite-3 already take into account much of the differential effect;
- An upper bound on the expected differential trail probability (averaged over some parameters of the primitive) can make only a limited statement about the differential probability taken for the fixed parameters or the maximum differential probability taken over all inputs. At the same time, it is exactly the latter properties which are related to the real-world attack complexities on hash functions.

Each of the limitations mentioned can be seen as an important research problem which can be informally summarized as deriving tighter bounds on more informative properties of larger objects. We think that having some formal arguments against at least basic types of attacks contributes to the confidence in the design.

A hash function that is provided with a sound security analysis, is not necessarily a 'good' function, nor is it a 'bad' function if only little security results are known. The quality of the hash function depends further on other criteria not covered in this classification, such as its software/hardware performance as well as the strength of a concrete instantiation towards more specific attacks, not necessarily of differential nature. Yet, both security reductions and provable resistance against basic differential cryptanalysis guarantee that the hash function has no severe structural weaknesses, and in particular that the design does not suffer weaknesses that can be trivially exploited by cryptanalysts.

Acknowledgements This work has been funded in part by the IAP Program P6/26 BCRYPT of the Belgian State (Belgian Science Policy), in part by the European Commission through the ICT program under contract ICT-2007-216676 ECRYPT II, and in part by the Research Council K.U.Leuven: GOA TENSE. The first author was supported by a Ph.D. Fellowship from the Flemish Research Foundation (FWO-Vlaanderen). The second author is supported by a Postdoctoral Fellowship from the Flemish Research Foundation (FWO-Vlaanderen). The third author is supported by a Ph.D. Fellowship from the Institute for the Promotion of Innovation through Science and Technology in Flanders (IWT-Vlaanderen). Parts of the work were done while the fifth author was with KU Leuven.

A Security Implications of Indifferentiability

Indifferentiability assures that a design is structurally correct, and that it can replace a random oracle without security loss. In particular, it guarantees that, up to a certain degree, the design is secured against any generic attack, like finding preimages, collisions, multicollisions, etc. In Thm. 1, we formally prove a security reduction to derive security against specific attacks from the indifferentiability of a hash function. This proof is based on a personal communication with Joan Daemen.

Theorem 1 *Let $\mathcal{H}$ be a hash function, built on underlying primitive π, and RO be a random oracle, where $\mathcal{H}$ and RO have the same domain and range space. Denote by $\boldsymbol{Adv}^{\text{pro}}_{\mathcal{H}}(q)$ the advantage of distinguishing $(\mathcal{H},\pi)$ from (RO,S), for some simulator S, maximized over all distinguishers $\mathcal{D}$ making at most q queries. Let* atk *be a security property of $\mathcal{H}$. Denote by $\boldsymbol{Adv}^{\text{atk}}_{\mathcal{H}}(q)$ the advantage of breaking $\mathcal{H}$ under* atk*, maximized over all adversaries $\mathcal{A}$ making at most q queries. Then:*

$$\boldsymbol{Adv}^{\text{atk}}_{\mathcal{H}}(q) \leq \boldsymbol{Pr}^{\text{atk}}_{RO}(q) + \boldsymbol{Adv}^{\text{pro}}_{\mathcal{H}}(q), \tag{1}$$

where $\boldsymbol{Pr}^{\text{atk}}_{RO}(q)$ denotes the success probability of a generic attack against $\mathcal{H}$ under atk*, after at most q queries.*

Proof Let $\mathcal{A}$ be any (q,ε) attacker for $\mathcal{H}$ under security notion atk, and assume $\varepsilon > \mathbf{Pr}^{\text{atk}}_{RO}(q)$. We define a distinguisher $\mathcal{D}$ for the indifferentiability of $\mathcal{H}$ as follows: $\mathcal{D}$ makes the same queries as $\mathcal{A}$, and obtains a query history Q (with query answers coming from either $\mathcal{H}$ or *RO*). Next, $\mathcal{D}$ outputs 1 if Q violates security notion atk, and 0 otherwise. Denote by $\mathbf{Adv}^{\text{pro}}_{\mathcal{H}}(\mathcal{D})$ the success probability of $\mathcal{D}$. By definition, we have $\left|\mathbf{Pr}\left(\mathcal{D}^{\mathcal{H},\pi}=1\right) - \mathbf{Pr}\left(\mathcal{D}^{RO,S}=1\right)\right| = \mathbf{Adv}^{\text{pro}}_{\mathcal{H}}(\mathcal{D}) \leq \mathbf{Adv}^{\text{pro}}_{\mathcal{H}}(q)$.

By $E_{\mathcal{H}}$ (resp. E_{RO}), we denote the event that Q defines a set of query pairs that break $\mathcal{H}$ (resp. *RO*) under security notion atk. The distinguisher outputs 1 if and only if Q violates security notion atk, and hence we obtain:

$$\begin{aligned}\mathbf{Pr}\left(\mathcal{D}^{\mathcal{H},\pi}=1\right) &= \mathbf{Pr}\left(\mathcal{D}^{\mathcal{H},\pi}=1 \mid E_{\mathcal{H}}\right)\mathbf{Pr}\left(E_{\mathcal{H}}\right)\\ &\quad + \mathbf{Pr}\left(\mathcal{D}^{\mathcal{H},\pi}=1 \mid \neg E_{\mathcal{H}}\right)\mathbf{Pr}\left(\neg E_{\mathcal{H}}\right)\\ &= 1\cdot\mathbf{Pr}\left(E_{\mathcal{H}}\right) + 0 = \varepsilon.\end{aligned}$$

Similarly, we get $\mathbf{Pr}\left(\mathcal{D}^{RO,S}=1\right) = \mathbf{Pr}\left(E_{RO}\right) = \mathbf{Pr}^{\text{atk}}_{RO}(q)$. As $\varepsilon > \mathbf{Pr}^{\text{atk}}_{RO}(q)$, we consequently derive $\varepsilon \leq \mathbf{Pr}^{\text{atk}}_{RO}(q) + \mathbf{Adv}^{\text{pro}}_{\mathcal{H}}(q)$. This holds for any (q,ε) adversary for $\mathcal{H}$ under security notion atk, which completes the proof. □

B Preservation of Collision Resistance

For the purpose of the analysis of the SHA-3 candidates, we generalize the well-known result by Merkle and Damgård. The result of Thm. 2 differs in three cases: we consider any suffix-free padding, the proof allows for different compression functions in one hash function evaluation, and it includes an optional chopping at the end. Related work can, a.o., be found in [38,40,68,69].

Theorem 2 *Let $l,m,n \in \mathbb{N}$ such that $l \geq n$. Let $\mathsf{pad} : \mathbb{Z}_2^* \to (\mathbb{Z}_2^m)^*$ be a suffix-free padding and let $f,g : \mathbb{Z}_2^l \times \mathbb{Z}_2^m \to \mathbb{Z}_2^l$ be two compression functions. Consider the hash function $\mathcal{H} : \mathbb{Z}_2^* \to \mathbb{Z}_2^n$ defined as follows (cf. Fig. 3), where $h_0 = \mathsf{IV}$ is the initialization vector:*

$$\begin{aligned}\mathcal{H}(M) = h, \text{ where: } (M_1,\ldots,M_k) &= \mathsf{pad}(M),\\ h_i &= f(h_{i-1},M_i) \text{ for } i = 1,\ldots,k-1,\\ h_k &= g(h_{k-1},M_k),\\ h &= \mathsf{chop}_{l-n}(h_k).\end{aligned}$$

Define the function $g' : \mathbb{Z}_2^l \times \mathbb{Z}_2^m \to \mathbb{Z}_2^n$ by $g' = \mathsf{chop}_{l-n} \circ g$. Then, the advantage of finding collisions for $\mathcal{H}$ is upper bounded by the advantage of finding collisions for f or g'. Formally, if f is (t_1,ε_1) collision secure, and g' is (t_2,ε_2) collision secure, then $\mathcal{H}$ is (t,ε) collision secure for $\varepsilon = \varepsilon_1 + \varepsilon_2$, and $t = \min\{t_1,t_2\} - 2(K-1)\tau_f$, where τ_f is the time to evaluate f and K is the maximum length of the messages, in blocks.

Proof Suppose $\mathcal{A}$ is a (t,ε) collision finding attacker for $\mathcal{H}$. We construct collision finding adversaries $\mathcal{B}_1$ and $\mathcal{B}_2$ for f and g, respectively, using the following observation.

Let M,M' be two distinct messages such that $\mathcal{H}(M) = \mathcal{H}(M')$. Let $(M_1,\ldots,M_k)$ be the padded message of M, and $(M'_1,\ldots,M'_{k'})$ be the padded message of M'. Define the intermediate state values h_i,h'_i similarly. A collision on M,M' means that

$$\mathsf{chop}_{l-n}\big(g(h_{k-1},M_k)\big) = \mathsf{chop}_{l-n}\big(g(h'_{k'-1},M'_{k'})\big).$$

Now, if $(h_{k-1},M_k) \neq (h'_{k'-1},M'_{k'})$ this results in a collision for g'. Assume the contrary, and let $j \in \{1,\ldots,\min\{k,k'\}-1\}$ be the minimal index such that $(h_{k-j-1},M_{k-j}) \neq (h'_{k'-j-1},M'_{k'-j})$. We notice that such index j exists: in case $k = k'$ it exists as $M \neq M'$, and in case $k \neq k'$ it exists as the padding rule is suffix-free. By definition of the index j, we have $h_{k-j} = h'_{k'-j}$, and in particular we obtain a collision for f:

$$f(h_{k-j-1},M_{k-j}) = h_{k-j} = h'_{k'-j} = f(h'_{k'-j-1},M'_{k'-j}).$$

Both $\mathcal{B}_1,\mathcal{B}_2$ follow this procedure. If M,M' define a collision for f, $\mathcal{B}_1$ outputs this collision. Similarly for $\mathcal{B}_2$ and g'. Both adversaries work in time at most $t + 2(K-1)\tau_f$, from which we deduce $t \geq \min\{t_1,t_2\} - 2(K-1)\tau_f$. The messages M,M' define a collision for f or g'. Thus, we obtain $\varepsilon \leq \varepsilon_1 + \varepsilon_2$. □

In case the design is based on the compression function f only (but it may still include the chopping), the above result can easily be simplified to $\mathbf{Adv}^{\text{gcol}}_{\mathcal{H}}(\mathcal{A}) \leq \mathbf{Adv}^{\text{gcol}}_{f'}(\mathcal{B}_1)$, where $f' = \mathsf{chop}_{l-n} \circ f$. Observe that this result also holds if $l = n$, and in particular, the basic theorems of Merkle and Damgård are covered as well. We note that Thm. 2 can be generalized arbitrarily, e.g. to more different compression functions, but for the purpose of this paper, the mentioned generalization of the Merkle-Damgård structure suffices.

References

1. Andreeva, E., Mennink, B., Preneel, B.: On the indifferentiability of the Grøstl hash function. In: SCN '10. LNCS, vol. 6280, pp. 88–105. Springer-Verlag, Berlin (2010)
2. Andreeva, E., Mennink, B., Preneel, B.: Security reductions of the second round SHA-3 candidates. In: ISC '10. LNCS, vol. 6531, pp. 39–53. Springer-Verlag, Berlin (2010)
3. Andreeva, E., Mennink, B., Preneel, B.: The parazoa family: Generalizing the sponge hash functions. Cryptology ePrint Archive, Report 2011/028 (2011)

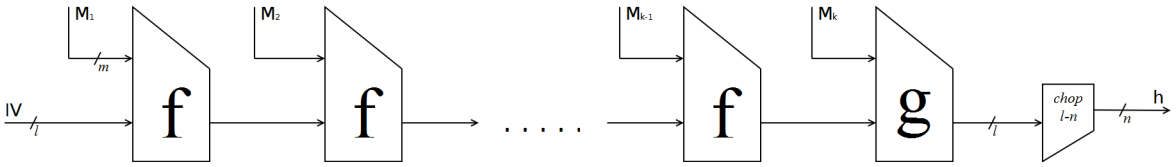

Fig. 3 A generalized Merkle-Damgård structure. f, g are two compression functions, and chop_{l-n} chops off $l-n$ bits of the state.

4. Andreeva, E., Neven, G., Preneel, B., Shrimpton, T.: Seven-property-preserving iterated hashing: ROX. In: ASIACRYPT '07. LNCS, vol. 4833, pp. 130–146. Springer-Verlag, Berlin (2007)
5. Andreeva, E., Luykx, A., Mennink, B.: Provable security of BLAKE with non-ideal compression function. Cryptology ePrint Archive, Report 2011/620 (2011)
6. Andreeva, E., Mennink, B., Preneel, B., Škrobot, M.: Security analysis and comparison of the SHA-3 Finalists BLAKE, Grøstl, JH, Keccak, and Skein (2012)
7. Aumasson, J.P.: Practical distinguisher for the compression function of Blue Midnight Wish (2010)
8. Aumasson, J., Henzen, L., Meier, W., Phan, R.: SHA-3 proposal BLAKE (2009), submission to NIST's SHA-3 competition
9. Aumasson, J.P., Phan, R.: On the cryptanalysis of the hash function Fugue: Partitioning and inside-out distinguishers. Inf. Process. Lett. 111(11), 512–515 (2011)
10. Aumasson, J.P.: On the pseudorandomness of Shabal's keyed permutation (2009)
11. Aumasson, J.P., Mashatan, A., Meier, W.: More on Shabal's permutation (2009)
12. Bellare, M., Canetti, R., Krawczyk, H.: Keying hash functions for message authentication. In: CRYPTO '96. LNCS, vol. 1109, pp. 1–15. Springer-Verlag, Berlin (1996)
13. Bellare, M., Kohno, T., Lucks, S., Ferguson, N., Schneier, B., Whiting, D., Callas, J., Walker, J.: Provable security support for the skein hash family (2009)
14. Benadjila, R., Billet, O., Gilbert, H., Macario-Rat, G., Peyrin, T., Robshaw, M., Seurin, Y.: SHA-3 Proposal: ECHO (2009), submission to NIST's SHA-3 competition
15. Bernstein, D.: CubeHash specification (2009), submission to NIST's SHA-3 competition
16. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: Sponge functions (2007), eCRYPT Hash Workshop
17. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: On the indifferentiability of the sponge construction. In: EUROCRYPT '08. LNCS, vol. 4965, pp. 181–197. Springer-Verlag, Berlin (2008)
18. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: The KECCAK sponge function family (2009), submission to NIST's SHA-3 competition
19. Bhattacharyya, R., Mandal, A., Nandi, M.: Indifferentiability characterization of hash functions and optimal bounds of popular domain extensions. In: INDOCRYPT '09. LNCS, vol. 5922, pp. 199–218. Springer-Verlag, Berlin (2009)
20. Bhattacharyya, R., Mandal, A., Nandi, M.: Security analysis of the mode of JH hash function. In: FSE '10. LNCS, vol. 6147, pp. 168–191. Springer-Verlag, Berlin (2010)
21. Biham, E., Dunkelman, O.: A framework for iterative hash functions – HAIFA. Cryptology ePrint Archive, Report 2007/278 (2007)
22. Biham, E., Dunkelman, O.: The SHAvite-3 Hash Function (2009), submission to NIST's SHA-3 competition
23. Biham, E., Shamir, A.: Differential cryptanalysis of DES-like cryptosystems. In: CRYPTO '90. LNCS, vol. 537, pp. 2–21. Springer-Verlag, Berlin (1991)
24. Black, J., Cochran, M., Shrimpton, T.: On the impossibility of highly-efficient blockcipher-based hash functions. In: EUROCRYPT '05. LNCS, vol. 3494, pp. 526–541. Springer-Verlag, Berlin (2005)
25. Black, J., Rogaway, P., Shrimpton, T.: Black-box analysis of the block-cipher-based hash-function constructions from PGV. In: CRYPTO '02. LNCS, vol. 2442, pp. 320–335. Springer-Verlag, Berlin (2002)
26. Bouillaguet, C., Fouque, P., Leurent, G.: Security analysis of SIMD. In: SAC '10. LNCS, vol. 6544, pp. 351–368. Springer-Verlag, Berlin (2011)
27. Bouillaguet, C., Fouque, P.: Practical hash functions constructions resistant to generic second preimage attacks beyond the birthday bound (2010), submitted to Information Processing Letters
28. Bresson, E., Canteaut, A., Chevallier-Mames, B., Clavier, C., Fuhr, T., Gouget, A., Icart, T., Misarsky, J.F., Naya-Plasencia, M., Paillier, P., Pornin, T., Reinhard, J., Thuillet, C., Videau, M.: Shabal, a Submission to NIST's Cryptographic Hash Algorithm Competition (2009), submission to NIST's SHA-3 competition
29. Burr, W.E.: NIST Comments on Cryptanalytic Attacks on SHA-1, `http://csrc.nist.gov/groups/ST/hash/statement.html`
30. Chang, D., Lee, S., Nandi, M., Yung, M.: Indifferentiable security analysis of popular hash functions with prefix-free padding. In: ASIACRYPT '06. LNCS, vol. 4284, pp. 283–298. Springer-Verlag, Berlin (2006)
31. Chang, D., Nandi, M., Yung, M.: Indifferentiability of the hash algorithm BLAKE. Cryptology ePrint Archive, Report 2011/623 (2011)
32. Coron, J., Dodis, Y., Malinaud, C., Puniya, P.: Merkle-Damgård revisited: How to construct a hash function. In: CRYPTO '05. LNCS, vol. 3621, pp. 430–448. Springer-Verlag, Berlin (2005)
33. Daemen, J., Lamberger, M., Pramstaller, N., Rijmen, V., Vercauteren, F.: Computational aspects of the expected differential probability of 4-round AES and AES-like ciphers. Computing 85(1-2), 85–104 (2009)
34. Daemen, J., Rijmen, V.: The Design of Rijndael: AES - The Advanced Encryption Standard. Springer-Verlag (2002)
35. Daemen, J., Rijmen, V.: Probability Distributions of Correlation and Differentials in Block Ciphers. Journal of Mathematical Cryptology 1(3), 221–242 (2007)
36. Daemen, J., Van Assche, G.: Differential propagation analysis of Keccak (2012), pre-proceedings of FSE 2012
37. Daemen, J., Rijmen, V.: The wide trail design strategy. In: IMA International Conference '01. LNCS, vol. 2260, pp. 222–238. Springer-Verlag, Berlin (2001)
38. Damgård, I.: A design principle for hash functions. In: CRYPTO '89. LNCS, vol. 435, pp. 416–427. Springer-Verlag, Berlin (1990)
39. De Cannière, C., Sato, H., Watanabe, D.: Hash Function Luffa (2009), submission to NIST's SHA-3 competition
40. Dodis, Y., Puniya, P.: Getting the best out of existing hash functions; or what if we are stuck with SHA? In: ACNS '08. LNCS, vol. 5037, pp. 156–173. Springer-Verlag, Berlin (2008)
41. Dodis, Y., Ristenpart, T., Shrimpton, T.: Salvaging merkle-damgård for practical applications. In: EUROCRYPT '09. LNCS, vol. 5479, pp. 371–388. Springer-Verlag, Berlin (2009)
42. Ferguson, N., Lucks, S., Schneier, B., Whiting, D., Bellare, M., Kohno, T., Callas, J., Walker, J.: The Skein Hash Function Family (2009), submission to NIST's SHA-3 competition
43. Ferguson, N., Lucks, S., Schneier, B., Whiting, D., Bellare, M., Kohno, T., Callas, J., Walker, J.: Engineering comparison of SHA-3 candidates (2010), `http://www.skein-hash.info/sha3-engineering`

44. FIPS: Advanced Encryption Standard: Publication 197, National Bureau of Standards, U.S. Department of Commerce (2001)
45. FIPS: Secure Hash Standard: Publication 180-3, National Bureau of Standards, U.S. Department of Commerce (2008)
46. Fleischmann, E., Forler, C., Gorski, M.: Classification of the SHA-3 candidates. Cryptology ePrint Archive, Report 2008/511 (2008)
47. Fouque, P., Stern, J., Zimmer, S.: Cryptanalysis of tweaked versions of SMASH and reparation. In: SAC '08. LNCS, vol. 5381, pp. 136–150. Springer-Verlag, Berlin (2009)
48. Gauravaram, P., Bagheri, N.: ECHO compression function is not indifferentiable from a FIL-RO (2010)
49. Gauravaram, P., Knudsen, L., Bagheri, N., Wei, L.: Improved security analysis of Fugue-256. In: ACISP 2011. LNCS, vol. 6812, pp. 428–432. Springer-Verlag, Berlin (2011)
50. Gauravaram, P., Knudsen, L., Matusiewicz, K., Mendel, F., Rechberger, C., Schläffer, M., Thomsen, S.: Grøstl – a SHA-3 candidate (2009), submission to NIST's SHA-3 competition
51. Gligoroski, D., Klima, V., Knapskog, S., El-Hadedy, M., Amundsen, J., Mjølsnes, S.: Cryptographic Hash Function BLUE MIDNIGHT WISH (2009), submission to NIST's SHA-3 competition
52. Gong, Z., Lai, X., Chen, K.: A synthetic indifferentiability analysis of some block-cipher-based hash functions. Des. Codes Cryptography 48(3), 293–305 (2008)
53. Halevi, S., Hall, W., Jutla, C.: The Hash Function "Fugue" (2009), submission to NIST's SHA-3 competition
54. Halevi, S., Krawczyk, H.: Strengthening digital signatures via randomized hashing. In: CRYPTO '06. LNCS, vol. 4117, pp. 41–59. Springer-Verlag, Berlin (2006)
55. Keliher, L., Sui, J.: Exact maximum expected differential and linear probability for 2-round Advanced Encryption Standard (AES). IET Inf. Secur. 1(2), 53–57 (2007)
56. Kelsey, J., Schneier, B.: Second preimages on n-bit hash functions for much less than 2^n work. In: EUROCRYPT '05. LNCS, vol. 3494, pp. 474–490. Springer-Verlag, Berlin (2005)
57. Khovratovich, D., Naya-Plasencia, M., Röck, A., Schläffer, M.: Cryptanalysis of *Luffa* v2 components. In: SAC '10. LNCS, vol. 6544, pp. 388–409. Springer-Verlag, Berlin (2011)
58. Knudsen, L., Matusiewicz, K., Thomsen, S.: Observations on the Shabal keyed permutation (2009)
59. Kobayashi, K., Ikegami, J., Matsuo, S., Sakiyama, K., Ohta, K.: Evaluation of hardware performance for the SHA-3 candidates using SASEBO-GII. Cryptology ePrint Archive, Report 2010/010 (2010)
60. Küçük, Ö.: The Hash Function Hamsi (2009), submission to NIST's SHA-3 competition
61. Küçük, Ö.: Design and Analysis of Cryptographic Hash Functions. Ph.D. thesis, Katholieke Universiteit Leuven, Leuven (2012)
62. Lai, X., Massey, J.: Hash function based on block ciphers. In: EUROCRYPT '92. LNCS, vol. 658, pp. 55–70. Springer-Verlag, Berlin (1992)
63. Lee, J., Hong, D.: Collision resistance of the JH hash function. Cryptology ePrint Archive, Report 2011/019 (2011)
64. Leurent, G., Bouillaguet, C., Fouque, P.: SIMD is a Message Digest (2009), submission to NIST's SHA-3 competition
65. Lucks, S.: A failure-friendly design principle for hash functions. In: ASIACRYPT '05. LNCS, vol. 3788, pp. 474–494. Springer-Verlag, Berlin (2005)
66. Luo, Y., Gong, Z., Duan, M., Zhu, B., Lai, X.: Revisiting the indifferentiability of PGV hash functions. Cryptology ePrint Archive, Report 2009/265 (2009)
67. Maurer, U., Renner, R., Holenstein, C.: Indifferentiability, impossibility results on reductions, and applications to the random oracle methodology. In: TCC '04. LNCS, vol. 2951, pp. 21–39. Springer-Verlag, Berlin (2004)
68. Merkle, R.: One way hash functions and DES. In: CRYPTO '89. LNCS, vol. 435, pp. 428–446. Springer-Verlag, Berlin (1990)
69. Nandi, M.: Characterizing padding rules of MD hash functions preserving collision security. In: ACISP '09. LNCS, vol. 5594, pp. 171–184. Springer-Verlag, Berlin (2009)
70. National Institute for Standards and Technology: Announcing request for candidate algorithm nominations for a new cryptographic hash algorithm (SHA3) family (November 2007)
71. Novotney, P.: Distinguisher for Shabal's permutation function. Cryptology ePrint Archive, Report 2010/398 (2010)
72. Preneel, B., Govaerts, R., Vandewalle, J.: Hash functions based on block ciphers: A synthetic approach. In: CRYPTO '93. LNCS, vol. 773, pp. 368–378. Springer-Verlag, Berlin (1993)
73. Rogaway, P.: Formalizing human ignorance. In: VIETCRYPT '06. LNCS, vol. 4341, pp. 211–228. Springer-Verlag, Berlin (2006)
74. Rogaway, P., Shrimpton, T.: Cryptographic hash-function basics: Definitions, implications, and separations for preimage resistance, second-preimage resistance, and collision resistance. In: FSE '04. LNCS, vol. 3017, pp. 371–388. Springer-Verlag, Berlin (2004)
75. Rogaway, P., Steinberger, J.: Security/efficiency tradeoffs for permutation-based hashing. In: EUROCRYPT '08. LNCS, vol. 4965, pp. 220–236. Springer-Verlag, Berlin (2008)
76. Stam, M.: Beyond uniformity: Better security/efficiency tradeoffs for compression functions. In: CRYPTO '08. LNCS, vol. 5157, pp. 397–412. Springer-Verlag, Berlin (2008)
77. Stam, M.: Blockcipher-based hashing revisited. In: FSE '09. LNCS, vol. 5665, pp. 67–83. Springer-Verlag, Berlin (2009)
78. Steinberger, J.: Stam's collision resistance conjecture. In: EUROCRYPT '10. LNCS, vol. 6110, pp. 597–615. Springer-Verlag, Berlin (2010)
79. Tillich, S., Feldhofer, M., Kirschbaum, M., Plos, T., Schmidt, J., Szekely, A.: High-speed hardware implementations of BLAKE, Blue Midnight Wish, CubeHash, ECHO, Fugue, Grøstl, Hamsi, JH, Keccak, Luffa, Shabal, SHAvite-3, SIMD, and Skein. Cryptology ePrint Archive, Report 2009/510 (2009)
80. Van Assche, G.: A rotational distinguisher on Shabal's keyed permutation and its impact on the security proofs (2010)
81. Wang, X., Yin, Y., Yu, H.: Finding collisions in the full SHA-1. In: CRYPTO '05. LNCS, vol. 3621, pp. 17–36. Springer-Verlag, Berlin (2005)
82. Wang, X., Yu, H.: How to break MD5 and other hash functions. In: EUROCRYPT '05. LNCS, vol. 3494, pp. 19–35. Springer-Verlag, Berlin (2005)
83. Wu, H.: The Hash Function JH (2009), submission to NIST's SHA-3 competition