Security Analysis and Comparison of the SHA-3 Finalists BLAKE, Grøstl, JH, Keccak, and Skein

Abstract. In 2007, the US National Institute for Standards and Technology announced a call for the design of a new cryptographic hash algorithm in response to the vulnerabilities identified in widely employed hash functions, such as MD5 and SHA-1. NIST received many submissions, 51 of which got accepted to the first round. At present, 5 candidates are left in the third round of the competition. An important criterion in the selection process is the SHA-3 hash function security and more concretely, the possible reductions of the hash function security to the security of its underlying building blocks. At NIST's second SHA-3 Candidate Conference 2010, Andreeva et al. provided a provable security classification of the second round SHA-3 candidates in the ideal model. In this work, we revisit this classification for the five SHA-3 finalists. We evaluate recent provable security results on the candidates, and resolve remaining open problems for Grøstl, JH, and Skein.

Keywords. SHA-3, security classification, (second) preimage resistance, collision resistance, indifferentiability.

1 Introduction

Hash functions are a building block for numerous cryptographic applications. In 2004 a series of attacks by Wang et al. [47, 48] showed security vulnerabilities in the design of the widely adopted hash function SHA-1. In response, 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 hash algorithm [38]. The call prescribes that SHA-3 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. Five candidates have reached the third and final round of the competition: BLAKE [7], Grøstl [28], JH [49], Keccak [10], and Skein [25]. These candidates are under active evaluation by the cryptographic community. As a result of comparative analysis, several classifications of the SHA-3 candidates, mostly concentrated on hardware performance, appeared in the literature [24, 26, 46, 31]. At NIST's second SHA-3 Candidate Conference 2010, Andreeva et al. [4, 5] provided a classification based on the specified by NIST security criteria. Below we recall the security requirements by NIST in their call for the SHA-3 hash function.

NIST Security Requirements. The future SHA-3 hash function is required to satisfy the following security requirements [38]: `(i)` at least one variant of the hash function must securely support HMAC and randomized hashing. Next, 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)` it 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 Contributions. We revisit the provable security classification of Andreeva et al. [4, 5], focussing on the five remaining SHA-3 finalists. More concretely, we reconsider the preimage, second preimage and collision resistance (security requirements `(ii)-(iv)`) for the $n = 256$ and $n = 512$ variants of the five candidates. We also include their indifferentiability security results. The security analysis in this work 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.

In our updated security classification of the SHA-3 finalists, we include the recent full security analysis of BLAKE by Andreeva et al. and Chang et al. [2, 20], and the collision security result of JH by Lee and Hong [33]. Despite these recent advances, there still remain open questions in the earlier security analysis and classification of [4, 5]. The main contribution of this work is to address these questions. More concretely, we do so by either providing new security results or improving some of the existing security bounds. We list our findings for the relevant hash functions below and refer to Table 1 for the summary of all results.

- Grøstl. We analyze Grøstl with respect to its second preimage security due to the lack of an optimal security result as indicated in [4, 5]. While optimal collision and preimage security are achieved following a property-preservation argument, this is not true for the second preimage security. Another way (than property-preservation) to derive security bounds for hash function properties is via an indifferentiability result (Thm. 2 in [4, 5]). Following this approach, an approximately 128-bit and 256-bit second preimage resistance bound is obtained, where the output size of the Grøstl hash function is 256 or 512 bits, respectively. This result is unfortunately not optimal. In this work we take a different approach to improve these bounds, and we provide a direct second preimage security proof for the Grøstl hash function. Our proof is realized in the ideal permutation model and indicates that Grøstl, in addition to collision and preimage security, is also optimally ($(256-L)$-bit and $(512-L)$-bit, respectively) second preimage secure, where 2^L is the length of the first preimage in blocks;
- JH. The existing bounds on JH for second and preimage security are derived via the indifferentiability result and are not optimal; approximately 170-bit security for both the 256 and 512 variants. To improve these results, we follow the direct approach and derive bounds for both security properties in the ideal permutation model. As a result we achieve optimal 256-bit security for the 256 variant of the hash function. The new bound for the 512 variant is still not optimal (as is the existing bound), but improved to 256-bit security;
- Skein. By the implications of the existing indifferentiability results of Skein we can directly conclude an optimal 256-bit second preimage security for the 256 version of the hash function. This is however not true for the 512 version, which offers only 256-bit security following the indifferentiability argument. We, thus, analyze the generic second preimage security of Skein in the ideal block cipher model and obtain optimal bounds for both its versions, confirming the second preimage result for the 256 version and optimally improving the bound for the 512 version.

Table 1. Security results of the SHA-3 finalists. Here, l and m denote the chaining value and the message input sizes, respectively. The last four columns of both tables correspond to the preimage, second preimage, collision, and indifferentiability security in bits. Regarding second preimage resistance, we assume that the first preimage is of length 2^L blocks. The results in **bold** are presented in this work. For a more detailed summary we refer to Table 2.

	l	m	pre	sec	coll	indiff
BLAKE-256	256	512	256	256	128	128
Grøstl-256	512	512	256	**256–*L***	128	128
JH-256	1024	512	**256**	**256**	128	170
Keccak-256	1600	1088	256	256	128	256
Skein-256	512	512	256	256	128	256
NIST's requirements [38]			256	256–L	128	—

	l	m	pre	sec	coll	indiff
BLAKE-512	512	1024	512	512	256	256
Grøstl-512	1024	1024	512	**512–*L***	256	256
JH-512	1024	512	**256**	**256**	256	170
Keccak-512	1600	576	512	512	256	512
Skein-512	512	512	512	**512**	256	256
NIST's requirements [38]			512	512–L	256	—

The results of Table 1 show that all candidates, with the exception of the (second) preimage security of JH-512, achieve optimal collision, second and preimage security for both their 256 and 512 variants. The optimal results refer to the general iterative structure of all the algorithms. The analysis in all cases is performed in the ideal setting. But more importantly, we claim that the provided comparison is sufficiently fair due to the fact that the ideality assumption is hypothesized on basic underlying primitives, such as block ciphers and permutations, as opposed to higher level compression function building blocks.

On the other hand, while optimality results hold for the five the hash function finalists, the security of their compression functions again in the ideal model differs. The security here varies from trivially insecure compression functions for JH and Keccak to optimally secure ones for BLAKE, Grøstl and Skein. We want to note that the latter remark does not reflect any security criteria indicated in the security requirements of NIST. In addition to the classical notions of collision, second and preimage security, we also investigate the notion of indifferentiability [36]. Indifferentiability encompasses structural attacks, such as the length extension attack in single round interactive protocols [40], and is therefore an important security criteria satisfied by all five candidates. We include the indifferentiability notion not only because it is relevant by itself, but it is also an important tool to derive further security results. The JH candidate offers 170-bit indifferentiability security for both its variants, and Skein offers 256-bit security for both its variants. BLAKE and Grøstl offer 128-bit and 256-bit security for their respective 256 and 512 variants. Keccak provides higher indifferentiability guarantees: 256-bit and 512-bit, respectively, and that is achieved by increasing the iterated state size to 1600 bits as compared to sizes from 256 bits to 1024 bits for the other hash function candidates.

Outline. Section 2 briefly covers the notation, and the basic principles of hash function design. In Sects. 3-7, we consider the five SHA-3 finalists from a provable security point of view. We give a high level algorithmic description of each hash function, and discuss the existing and new security results. The revisited security classification, including the newly found results on Grøstl, JH, and Skein, is given in Table 2. We conclude the paper with Sect. 8 and give some final remarks on the security comparison.

2 Preliminaries

For $n \in \mathbb{N}$, 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. For two bit strings x, y, $x\|y$ denotes their concatenation and $x \oplus y$ their bitwise XOR. For $m, n \in \mathbb{N}$ we denote by $\langle m \rangle_n$ the encoding of m as an n-bit string. The function $\mathsf{chop}_n(x)$ takes the n leftmost bits of a bit string x. We denote by $\mathrm{Func}(m, n)$ the set of all functions $f : \mathbb{Z}_2^m \to \mathbb{Z}_2^n$. A random oracle [9] is a function which provides a random output for each new query. A random m-to-n-bit function is a function sampled uniformly at random from $\mathrm{Func}(m, n)$. A random primitive will also be called "ideal". The set of functions Func may be restricted, for instance to contain block ciphers or permutations only.

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 Preimage, Second Preimage and Collision Security

In our analysis we model the adversary $\mathcal{A}$ as a probabilistic algorithm with oracle access to a randomly sampled primitive $\mathcal{P} \xleftarrow{\$} \mathrm{Prims}$. The set Prims depends on the hash function to be analyzed. We consider information-theoretic adversaries only. This type of adversary has

unbounded computational power, and its complexity is measured by the number of queries made to his oracle. The adversary can make queries to $\mathcal{P}$. These queries are stored in a query history $\mathcal{Q}$ as indexed elements. In the remainder, we assume that $\mathcal{Q}$ always contains the queries required for the attack and that the adversary never makes queries to which it knows the answer in advance.

Let $F : \mathbb{Z}_2^p \to \mathbb{Z}_2^n$ for $p \geq n$ be a compressing function instantiated with a randomly chosen primitive $\mathcal{P} \xleftarrow{\$}$ Prims. Throughout, F will either denote the compression function f or the hash function $\mathcal{H}$ specification of one of the SHA-3 finalists. For the preimage and second preimage security analysis in this work, we consider the notions of everywhere preimage and second preimage resistance [41], which guarantees security on every range (resp. domain) point.

Definition 1. *Let $p, n \in \mathbb{N}$ with $p \geq n$ and let $F : \mathbb{Z}_2^p \to \mathbb{Z}_2^n$ be a compressing function using primitive $\mathcal{P} \in$ Prims. The advantage of an everywhere preimage finding adversary $\mathcal{A}$ is defined as*

$$\boldsymbol{Adv}_F^{\mathrm{epre}}(\mathcal{A}) = \max_{y \in \mathbb{Z}_2^n} \boldsymbol{Pr}\left(\begin{array}{c} \mathcal{P} \xleftarrow{\$} \text{Prims},\ z \leftarrow \mathcal{A}^{\mathcal{P}}(y)\ : \\ F(z) = y \end{array}\right).$$

We define by $\boldsymbol{Adv}_F^{\mathrm{epre}}(q)$ the maximum advantage of any adversary making q queries to its oracles.

Definition 2. *Let $p, n \in \mathbb{N}$ with $p \geq n$ and let $F : \mathbb{Z}_2^p \to \mathbb{Z}_2^n$ be a compressing function using primitive $\mathcal{P} \in$ Prims. Let $\lambda \leq p$. The advantage of an everywhere second preimage finding adversary $\mathcal{A}$ is defined as*

$$\boldsymbol{Adv}_F^{\mathrm{esec}[\lambda]}(\mathcal{A}) = \max_{z' \in \mathbb{Z}_2^{\lambda}} \boldsymbol{Pr}\left(\begin{array}{c} \mathcal{P} \xleftarrow{\$} \text{Prims},\ z \leftarrow \mathcal{A}^{\mathcal{P}}(z')\ : \\ z \neq z' \ \wedge\ F(z) = F(z') \end{array}\right).$$

We define by $\boldsymbol{Adv}_F^{\mathrm{esec}[\lambda]}(q)$ the maximum advantage of any adversary making q queries to its oracles.

If F is a compression function, we require $\lambda = p$. Note that, while the length of the first preimage is of 2^L blocks following NIST's security requirements, here we bound the length by λ bits. This translates to $2^L \approx \lambda/m$, where m is the size of the message block.

We define the collision security of F as follows.

Definition 3. *Let $p, n \in \mathbb{N}$ with $p \geq n$ and let $F : \mathbb{Z}_2^p \to \mathbb{Z}_2^n$ be a compressing function using primitive $\mathcal{P} \in$ Prims. Fix a constant $h_0 \in \mathbb{Z}_2^n$. The advantage of a collision finding adversary $\mathcal{A}$ is defined as*

$$\boldsymbol{Adv}_F^{\mathrm{col}}(\mathcal{A}) = \boldsymbol{Pr}\left(\begin{array}{c} \mathcal{P} \xleftarrow{\$} \text{Prims},\ z, z' \leftarrow \mathcal{A}^{\mathcal{P}}\ : \\ z \neq z' \ \wedge\ F(z) \in \{F(z'), h_0\} \end{array}\right).$$

We define by $\boldsymbol{Adv}_F^{\mathrm{col}}(q)$ the maximum advantage of any adversary making q queries to its oracles.

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 [41]. Formally, we have

[1] Kelsey and Schneier [30] 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 ($l \gg n$) remain mostly unaffected due to their increased internal state (see the remark on Thm. 3).

$\Omega(q^2/2^n) = \mathbf{Adv}_F^{\mathrm{col}} = O(1)$, $\Omega(q/2^n) = \mathbf{Adv}_F^{\mathrm{esec}[\lambda]} \leq \mathbf{Adv}_F^{\mathrm{col}}$, and $\Omega(q/2^n) = \mathbf{Adv}_F^{\mathrm{epre}} = O(1)$. 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 Indifferentiability

The indifferentiability framework introduced by Maurer et al. [36] is an extension of the classical notion of indistinguishability; it ensures that a hash function has no structural defects. We denote the indifferentiability security of a hash function $\mathcal{H}$ by $\mathbf{Adv}_{\mathcal{H}}^{\mathrm{pro}}$, maximized over all distinguishers making at most q queries of maximal length $K \geq 0$ message blocks to their oracles. We refer to Coron et al. [21] for a formal definition. An indifferentiability bound guarantees security of the hash function against specific attacks. Although recent results by Ristenpart et al. [40] show that indifferentiability does not capture all properties of a random oracle, indifferentiability still remains the best way to rule out structural attacks for a large class of hash function applications.

It has been demonstrated in [4, 5] that

$$\mathbf{Adv}_{\mathcal{H}}^{\mathrm{atk}} \leq \mathbf{Pr}_{RO}^{\mathrm{atk}} + \mathbf{Adv}_{\mathcal{H}}^{\mathrm{pro}} \tag{1}$$

for any security notion atk, where $\mathbf{Pr}_{RO}^{\mathrm{atk}}$ denotes the success probability of a generic attack against $\mathcal{H}$ under atk and RO is an ideal function with the same domain and range space as $\mathcal{H}$.

2.3 Compression Function Design Strategies

A common way to build compression functions is to base it on a block cipher [17, 39, 44], or on a (limited number of) permutation(s) [16, 42, 43]. Preneel et al. [39] analyzed and categorized 64 block cipher based compression functions. Twelve of them are formally proven secure by Black et al. [17]. These results have been generalized by Stam [44]. By 'PGVx' we denote the x-th type compression function of [39].

In the context of permutation based compression functions, Black et al. [16] 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 [42], Stam [43] and Steinberger [45] 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.

2.4 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* [32]: on input of an initialization vector IV and a message M, the iterated hash function $\mathcal{H}^f$ based on the compression function f, applies a padding mechanism pad to M resulting in $(M_1, \ldots, M_k)$, and proceeds 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 [22, 37]. Each of the five remaining candidates is based on this design, possibly followed by a final transformation (FT), and/or a chop-function[2].

[2] The chop-function is not considered to be (a part of) a final transformation. It refers to the chopping off or discarding a specified number of bits from the output.

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 designs the message blocks are compressed along with specific counters or tweaks, that 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 [32]) preserves collision resistance [22, 37]. This result has been generalized in [4, 5]: informally, this preservation result also holds if the iteration is finalized by a distinct compression function and/or the chop-function. Similarly, everywhere preimage resistance is preserved. Other security properties, such as second preimage resistance, are however not preserved in the MD design [6]. It is also proven that the MD design with a suffix-free padding need not necessarily be indifferentiable [21]. 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 [21, 29, 35].

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 [19, 21, 29, 35]. Everywhere preimage resistance is preserved by the MD design with prefix-free padding. Security notions such as 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 by Biham and Dunkelman [15]. 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 and the indifferentiability results of Coron et al. [21] carry over. For the HAIFA design, these indifferentiability results are improved by Bhattacharyya et al. in [13]. Furthermore, the HAIFA construction is proven optimally secure against second preimage attacks if the underlying compression function is assumed to behave like an ideal primitive [18].
Wide-pipe design. In the wide-pipe design [34], 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 [21], the MD construction with a distinct final transformation and/or chopping at the end is indifferentiable from a random oracle.
Sponge functions. The sponge hash function design is a particular design by Bertoni et al. [12]. It has been generalized by Andreeva et al. [1]. Two SHA-3 finalists are known to be sponge(-like) functions, JH and Keccak. We note that both hash functions can also be described in terms of the chop-MD construction.

3 BLAKE

The **BLAKE** hash function [7] is a HAIFA construction. The message blocks are accompanied with a HAIFA-counter, and the function employs a suffix- and prefix-free padding rule. The underlying compression function f is based on a block cipher $E : \mathbb{Z}_2^{2l} \times \mathbb{Z}_2^m \to \mathbb{Z}_2^{2l}$.[3] It moreover

[3] As observed in [7, 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.

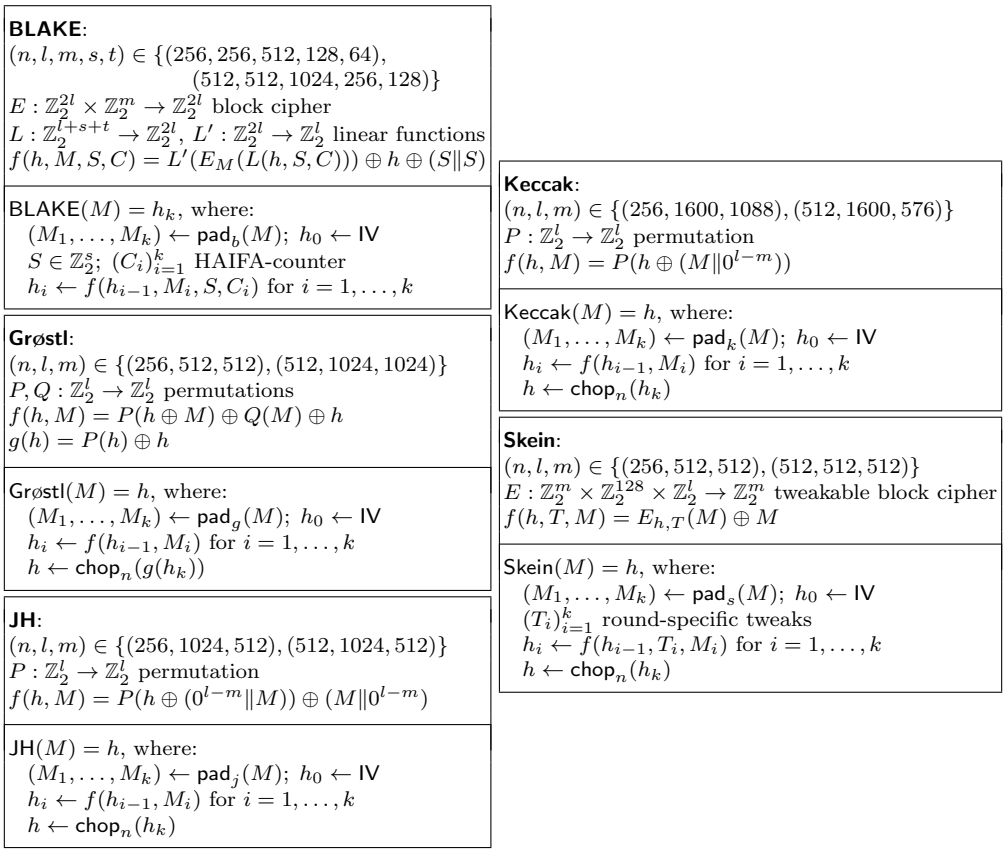

Fig. 1. In all algorithmic descriptions, IV denotes an initialization vector, h denotes state values, M denotes message blocks, S denotes a (fixed) salt, C denotes a counter and T denotes a tweak. The padding rules employed by the functions are summarized in Fig. 2. The functions L, L' underlying BLAKE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

employs an injective linear function L, and a linear function L' that XORs the first and second halves of the input. The BLAKE hash function design is given in Fig. 1.

As the mode of operation of BLAKE is based on the HAIFA structure, all security properties regarding this type (cf. Sect. 2.4) hold [15], provided the compression function is assumed to be ideal. However, as independently shown by Andreeva et al. [2] and Chang et al. [20], 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. This attack has invalidated the results on BLAKE reported in the second round SHA-3 classification of [4, 5].

The security results have been reconfirmed by Andreeva et al. [2] in the ideal cipher model. Firstly, the authors prove optimal security bounds on the compression function, $\mathbf{Adv}_f^{\mathrm{epre}} = \Theta(q/2^n)$ and $\mathbf{Adv}_f^{\mathrm{col}} = \Theta(q^2/2^n)$. 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). The BLAKE mode of operation preserves collision resistance and everywhere preimage resistance due to which we obtain $\mathbf{Adv}_{\mathcal{H}}^{\mathrm{col}} = \Theta(q^2/2^n)$ and $\mathbf{Adv}_{\mathcal{H}}^{\mathrm{epre}} = \Theta(q/2^n)$. The hash function is moreover proven optimally second preimage resistance in the ideal cipher model by Andreeva et al. [2], which gives $\mathbf{Adv}_{\mathcal{H}}^{\mathrm{esec}[\lambda]} = \Theta(q/2^n)$. Finally, the BLAKE hash function is reproven indifferentiable from a random oracle up to bound $\Theta((Kq)^2/2^n)$, this time under the assumption that the underlying block cipher is assumed to be ideal [2, 20].

$$
\begin{aligned}
&\mathsf{BLAKE}: \mathsf{pad}_b(M) = M\|10^{-|M|-t-2 \bmod m}1\|\langle|M|\rangle_t,\\
&\mathsf{Grøstl}: \mathsf{pad}_g(M) = M\|10^{-|M|-65 \bmod l}\|\langle\lceil(|M|+65)/l\rceil\rangle_{64},\\
&\mathsf{JH}: \mathsf{pad}_j(M) = M\|10^{383+(-|M| \bmod m)}\|\langle|M|\rangle_{128},\\
&\mathsf{Keccak}: \mathsf{pad}_k(M) = M\|10^{-|M|-2 \bmod m}1,\\
&\mathsf{Skein}: \mathsf{pad}_s(M) = M'\|0^{(-|M'| \bmod m)+m}, \text{where } M' = \begin{cases} M \text{ if } |M| \equiv 0 \bmod 8,\\ M\|1\|0^{-|M|-1 \bmod 8} \text{ otherwise.}\end{cases}
\end{aligned}
$$

Fig. 2. The padding rules of all SHA-3 hash function candidates 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. In our interpretation the complete padding rule of BLAKE and Skein is additionally defined by a counter or tweak (as explained in Sects. 3 and 7).

4 Grøstl

The **Grøstl** hash function [28] is a chop-MD construction, with a final transformation before chopping. The hash function employs a suffix-free padding rule. The underlying compression function f is based on two permutations $P, Q : \mathbb{Z}_2^l \to \mathbb{Z}_2^l$. The final transformation g is defined as $g(h) = P(h) \oplus h$. The Grøstl hash function design is given in Fig. 1.

The compression function of Grøstl is permutation based, and the results of [42, 43] apply. Furthermore, the preimage resistance of the compression function is analyzed in [27], and an upper bound for collision resistance can be obtained easily. As a consequence, we obtain tight security bounds on the compression function, $\mathbf{Adv}_f^{\mathrm{epre}} = \Theta(q^2/2^l)$ and $\mathbf{Adv}_f^{\mathrm{col}} = \Theta(q^4/2^l)$. 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). The Grøstl mode of operation preserves collision resistance and everywhere preimage resistance due to which we obtain $\mathbf{Adv}_{\mathcal{H}}^{\mathrm{col}} = \Theta(q^2/2^n)$ and $\mathbf{Adv}_{\mathcal{H}}^{\mathrm{epre}} = \Theta(q/2^n)$. Finally, it is proven indifferentiable from a random oracle up to bound $O((Kq)^4/2^l)$ if the underlying permutations are ideal [3].

As an addition to above results, in this work we consider second preimage resistance of the Grøstl hash function. We prove that optimal second preimage resistance (up to a constant) is achieved for all versions.

Theorem 1. *Let $n \in \mathbb{N}$, and $\lambda \geq 0$. The advantage of any adversary $\mathcal{A}$ in finding a second preimage for the* Grøstl *hash function $\mathcal{H}$ after $q < 2^{l-1}$ queries can be upper bounded by*

$$
\boldsymbol{Adv}_{\mathcal{H}}^{\mathrm{esec}[\lambda]}(q) \leq \frac{((\lambda+65)/m+2)q(q-1)}{2^l} + \frac{2q}{2^n}.
$$

Proof. Let $M' \in \mathbb{Z}_2^\lambda$ be any target preimage. Denote by $h'_0, \ldots, h'_{k'}$ the state values corresponding to the evaluation of $\mathcal{H}(M')$, and let $h = \mathsf{chop}_n(P(h'_{k'}) \oplus h'_{k'})$.

We consider any adversary $\mathcal{A}$ making q queries to its underlying permutations P and Q. Associated to these queries, we introduce an initially empty graph G that indicates compression function calls for Grøstl that can be derived from these queries. Note that any P-query (x_P, y_P) and any Q-query (x_Q, y_Q) correspond to exactly one compression function call, namely $x_P \oplus x_Q \to x_P \oplus y_P \oplus x_Q \oplus y_Q$ where the message input is x_Q. In order to find a second preimage, the adversary

(1) either needs to end up with a graph that contains a path (labeled differently from the first preimage) from IV to any node of $\{h'_0, \ldots, h'_{k'}\}$,
(2) or he needs to find a P-query (x_P, y_P) with $x_P \neq h'_{k'}$ such that $\mathsf{chop}_n(x_P \oplus y_P) = h$ and G contains a path from IV to x_P.

A proof of this claim can be found in [2, 18]. To achieve the first goal, the adversary needs to find a preimage for the Grøstl compression function, for any image in $\{h'_0, \ldots, h'_{k'}\}$. To achieve the second goal, the adversary needs to find a preimage for the final transformation of the Grøstl compression function. For $i = 1, \ldots, q$, we consider the probability of the i-th query to render success. We distinguish between the two success cases.

Case (1). Without loss of generality the i-th query is a forward query x_P to P, let y_P be the oracle answer drawn uniformly at random from a set of size at least $2^l - q$. Let (x_Q, y_Q) be any Q-query in the query history. The query results in a compression function call $x_P \oplus x_Q \rightarrow x_P \oplus y_P \oplus x_Q \oplus y_Q$. This value hits any of $\{h'_0, \ldots, h'_{k'}\}$ with probability at most $\frac{k'+1}{2^l-q}$. Considering any of the at most $i-1$ possible Q-queries, case (1) is achieved with probability at most $\frac{(k'+1)(i-1)}{2^l-q}$. The same bound is found for queries to Q and for inverse queries.

Case (2). Case (2) can only be achieved in a query to P. Without loss of generality, the i-th query is a forward query x_P, let y_P be the oracle answer drawn uniformly at random from a set of size at least $2^l - q$. This value satisfies $\mathsf{chop}_n(x_P \oplus y_P) = h$ with probability at most $\frac{2^{l-n}}{2^l-q}$.

By the union bound, we obtain the following bound on the second preimage resistance of Grøstl:

$$\mathbf{Adv}^{\mathrm{esec}[\lambda]}_{\mathcal{H}}(q) \leq \sum_{i=1}^{q} \frac{(k'+1)(i-1)}{2^l - q} + \frac{2^{l-n}}{2^l - q} \leq \frac{(k'+1)q(q-1)}{2(2^l - q)} + \frac{q2^{l-n}}{2^l - q}.$$

As for $q < 2^{l-1}$ we have $\frac{1}{2^l-q} \leq \frac{2}{2^l}$ and $k' \leq (\lambda + 65)/m + 1$, we obtain our result. □

Given that for Grøstl we have $l = 2n$, for $q < 2^n$ the result of Thm. 1 directly implies a $\Theta(\lambda/m \cdot q/2^n)$ bound on the second preimage resistance.

5 JH

The **JH** hash function [49] is a sponge-like function, but can also be considered as a parazoa function [1] or a chop-MD construction. The hash function employs a suffix-free padding rule. The compression function f is based on a permutation $\mathbb{Z}_2^l \rightarrow \mathbb{Z}_2^l$. The JH hash function design is given in Fig. 1. Note that the parameters of JH satisfy $l = 2m$.

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 [16]. The JH hash function is proven optimally collision resistant [33], and we obtain $\mathbf{Adv}^{\mathrm{col}}_{\mathcal{H}} = \Theta(q^2/2^n)$. Furthermore, it is proven indifferentiable from a random oracle up to bound $O\left(\frac{q^3}{2^{l-m}} + \frac{Kq^3}{2^{l-n}}\right)$ if the underlying permutation is assumed to be ideal [14]. As explained in [4, 5], using (1) this indifferentiability bound additionally renders an improved upper bound $O\left(\frac{q}{2^n} + \frac{q^3}{2^{l-m}}\right)$ on the preimage and second preimage resistance.

We note, however, that these bounds on the preimage and second preimage resistance of JH are non-optimal for both variants. We improve these bounds in Thms. 2 and 3. Although the new bounds are still not better than the trivial bound for $n = 512$ (as was the previous bound), they are now optimal (up to a constant) for the 256 variant.

In the proofs of Thms. 2 and 3 we will use the chop-function for both the left and right side of x. Therefore, we introduce the functions $\mathsf{left}_n(x)$ and $\mathsf{right}_n(x)$ that take the n leftmost and rightmost bits of x, respectively.

Theorem 2. *Let $n \in \mathbb{N}$. The advantage of any adversary $\mathcal{A}$ in finding a preimage for the* JH *hash function $\mathcal{H}$ after $q < 2^{l-1}$ queries can be upper bounded by*

$$\boldsymbol{Adv}^{\mathrm{epre}}_{\mathcal{H}}(q) \leq \frac{6q^2}{2^{l-m}} + \frac{2q}{2^n}.$$

Proof. Let $h \in \mathbb{Z}_2^n$ be any point to be inverted (cf. Def. 1). IV denotes the initialization vector of size l bits. We consider any adversary $\mathcal{A}$ making q queries to its underlying permutation P. Associated to these queries, we introduce an initially empty graph G that indicates compression function calls for JH that can be derived from these queries. We denote G_i as the graph after the i-th query ($i = 0, \ldots, q$). Each query adds 2^m edges to the graph, and G_i thus contains $i2^m$ edges. In order to find a preimage, the adversary must necessarily end up with a graph that contains a path from node IV to any node in $H := \{h \| h' \mid h' \in \mathbb{Z}_2^{l-n}\}$. We denote by winA_i the event that the i-th query makes this property satisfied.

We denote by G_i^{out}, resp. G_i^{in}, the set of nodes in G_i with an outgoing, resp. incoming, edge. We denote by τ_i^{IV} the subgraph of G_i consisting of all nodes and edges reachable from IV. Similarly, τ_i^H denotes the subgraph of G_i consisting of all nodes and edges from which any node in H can be reached. Next to event winA_i, we say the adversary also wins if either of the following events occurs for any $i = 1, \ldots, q$:

$$\begin{aligned}
\mathsf{winB}_i &: \quad \tau_i^{\mathsf{IV}} \text{ contains two nodes } v, v' \text{ with } \mathsf{left}_{l-m}(v) = \mathsf{left}_{l-m}(v'),\\
\mathsf{winC}_i &: \quad \tau_i^H \backslash H \text{ contains two nodes } v, v' \text{ with } \mathsf{right}_{l-m}(v) = \mathsf{right}_{l-m}(v').
\end{aligned}$$

We denote by $\mathsf{win}_i = \mathsf{winA}_i \cup \mathsf{winB}_i \cup \mathsf{winC}_i$ the event that the i-th query results in a winning adversary. We have

$$\mathbf{Adv}^{\mathrm{epre}}_{\mathcal{H}}(q) \leq \mathbf{Pr}\left(\mathsf{winA}_q\right) \leq \mathbf{Pr}\left(\mathsf{win}_q\right) \leq \sum_{i=1}^{q} \mathbf{Pr}\left(\mathsf{win}_i \mid \neg\mathsf{win}_{i-1}\right). \tag{2}$$

For $i = 1, \ldots, q$, we consider the probability of the i-th query to render success. We distinguish between forward and inverse queries.

Forward query. Suppose the adversary makes a forward query x_i to receive a random y_i. By $\neg\mathsf{winB}_{i-1}$, there is at most one $v \in \tau_{i-1}^{\mathsf{IV}}$ such that $\mathsf{left}_{l-m}(v) = \mathsf{left}_{l-m}(x_i)$. Denote $M = \mathsf{right}_{l-m}(v) \oplus \mathsf{right}_{l-m}(x_i)$; this query will add only the edge $v \to y_i \oplus (M \| 0^{l-m}) =: w$ to the tree. We define the following events.

$$\begin{aligned}
\mathsf{badA}_i &: \quad \mathsf{right}_{l-m}(y_i) \in \{\mathsf{right}_{l-m}(w) \mid w \in \tau_{i-1}^H\},\\
\mathsf{badB}_i &: \quad \mathsf{left}_{l-m}(w) \in \{\mathsf{left}_{l-m}(v) \mid v \in \tau_{i-1}^{\mathsf{IV}}\},\\
\mathsf{badC}_i &: \quad w \in G_i^{\mathrm{out}},\\
\mathsf{badD}_i &: \quad \mathsf{left}_n(w) = h.
\end{aligned}$$

Here, badA_i covers the event that τ_{i-1}^H is extended. Event badB_i covers the case that the updated τ_i^{IV} contains two nodes with the same left half (note that this would directly make winB_i satisfied). The case badC_i covers the event that the newly added edge to τ_i^{IV} hits any node with outgoing edge, and badD_i covers the event that the newly added edge to the tree would hit h (in both cases a valid preimage path may have been established). Denote $\mathsf{bad}_i = \mathsf{badA}_i \cup \mathsf{badB}_i \cup \mathsf{badC}_i \cup \mathsf{badD}_i$.

By basic probability theory, we have in case of forward queries

$$\mathbf{Pr}\left(\mathsf{win}_i \mid \neg\mathsf{win}_{i-1}\right) \leq \mathbf{Pr}\left(\mathsf{win}_i \mid \neg\mathsf{win}_{i-1} \wedge \neg\mathsf{bad}_i\right) + \mathbf{Pr}\left(\mathsf{bad}_i \mid \neg\mathsf{win}_{i-1}\right).$$

We consider the first probability. Assume $\neg\mathsf{win}_{i-1} \wedge \neg\mathsf{bad}_i$. Recall that by $\neg\mathsf{winB}_{i-1}$, $v \to w$ is the only edge added to τ_{i-1}^{IV}. Now, we have $\neg\mathsf{winA}_i$ by $\neg\mathsf{winA}_{i-1}$ and as by $\neg\mathsf{badC}_i \wedge \neg\mathsf{badD}_i$ this new edge does not connect τ_i^{IV} with H. Case $\neg\mathsf{winB}_i$ follows from $\neg\mathsf{winB}_{i-1} \wedge \neg\mathsf{badB}_i$. Finally, by $\neg\mathsf{badA}_i$, the tree τ_{i-1}^H is not extended, and hence $\neg\mathsf{winC}_i$ follows from $\neg\mathsf{winC}_{i-1}$. Thus, for forward queries we have $\mathbf{Pr}\left(\mathsf{win}_i \mid \neg\mathsf{win}_{i-1}\right) \leq \mathbf{Pr}\left(\mathsf{bad}_i \mid \neg\mathsf{win}_{i-1}\right)$. This probability will be analyzed later.

Inverse query. Suppose the adversary makes an inverse query y_i to receive a random x_i. By $\neg\mathsf{winC}_{i-1}$, there is at most one $v \in \tau_{i-1}^H$ such that $\mathsf{right}_{l-m}(v) = \mathsf{right}_{l-m}(y_i)$. Denote $M = \mathsf{left}_{l-m}(v) \oplus \mathsf{left}_{l-m}(y_i)$; this query will add only the edge $w := x_i \oplus (0^{l-m} \| M) \to v$ to the tree. We define the following events.

$$\begin{aligned}
\mathsf{badA}'_i &: \quad \mathsf{left}_{l-m}(x_i) \in \{\mathsf{left}_{l-m}(v) \mid v \in \tau_{i-1}^{\mathsf{IV}}\},\\
\mathsf{badB}'_i &: \quad \mathsf{right}_{l-m}(v) \in \{\mathsf{right}_{l-m}(w) \mid w \in \tau_{i-1}^H\},\\
\mathsf{badC}'_i &: \quad v \in G_i^{\mathrm{in}},\\
\mathsf{badD}'_i &: \quad v = \mathsf{IV}.
\end{aligned}$$

Here, badA'_i covers the event that τ_{i-1}^{IV} is extended. Event badB'_i covers the case that the updated τ_i^H contains two nodes with the same right half (note that this would directly make winC_i satisfied). The case badC'_i covers the event that the newly added edge to τ_i^H hits any node with incoming edge, and badD'_i covers the event that the newly added edge to the tree would hit IV (in both cases a valid preimage path may have been established). Denote $\mathsf{bad}'_i = \mathsf{badA}'_i \cup \mathsf{badB}'_i \cup \mathsf{badC}'_i \cup \mathsf{badD}'_i$.

By basic probability theory, we have in case of inverse queries

$$\mathbf{Pr}\left(\mathsf{win}_i \mid \neg\mathsf{win}_{i-1}\right) \leq \mathbf{Pr}\left(\mathsf{win}_i \mid \neg\mathsf{win}_{i-1} \wedge \neg\mathsf{bad}'_i\right) + \mathbf{Pr}\left(\mathsf{bad}'_i \mid \neg\mathsf{win}_{i-1}\right).$$

We consider the first probability. Assume $\neg\mathsf{win}_{i-1} \wedge \neg\mathsf{bad}'_i$. Recall that by $\neg\mathsf{winC}_{i-1}$, $v \to w$ is the only edge added to τ_{i-1}^H. Now, we have $\neg\mathsf{winA}_i$ by $\neg\mathsf{winA}_{i-1}$ and as by $\neg\mathsf{badC}'_i \wedge \neg\mathsf{badD}'_i$ this new edge does not connect IV with τ_i^H. By $\neg\mathsf{badA}'_i$, the tree τ_{i-1}^{IV} is not extended, and hence $\neg\mathsf{winB}_i$ follows from $\neg\mathsf{winB}_{i-1}$. Finally, case $\neg\mathsf{winC}_i$ follows from $\neg\mathsf{winC}_{i-1} \wedge \neg\mathsf{badB}'_i$. Thus, for inverse queries we have $\mathbf{Pr}\left(\mathsf{win}_i \mid \neg\mathsf{win}_{i-1}\right) \leq \mathbf{Pr}\left(\mathsf{bad}'_i \mid \neg\mathsf{win}_{i-1}\right)$. This probability will be analyzed later.

As each query is either a forward or an inverse query, we obtain for $i = 1, \ldots, q$:

$$\begin{aligned}
\mathbf{Pr}\left(\mathsf{win}_i \mid \neg\mathsf{win}_{i-1}\right) \leq\ & \mathbf{Pr}\left(\mathsf{bad}_i \mid \neg\mathsf{win}_{i-1};\ \text{forward query}\right) +\\
& \mathbf{Pr}\left(\mathsf{bad}'_i \mid \neg\mathsf{win}_{i-1};\ \text{inverse query}\right). \qquad (3)
\end{aligned}$$

As explained above, provided $\neg\mathsf{win}_{i-1}$, the i-th query adds at most one node to τ_{i-1}^{IV} and at most one node to τ_{i-1}^H, regardless whether it is a forward or inverse query. This particularly means that $|\tau_{i-1}^{\mathsf{IV}}|, |\tau_{i-1}^H| \leq i-1$. Additionally, $|G_i^{\mathrm{out}}|, |G_i^{\mathrm{in}}| \leq i2^m$. It is now straightforward to analyze the success probabilities of $\mathsf{bad}_i, \mathsf{bad}'_i$ to occur. As the answer from P is drawn uniformly at random from a set of size at least $2^l - q$, we obtain from (3):

$$\mathbf{Pr}\left(\mathsf{win}_i \mid \neg\mathsf{win}_{i-1}\right) \leq \frac{4(i-1)2^m}{2^l - q} + \frac{2i2^m}{2^l - q} + \frac{2^{l-n}}{2^l - q} + \frac{1}{2^l - q}. \qquad (4)$$

This combines with (2) to

$$\mathbf{Adv}_{\mathcal{H}}^{\mathrm{epre}}(q) \leq \sum_{i=1}^{q} \frac{(6i-4)2^m}{2^l - q} + \frac{2^{l-n}+1}{2^l - q} \leq \frac{3q^2 2^m}{2^l - q} + \frac{q2^{l-n}}{2^l - q},$$

The result is now obtained as for $q < 2^{l-1}$ we have $\frac{1}{2^l - q} \leq \frac{2}{2^l}$. □

The proof of second preimage resistance of JH is similar. Note that the attack by Kelsey and Schneier [30] only impacts JH in the internal state, which is reflected by the second part of the bound. In accordance with NIST's security requirements, we can assume $q < 2^{n-L}$, or in particular that $\lambda/m \cdot q \lesssim 2^n$ (see the remark below Def. 2). Consequently, the second term of the second preimage bound is negligible.

Theorem 3. *Let $n \in \mathbb{N}$, and $\lambda \geq 0$. The advantage of any adversary $\mathcal{A}$ in finding a second preimage for the* JH *hash function $\mathcal{H}$ after $q < 2^{l-1}$ queries can be upper bounded by*

$$\boldsymbol{Adv}^{\mathrm{esec}[\lambda]}_{\mathcal{H}}(q) \leq \frac{6q^2}{2^{l-m}} + \frac{4(\lambda/m+2)q}{2^l} + \frac{2q}{2^n}.$$

Proof. The proof follows the same argument as the proof of Thm. 2; we only highlight the differences. Let $M' \in \mathbb{Z}_2^\lambda$ be any target preimage. Denote by $h'_0, \ldots, h'_{k'}$ the state values corresponding to the evaluation of $\mathcal{H}(M')$, and set $\mathsf{left}_n(h'_{k'}) = h$. Now, the adversary necessarily needs to end up with a graph that contains a path from IV to any node in

$$\{h'_0, \ldots, h'_{k'}\} \cup \{h \| h' \mid h' \in \mathbb{Z}_2^{l-n}\}.$$

This path must be labeled by a message different from M'. The analysis of Thm. 2 carries over, with the minor difference that badC_i and badC'_i are replaced by

$$\begin{aligned} \mathsf{badC}_i &: \quad w \in G_i^{\mathrm{out}} \cup \{h'_0, \ldots, h'_{k'-1}\}, \\ \mathsf{badC}'_i &: \quad v \in G_i^{\mathrm{in}} \cup \{h'_1, \ldots, h'_{k'}\}. \end{aligned}$$

Similar as before, we obtain

$$\mathbf{Adv}^{\mathrm{esec}[\lambda]}_{\mathcal{H}}(q) \leq \sum_{i=1}^{q} \frac{(6i-4)2^m}{2^l - q} + \frac{2k'}{2^l - q} + \frac{2^{l-n}+1}{2^l - q} \leq \frac{3q^2 2^m}{2^l - q} + \frac{2k'q}{2^l - q} + \frac{q2^{l-n}}{2^l - q}.$$

The result is now obtained from the fact that $k' \leq \lambda/m + 2$. □

6 Keccak

The **Keccak** hash function [10] is a sponge function, but can also be considered as a parazoa function [1] or a chop-MD construction. The compression function f is based on a permutation $\mathbb{Z}_2^l \to \mathbb{Z}_2^l$. The hash function output is obtained by chopping off $l - n$ bits of the state[4]. Notice that the parameters of Keccak satisfy $l = 2n + m$. The Keccak hash function design is given in Fig. 1.

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 [16]. The Keccak hash function is proven indifferentiable from a random oracle up to bound $\Theta((Kq)^2/2^{l-m})$ if the underlying permutation is assumed to be ideal [11]. Using (1), this indifferentiability bound renders an optimal collision resistance bound for Keccak, $\mathbf{Adv}^{\mathrm{col}}_{\mathcal{H}} = \Theta(q^2/2^n)$, as well as optimal preimage second preimage resistance bounds $\Theta(q/2^n)$.

[4] We notice that sponge function designs are more general [12], but for Keccak this description suffices.

7 Skein

The **Skein** hash function [25] is a chop-MD construction. The message blocks are accompanied with a round-specific tweak[5], and the function employs a suffix- and prefix-free padding rule. The compression function f is based on a tweakable block cipher $E : \mathbb{Z}_2^m \times \mathbb{Z}_2^{128} \times \mathbb{Z}_2^l \to \mathbb{Z}_2^m$. The Skein hash function design is given in Fig. 1.

The compression function of Skein is the PGV1, or Matyas-Meyer-Oseas, compression function, with a difference that a tweak is involved. As claimed in [8], the results of [17] carry over, which in turn results in optimal security bounds on the compression function. 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). The Skein mode of operation preserves collision resistance and everywhere preimage resistance due to which we obtain $\mathbf{Adv}^{\mathrm{col}}_{\mathcal{H}} = \Theta(q^2/2^n)$ and $\mathbf{Adv}^{\mathrm{epre}}_{\mathcal{H}} = \Theta(q/2^n)$. Furthermore, the Skein hash function is proven indifferentiable from a random oracle up to bound $O((Kq)^2/2^l)$ if the underlying tweakable block cipher is assumed to be ideal [8]. This proof is based on the preimage awareness approach [23]. Using (1), this indifferentiability bound additionally renders an improved upper bound $O\left(\frac{q}{2^n} + \frac{q^2}{2^l}\right)$ on the second preimage resistance.

The second preimage bound for Skein is optimal for the $n = 256$ variant, but meets the trivial bound for the $n = 512$ variant. Therefore, we reconsider second preimage resistance of the Skein hash function. We prove that optimal second preimage resistance (up to a constant) is achieved for all versions.

Theorem 4. *Let $n \in \mathbb{N}$, and $\lambda \geq 0$. The advantage of any adversary $\mathcal{A}$ in finding a second preimage for the* Skein *hash function $\mathcal{H}$ after $q < 2^{l-1}$ queries can be upper bounded by*

$$\boldsymbol{Adv}^{\mathrm{esec}[\lambda]}_{\mathcal{H}}(q) \leq \frac{2q}{2^l} + \frac{2q}{2^n}.$$

Proof. The proof follows a similar reasoning as the proof of Thm. 1, and we only highlight the differences. Let $M' \in \mathbb{Z}_2^\lambda$ be any target preimage. Denote by $h'_0, \ldots, h'_{k'}$ the state values corresponding to the evaluation of $\mathcal{H}(M')$, and let $h = \mathsf{chop}_n(h'_{k'})$.

We consider any adversary $\mathcal{A}$ making q queries to its underlying block cipher E. Associated to these queries, we introduce an initially empty graph G that indicates compression function calls for Skein that can be derived from these queries. Note that any query tuple $(M, T, h) \to C$ corresponds to exactly one compression function call, namely $h \to C \oplus M$ where the message input is M and where T is a tweak value. These tweaks are round-specific (see Fig. 1). In order to find a second preimage, the adversary needs to end up with a graph that contains a path (labeled different from the first preimage) from IV to any node of $\{h'_0, \ldots, h'_{k'}\} \cup \{h \| h' \mid h' \in \mathbb{Z}_2^{l-n}\}$, where the associated tweaks need to be compliant with the hash function evaluation corresponding to the path. A proof of this claim can be found in [2, 18]. To achieve the first goal, the adversary needs to find a preimage for the Skein compression function, for any image in $H_1 := \{h'_0, \ldots, h'_{k'}\}$ or $H_2 := \{h \| h' \mid h' \in \mathbb{Z}_2^{l-n}\}$ (where the tweak is compliant). For $i = 1, \ldots, q$, we consider the probability of the i-th query to render success. We distinguish between the two sets H_1, H_2. Without loss of generality, let the i-th query be a forward query

[5] 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.

M, T, h, and let C be the oracle answer drawn uniformly at random from a set of size at least $2^l - q$. The same bounds are found for inverse queries.

Set H_1. As the tweaks need to be compliant, depending on T there is at most one value $h \in H_1$ for which a collision $h = C \oplus M$ may result in a valid second preimage. The i-th query thus renders a collision with H_1 probability at most $\frac{1}{2^l - q}$.

Set H_2. A collision with any element from H_2 may result in a valid preimage. $C \oplus M$ collides with any element from H_2 with probability at most $\frac{2^{l-n}}{2^l - q}$.

By the union bound, we obtain the following bound on the second preimage resistance of Skein:

$$\mathbf{Adv}_{\mathcal{H}}^{\mathrm{esec}[\lambda]}(q) \leq \sum_{i=1}^{q} \frac{1}{2^l - q} + \frac{2^{l-n}}{2^l - q} \leq \frac{q}{2^l - q} + \frac{q2^{l-n}}{2^l - q}.$$

The result is now obtained as for $q < 2^{l-1}$ we have $\frac{1}{2^l - q} \leq \frac{2}{2^l}$. □

8 Conclusions

In this work we revisited the previous summary of [4, 5] with respect to the five finalist SHA-3 hash functions. More concretely, we updated existing results with the new results in the area in Table 2, part of which are freshly proved in this paper. A main improvement of this work is that all results in our analysis hold for ideal primitives of comparable size; either ideal ciphers or permutations. Secondly, most "security gaps" (with respect to preimage, second preimage, and collision resistance) remaining from [4, 5] are closed. One of the few open problems left for the security analysis of the five finalist hash functions in the ideal model is achieving an optimal (second) preimage bound of the 512 variant of the JH hash function.

We note that our security analysis needs to be read with care and for this purpose we provide the following discussion:

- Ideal primitives do not exist and the ideal model proofs are only an indication for security. In particular, none of the candidates' underlying block cipher or permutation is ideal. However, due to the lack of security proofs in the standard model (other than preserving collision security of the compression function in MD based designs), assuming ideality of these underlying primitives gives significantly more confidence in the security of the higher level hash function structure than any ad-hoc analysis or no proof at all;
- While assuming ideality of sizable underlying building blocks like permutations and block ciphers allows for a fair security comparison of the candidates on one hand, it disregards internal differences between the idealized primitives on the other. Such specific design details can distort the security results for the distinct hash functions when concrete attacks exploiting the internal primitive weaknesses are applied. Moreover, further differences, such as chaining sizes and message input sizes, are also not fully reflected in the security results.

References

[1] Andreeva, E., Mennink, B., Preneel, B.: The parazoa family: Generalizing the sponge hash functions. Cryptology ePrint Archive, Report 2011/028 (2011)

[2] Andreeva, E., Luykx, A., Mennink, B.: Provable security of BLAKE with non-ideal compression function. Cryptology ePrint Archive, Report 2011/620 (2011)

[3] Andreeva, E., Mennink, B., Preneel, B.: On the indifferentiability of the Grøstl hash function. In: SCN 2010. LNCS, vol. 6280, pp. 88–105. Springer-Verlag, Berlin (2010)

[4] Andreeva, E., Mennink, B., Preneel, B.: Security reductions of the second round SHA-3 candidates. In: ISC 2010. LNCS, vol. 6531, pp. 39–53. Springer-Verlag, Berlin (2010)

Table 2. A schematic summary of all security results of the SHA-3 finalists. The second column summarizes the parameters n, l, m, which denote the hash function output size, the chaining value size and the message input size, respectively. The last row of the table gives a representation of the security requirements (ii)-(iv) by NIST.

	$n/l/m$	$\mathbf{Adv}_f^{\mathrm{epre}}$	$\mathbf{Adv}_f^{\mathrm{esec}[\lambda]}$	$\mathbf{Adv}_f^{\mathrm{col}}$	$\mathbf{Adv}_{\mathcal{H}}^{\mathrm{epre}}$	$\mathbf{Adv}_{\mathcal{H}}^{\mathrm{esec}[\lambda]}$	$\mathbf{Adv}_{\mathcal{H}}^{\mathrm{col}}$	$\mathbf{Adv}_{\mathcal{H}}^{\mathrm{pro}}$
BLAKE	256/256/512, 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	$\Theta(q^2/2^n)$ E ideal	$\Theta((Kq)^2/2^n)$ E ideal
Grøstl	256/512/512, 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	$\Theta(\lambda/m \cdot q/2^n)$ P, Q ideal	$\Theta(q^2/2^n)$ P, Q ideal	$O((Kq)^4/2^l)$ P, Q ideal
JH	256/1024/512, 512/1024/512	$\Theta(1)$ P ideal	$\Theta(1)$ P ideal	$\Theta(1)$ P ideal	$O(q/2^n + q^2/2^{l-m})$ P ideal	$O(q/2^n + q^2/2^{l-m})$ P ideal	$\Theta(q^2/2^n)$ P ideal	$O(Kq^3/2^{l-n} + q^3/2^{l-m})$ P ideal
Keccak	256/1600/1088, 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	$\Theta(q^2/2^n)$ P ideal	$\Theta((Kq)^2/2^{l-m})$ P ideal
Skein	256/512/512, 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	$\Theta(q^2/2^n)$ E ideal	$O((Kq)^2/2^l)$ E ideal
NIST's security requirements [38]		(not specified)	(not specified)	(not specified)	$O(q/2^n)$	$O(\lambda/m \cdot q/2^n)$	$O(q^2/2^n)$	(not specified)

[5] Andreeva, E., Mennink, B., Preneel, B.: Security reductions of the SHA-3 candidates (2010), NIST's 2nd SHA-3 Candidate Conference 2010. Workshop record

[6] Andreeva, E., Neven, G., Preneel, B., Shrimpton, T.: Seven-property-preserving iterated hashing: ROX. In: ASIACRYPT 2007. LNCS, vol. 4833, pp. 130–146. Springer-Verlag, Berlin (2007)

[7] Aumasson, J., Henzen, L., Meier, W., Phan, R.: SHA-3 proposal BLAKE (2010), submission to NIST's SHA-3 competition

[8] 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)

[9] Bellare, M., Rogaway, P.: Random oracles are practical: A paradigm for designing efficient protocols. In: ACM Conference on Computer and Communications Security. pp. 62–73. ACM, New York (1993)

[10] Bertoni, G., Daemen, J., Peeters, M., Assche, G.: The KECCAK sponge function family (2011), submission to NIST's SHA-3 competition

[11] Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: On the indifferentiability of the sponge construction. In: EUROCRYPT 2008. LNCS, vol. 4965, pp. 181–197. Springer-Verlag, Berlin (2008)

[12] Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: Sponge functions (ECRYPT Hash Workshop 2007)

[13] Bhattacharyya, R., Mandal, A., Nandi, M.: Indifferentiability characterization of hash functions and optimal bounds of popular domain extensions. In: INDOCRYPT 2009. LNCS, vol. 5922, pp. 199–218. Springer-Verlag, Berlin (2009)

[14] Bhattacharyya, R., Mandal, A., Nandi, M.: Security analysis of the mode of JH hash function. In: FSE 2010. LNCS, vol. 6147, pp. 168–191. Springer-Verlag, Berlin (2010)

[15] Biham, E., Dunkelman, O.: A framework for iterative hash functions – HAIFA. Cryptology ePrint Archive, Report 2007/278 (2007)

[16] Black, J., Cochran, M., Shrimpton, T.: On the impossibility of highly-efficient blockcipher-based hash functions. In: EUROCRYPT 2005. LNCS, vol. 3494, pp. 526–541. Springer-Verlag, Berlin (2005)

[17] Black, J., Rogaway, P., Shrimpton, T.: Black-box analysis of the block-cipher-based hash-function constructions from PGV. In: CRYPTO 2002. LNCS, vol. 2442, pp. 320–335. Springer-Verlag, Berlin (2002)

[18] Bouillaguet, C., Fouque, P.: Practical hash functions constructions resistant to generic second preimage attacks beyond the birthday bound (2010), submitted to Information Processing Letters

[19] Chang, D., Lee, S., Nandi, M., Yung, M.: Indifferentiable security analysis of popular hash functions with prefix-free padding. In: ASIACRYPT 2006. LNCS, vol. 4284, pp. 283–298. Springer-Verlag, Berlin (2006)

[20] Chang, D., Nandi, M., Yung, M.: Indifferentiability of the hash algorithm BLAKE. Cryptology ePrint Archive, Report 2011/623 (2011)

[21] Coron, J.S., Dodis, Y., Malinaud, C., Puniya, P.: Merkle-Damgård revisited: How to construct a hash function. In: CRYPTO 2005. LNCS, vol. 3621, pp. 430–448. Springer-Verlag, Berlin (2005)

[22] Damgård, I.: A design principle for hash functions. In: CRYPTO '89. LNCS, vol. 435, pp. 416–427. Springer-Verlag, Berlin (1990)

[23] Dodis, Y., Ristenpart, T., Shrimpton, T.: Salvaging Merkle-Damgård for practical applications. In: EUROCRYPT 2009. LNCS, vol. 5479, pp. 371–388. Springer-Verlag, Berlin (2009)

[24] Ferguson, N., Lucks, S., Schneier, B., Whiting, D., Bellare, M., Kohno, T., Callas, J., Walker, J.: Engineering comparison of SHA-3 candidates (2010)
[25] Ferguson, N., Lucks, S., Schneier, B., Whiting, D., Bellare, M., Kohno, T., Callas, J., Walker, J.: The Skein Hash Function Family (2010), submission to NIST's SHA-3 competition
[26] Fleischmann, E., Forler, C., Gorski, M.: Classification of the SHA-3 candidates. Cryptology ePrint Archive, Report 2008/511 (2008)
[27] Fouque, P.A., Stern, J., Zimmer, S.: Cryptanalysis of tweaked versions of SMASH and reparation. In: SAC 2008. LNCS, vol. 5381, pp. 136–150. Springer-Verlag, Berlin (2009)
[28] Gauravaram, P., Knudsen, L., Matusiewicz, K., Mendel, F., Rechberger, C., Schläffer, M., Thomsen, S.: Grøstl – a SHA-3 candidate (2011), submission to NIST's SHA-3 competition
[29] 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)
[30] Kelsey, J., Schneier, B.: Second preimages on n-bit hash functions for much less than 2^n work. In: EUROCRYPT 2005. LNCS, vol. 3494, pp. 474–490. Springer-Verlag, Berlin (2005)
[31] 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)
[32] Lai, X., Massey, J.: Hash function based on block ciphers. In: EUROCRYPT '92. LNCS, vol. 658, pp. 55–70. Springer-Verlag, Berlin (1992)
[33] Lee, J., Hong, D.: Collision resistance of the JH hash function. Cryptology ePrint Archive, Report 2011/019 (2011)
[34] Lucks, S.: A failure-friendly design principle for hash functions. In: ASIACRYPT 2005. LNCS, vol. 3788, pp. 474–494. Springer-Verlag, Berlin (2005)
[35] Luo, Y., Gong, Z., Duan, M., Zhu, B., Lai, X.: Revisiting the indifferentiability of PGV hash functions. Cryptology ePrint Archive, Report 2009/265 (2009)
[36] Maurer, U., Renner, R., Holenstein, C.: Indifferentiability, impossibility results on reductions, and applications to the random oracle methodology. In: TCC 2004. LNCS, vol. 2951, pp. 21–39. Springer-Verlag, Berlin (2004)
[37] Merkle, R.: One way hash functions and DES. In: CRYPTO '89. LNCS, vol. 435, pp. 428–446. Springer-Verlag, Berlin (1990)
[38] National Institute for Standards and Technology. Announcing Request for Candidate Algorithm Nominations for a New Cryptographic Hash Algorithm (SHA3) Family (November 2007)
[39] 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)
[40] Ristenpart, T., Shacham, H., Shrimpton, T.: Careful with composition: Limitations of the indifferentiability framework. In: EUROCRYPT 2011. LNCS, vol. 6632, pp. 487–506. Springer-Verlag, Berlin (2011)
[41] Rogaway, P., Shrimpton, T.: Cryptographic hash-function basics: Definitions, implications, and separations for preimage resistance, second-preimage resistance, and collision resistance. In: FSE 2004. LNCS, vol. 3017, pp. 371–388. Springer-Verlag, Berlin (2004)
[42] Rogaway, P., Steinberger, J.: Security/efficiency tradeoffs for permutation-based hashing. In: EUROCRYPT 2008. LNCS, vol. 4965, pp. 220–236. Springer-Verlag, Berlin (2008)
[43] Stam, M.: Beyond uniformity: Better security/efficiency tradeoffs for compression functions. In: CRYPTO 2008. LNCS, vol. 5157, pp. 397–412. Springer-Verlag, Berlin (2008)
[44] Stam, M.: Blockcipher-based hashing revisited. In: FSE 2009. LNCS, vol. 5665, pp. 67–83. Springer-Verlag, Berlin (2009)
[45] Steinberger, J.: Stam's collision resistance conjecture. In: EUROCRYPT 2010. LNCS, vol. 6110, pp. 597–615. Springer-Verlag, Berlin (2010)
[46] Tillich, S., Feldhofer, M., Kirschbaum, M., Plos, T., Schmidt, J.M., 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)
[47] Wang, X., Yin, Y.L., Yu, H.: Finding collisions in the full SHA-1. In: CRYPTO 2005. LNCS, vol. 3621, pp. 17–36. Springer-Verlag, Berlin (2005)
[48] Wang, X., Yu, H.: How to break MD5 and other hash functions. In: EUROCRYPT 2005. LNCS, vol. 3494, pp. 19–35. Springer-Verlag, Berlin (2005)
[49] Wu, H.: The Hash Function JH (2011), submission to NIST's SHA-3 competition