Classification of the SHA-3 Candidates

Ewan Fleischmann[1], Christian Forler[1,2], and Michael Gorski[1]

[1] Bauhaus-University Weimar {Ewan.Fleischmann, Michael.Gorski}@uni-weimar.de
[2] Sirrix AG security technologies c.forler@sirrix.com

Version 0.1
December 3, 2008

Abstract. In this note we give an overview on the current state of the SHA-3 candidates. First, we classify all publicly known candidates and, second, we outline and summarize the performance data as given in the candidates documentation for 64-bit and 32-bit implementations. We define performance classes and classify the hash algorithms. Note, that this article will be updated as soon as new candidates arrive or new cryptanalytic results get published. Comments to the authors of this article are welcome.

Keywords: hash function, SHA-3, classification.

1 Introduction

The design of secure and practical hash functions is of great interest since most practical hash functions, like MD5 [48], SHA-0 [45] or SHA-1 [43] have been broken. Due to the SHA-3 competition [42], many new proposals for hash function primitives have been submitted to become the new SHA-3 algorithm.

This article is organized as follows: In Section 2 we define criteria that we will use to classify the SHA-3 candidate algorithms. In Section 3 we give an overview of the software performance claimed by the algorithm's authors.

2 Classification of the SHA-3 Candidates

We have defined in the following some attributes that are used in our classification.

Balanced Feistel Network (BFN) [54]

A compression function is called a balanced feistel network, when

1. the internal state is divided into a left and right part of equal size n.
2. a message depended, nonlinear function F maps those parts to two output parts of the same length.

Feistel networks usually consists of a series of rounds.

Unbalanced feistel network (UFN) [54]

A compression function is called an unbalanced feistel network is based on a feistel network where the internal state is divided into more resp. less then two parts or into two parts of unequal size.

Wide Pipe design (WP) [31]
The internal state, i.e. chaining value, of the hash function is larger than the message digest.

Key Schedule (KEY)
The hash function has an explicit key schedule or a message expansion algorithm.

MDS Matrix (MDS) [51]
One or more Maximum Distance Separable (MDS) matrices are used as a building block of the compression function. A MDS matrix has strong diffusion properties that can be exploited in certain cryptographic primitives..

Output Transformation (OUT)
Is a function with the“final” chaining value as input and the message digest as output. The identity does not count at all.

S-box (SBOX)
The hash function uses one or more substitution boxes. In general a S-box is a non linear function that maps m input bits to n output bits. Usually, a S-box is implemented as lookup table.

Feedback Register (FSR)
The compression functions is/uses a (N)LFSRs. The input bits of a (non-)linear feedback shift register ((N)LFSR) are computed via a (non-)linear function from the previous state.

Collision Attack
The best known collision attack that is better than the birthday attack.

(Second) Preimage Attack
The best known (2nd) preimage attack that is better than then long second preimage attack [24].

Hash algorithm	**BFN**	**UFN**	**WP**	**KEY**	**MDS**	**OUT**	**SBOX**	**FSR**	**COL**	**PRE**
BLAKE [4]	-	X	-	X	-	-	-	-	-	-
BMW [15]	-	-	X	X	-	-	-	-	[55]†	-
Boole [50]	-	-	-	-	-	X	-	X	-	$2^{\frac{9n}{16}}$ [41]
Chi [20]	-	X	X	X	-	-	X	-	-	-
CRUNCH [17]	-	X	-	X	-	-	X	-	-	-
CubeHash8/1 [6]	-	-	-	-	-	-	-	-	-	-
DHC [60]	-	-	-	X	-	-	X	-	$2^{45}/2^{45}$ [26]	$2^{45}/2^{45}$ [26]
Edon-R [16]	-	-	X	X	-	-	-	-	-	$2^{\frac{2n}{3}}/2^{\frac{2n}{3}}$ [27]
EnRUPT [46]	-	-	(X)	-	-	-	-	-	2^{40} [21]	$2^{480}/2^{480}$ [25]
Essence [34]	-	-	-	-	-	-	-	X	-	-
FSB [3]	-	-	X	-	-	X	-	-	-	-
Fugue [18]	-	-	X	-	X	X	X	-	-	-

† Near Collision.

Table 1. Attribute list of known SHA-3 candidates(A-F).

Hash algorithm	**BFN**	**UFN**	**WP**	**KEY**	**MDS**	**OUT**	**SBOX**	**FSR**	**COL**	**PRE**
Fugue [18]	-	-	X	-	X	X	X	-	-	-
Grøstl [14]	-	-	X	-	X	X	X	-	-	-
JH [61]	X	-	X	-	X	-	X	-	-	-
Keccak [8]	-	-	X	-	-	X	-	-	-	-
LANE [22]	-	-	-	X	X	X	X	-	-	-
Maraca [23]	-	-	X	X	-	-	-	-	-	-
MCSSHA-3 [35]	-	-	-	-	-	-	-	X	$2^{3n/8}$ [5]	$2^{3n/4}$ [5]
MD6 [49]	-	-	X	-	-	-	-	X	-	-
MeshHash [12]	-	-	-	-	-	X	X	-	-	$2^{323.2}/2^{n/2}$ [56]
NaSHA [33]	X	-	-	-	-	-	X	X	-	-
NKS2D [47]	-	-	-	-	-	-	-	-	[9, 10]	-
Ponic [53]	-	-	X	-	-	X	X	X	-	2^{265} [39]

† Near Collision.

Table 2. Attribute list of known SHA-3 candidates(G-P).

Hash algorithm	**BFN**	**UFN**	**WP**	**KEY**	**MDS**	**OUT**	**SBOX**	**FSR**	**COL**	**PRE**
Sarmal [57]	-	X	-	-	X	-	X	-	-	$2^{384}/2^{128}$ [40]
Sgàil [37]	-	-	X	X	X	-	X	-	[36]	[36]
SHAMATA [2]	X	-	X	X	X	-	X	-	-	-
SIMD [30]	-	X	X	X	-	-	-	-	-	-
Skein [13]	X	-	-	X	-	X	-	-	-	-
Spectral Hash [52]	-	-	-	-	-	X	X	-	[11]	-
SWIFFTX [1]	-	-	-	-	-	-	X	-	-	-
TIB3 [38]	-	X	-	X	-	-	X	-	-	-
Vortex [29]	-	-	-	-	X	X	X	-	$2^{n/4}$ [28]†	$2^{3n/4}$ [28]
WAMM [58]	-	-	X	-	-	X	X	-	[59]	[59]
Waterfall [19]	-	-	X	-	-	X	X	X	-	-

† Near Collision.

Table 3. Attribute list of known SHA-3 candidates (Q-Z).

3 Software Speed of the SHA-3 Candidates

In this section we give an overview of the claimed software performance of the public known SHA-3 candidates. We compare each candidate for their 32 and 64 bit performance. Therefore, we define five speed classes, which are listed in Table 4.
Tables 5-7 compare the SHA-3 candidates and their speed classes. As a reference algorithm we add SHA-256/ 512 [44]. Since each SHA-2 version is in class C for the 32 bit performance

Speed (cpb)	Classification
1-12	A
13-25	B
26-55	C
56-80	D
81+	E

Table 4. Speed classification table

and in class B for the 64 bit performance, we think that this can be seen as a benchmark for all algorithms submitted. Nevertheless, there is a tradeoff between speed and security. One can easily design a hash function with a high level of security which is very slow and therefore may be useless in practice. For practical interest algorithms that are in speed class D or E will have a disadvantage for practical purpose, but they could possibly face a strong design. On the other side if an algorithm is very fast, i.e. in speed class A, this could be a hint that the security margin is not chosen so high. Recent breaks of very fast hash functions, i.e. EnRUPT [46] or Boole [50], have verified this conjecture.

Hash algorithm	**Performance 32 Bit**		**Performance 64 Bit**	
	cpb	class	cpb	class
SHA-256 [44]	29.3	C	20.1	B
SHA-512 [44]	55.2	C	13.1	B
BLAKE-32[4]	28.3	C	16.7	B
BLAKE-64[4]	61.7	D	12.3	A
BMW-256 [15]	8.6	A	7.85	A
BMW-512 [15]	13.37	B	4.06	A
Boole [50]	8.9	A	6.1	A
Chi-256 [20]	49	C	26	C
Chi-512 [20]	78	D	16	B
CRUNCH-256 [17]	29.9	C	16.9	B
CRUNCH-512 [17]	86.4	E	46.9	C
CubeHash8/1 [7]	200	E	148	E
DHC [60]	230	E	160	E
Edon-R-256 [16]	9.1	A	5.9	A
Edon-R-512 [16]	13.7	B	2.9	A
EnRUPT-256 [46]	8.3	A	8.3	A
EnRUPT-512 [46]	5.1	A	5.1	A
Essence-256 [34]	149.8	E	19.5	B
Essence-512 [34]	176.5	E	23.5	B
FSB-256 [3]	324	E	-	-
FSB-512 [3]	507	E	-	-
Fugue-256 [18]	36,2‡	C	61‡	D
Fugue-512 [18]	74.6‡	D	132.7‡	E

‡ Test platform is Intel Family 6 Model 15 XEON 5150 for 32-bit and Intel Family 15 Model 4 Xeon for 64-bit performance tests. The cpb values are approximated from documented MB/sec.

Table 5. Claimed software speed list of SHA-3 candidates (A-F). Benchmarks are in cycles per byte (cpb) on NIST target platform (Intel Core 2 Duo).

Hash algorithm	Performance 32 Bit		Performance 64 Bit	
	cpb	class	cpb	class
SHA-256 [44]	29.3	C	20.1	B
SHA-512 [44]	55.2	C	13.1	B
Grøstl-256 [14]	77.9	D	25.4	B
Grøstl-512 [14]	123.4	E	36.9	C
JH [61]	21.3	B	16.8	B
Keccak-256 [8]	80.3	D	34.4	C
Keccak-512 [8]	159.6	E	17.1	B
LANE-256 [22]	40.4	C	25.6	B
LANE-512 [22]	152.2	E	145.3	E
Maraca [23]	5.5	A	$5.3^{\diamond}$	A
MCSSHA-3 [35]	?	?	?	?
MeshHash-256 [12]	14.7	B	4.4	A
MeshHash-512 [12]	39.1	C	10.3	A
MD6-256 [49]	68	D	28	C
MD6-512 [49]	106	E	44	C
NaSHA-256 [32]	39	C	28.4	C
NaSHA-512 [32]	38.9	C	29.3	C
NKS2D-256 [47]	178^{+}	E	117^{+}	E
NKS2D-256 [47]	350^{+}	E	243^{+}	E
Ponic [53]	$7.2^{\cap}$	A	$3.2^{\cap}$	A

$^{\diamond}$ Test platform: Intel Dual E5320 Quad Core.

$^{+}$ Test platform: AMD Phenom 9500 Quad Core.

$^{\cap}$ Test platform: AMD Athlon.

Table 6. Claimed software speed list of SHA-3 candidates (G-P). Benchmarks are in cycles per byte (cpb) on NIST target platform (Intel Core 2 Duo).

Hash algorithm	Performance 32 Bit		Performance 64 Bit	
	cpb	class	cpb	class
SHA-256 [44]	29.3	C	20.1	B
SHA-512 [44]	55.2	C	13.1	B
Sarmal-256 [57]	19.2	B	10	A
Sarmal-512 [57]	23.3	B	12.6	A
Sgàil [37]	-	-	61	D
SHAMATA-224/256 [2]	15	B	8	A
SHAMATA-384/512 [2]	22	B	11	A
SIMD-256 [30]	12	A	11	A
SIMD-512 [30]	118	E	85	E
Skein-256 [13]	32.8	C	7.6	A
Skein-512 [13]	32.5	C	6.1	A
Spectral Hash [52]	454.68 †	E	454.68 †	E
TIB3-256 [38]	12.9	A	7.6	A
TIB3-512 [38]	17.5	B	6.3	A
WAMM [58]	268†	E	268 †	E
SWIFFTX [1]	57	D	?	-
Vortex-256 [29]	46.26	C	69.44	D
Vortex-512 [29]	56.05	D	90.07	E
WAMM [58]	268 †	E	268 †	E
Waterfall [19]	16.33	B	$^{\#}$	–

$^{\#}$ not specified in the document.

† Not specified whether on 32-bit or 64-bit tested, cpb value approximated from documented MB/sec.

Table 7. Claimed software speed list of SHA-3 candidates (Q-Z). Benchmarks are in cycles per byte (cpb) on NIST target platform (Intel Core 2 Duo).

References

[1] Yuriy Arbitman, Gil Dogon, Vadim Lyubashevsky, Daniele Micciancio, Chris Peikert, and Alon Rosen. SWIFFTX: A Proposal for the SHA-3 Standard. Submission to NIST, 2008.

[2] Adem Atalay, Orhun Kara, Ferhat Karakoc, and Cevat Manap. SHAMATA HASH FUNCTION ALGORITHM SPECIFICATIONS. Submission to NIST, 2008.

[3] Daniel Augot, Matthieu Finiasz, Philippe Gaborit, Stphane Manuel, and Nicolas Sendrier. SHA-3 proposal: FSB. Submission to NIST, 2008.

[4] Jean-Philippe Aumasson, Luca Henzen, Willi Meier, and Raphael C.-W. Phan. SHA-3 proposal BLAKE. Submission to NIST, 2008.

[5] Jean-Philippe Aumasson and Mara Naya-Plasencia. Second preimages on MCSSHA-3. Available online, 2008.

[6] Daniel J. Bernstein. CubeHash Specification (2.B.1). Submission to NIST, 2008.

[7] Daniel J. Bernstein. CubeHash8/1 Performance. Submission to NIST, 2008.

[8] G. Bertoni, J. Daemen, M. Peeters, and G. Van Assche.

[9] Christophe De Cannire. Collisions for NKS2D-224. NIST mailing list (local link), 2008.

[10] Brandon Enright. Collisions for NKS2D-512. NIST mailing list (local link), 2008.

[11] Brandon Enright. Near and truncated collisions in Spectral Hash. NIST mailing list (local link), 2008.

[12] Bjrn Fay. MeshHash. Submission to NIST, 2008.

[13] Niels Ferguson, Stefan Lucks, Bruce Schneier, Doug Whiting, Mihir Bellare, Tadayoshi Kohno, Jon Callas, and Jesse Walker. The Skein Hash Function Family. Submission to NIST, 2008.

[14] Praveen Gauravaram, Lars R. Knudsen, Krystian Matusiewicz, Florian Mendel, Christian Rechberger, Martin Schlffer, and Sren S. Thomsen. Grstl – a SHA-3 candidate. Submission to NIST, 2008.

[15] Danilo Gligoroski, Vlastimil Klima, Svein Johan Knapskog, Mohamed El-Hadedy, Jrn Amundsen, and Stig Frode Mjlsnes. Cryptographic Hash Function BLUE MIDNIGHT WISH. Submission to NIST, 2008.

[16] Danilo Gligoroski, Rune Steinsmo degrd, Marija Mihova, Svein Johan Knapskog, Ljupco Kocarev, and Ale Drpal. Cryptographic Hash Function EDON-R. Submission to NIST, 2008.

[17] Louis Goubin, Mickael Ivascot, William Jalby, Olivier Ly, Valerie Nachef, Jacques Patarin, Joana Treger, and Emmanuel Volte. CRUNCH. Submission to NIST, 2008.

[18] Shai Halevi, William E. Hall, and Charanjit S. Jutla. The Hash Function Fugue, 2008.

[19] Bob Hattersley. Waterfall Hash - Algorithm Specification and Analysis. Submission to NIST, 2008.

[20] Phil Hawkes and Cameron McDonald. Submission to the SHA-3 Competition: The CHI Family of Cryptographic Hash Algorithms. Submission to NIST, 2008.

[21] Sebastiaan Indesteege. Collisions for enrupt. Available online, 2008.

[22] Sebastiaan Indesteege. The LANE hash function. Submission to NIST, 2008.

[23] Robert J. Jenkins Jr. [algorithm specification.

[24] John Kelsey and Bruce Schneier. Second Preimages on n-bit Hash Functions for Much Less than 2^n Work. Cryptology ePrint Archive, Report 2004/304, 2004. http://eprint.iacr.org/.

[25] Dmitry Khovratovich and Ivica Nikoli. Cryptanalysis of enrupt. Available online, 2008.

[26] Dmitry Khovratovich and Ivica Nikoli. Cryptanalysis of DCH-n. Available online, 2008.

[27] Dmitry Khovratovich, Ivica Nikoli, and Ralf-Philipp Weinmann. Cryptanalysis of edon-r. Available online, 2008.

[28] Lars R. Knudsen, Florian Mendel, Christian Rechberger, and Sren S. Thomsen. Collision and Preimage Attacks on Vortex as submitted to the SHA-3 competition. Available online, 2008.

[29] Michael Kounavis and Shay Gueron. Vortex: A New Family of One Way Hash Functions based on Rijndael Rounds and Carry-less Multiplication. Submission to NIST, 2008.

[30] Gatan Leurent, Charles Bouillaguet, and Pierre-Alain Fouque. Simd is a message digest. Submission to NIST, 2008.

[31] Stefan Lucks. Design principles for iterated hash functions. Cryptology ePrint Archive, Report 2004/253, 2004. http://eprint.iacr.org/.

[32] Smile Markovski and Aleksandra Mileva. 2.B Algorithm Specifications and Supporting Documentations. Submission to NIST, 2008.

[33] Smile Markovski and Aleksandra Mileva. 2.B.1 Algorithm Specification. Submission to NIST, 2008.

[34] Jason Worth Martin. ESSENCE: A Candidate Hashing Algorithm for the NIST Competition. Submission to NIST, 2008.

[35] Mikhail Maslennikov. SECURE HASH ALGORITHM MCSSHA-3. Submission to NIST, 2008.

[36] Peter Maxwell. Aww, p*sh! Available online, 2008.

[37] Peter Maxwell. The Sgil Cryptographic Hash Function. Submission to NIST, 2008.
[38] Miguel Montes and Daniel Penazzi. The TIB3 Hash. Submission to NIST, 2008.
[39] Mara Naya-Plasencia. Second preimage attack on Ponic. Available online, 2008.
[40] Ivica Nikoli. Preimage attack on Sarmal-512. Available online, 2008.
[41] Ivica Nikoli. Preimage attack on Boole-n. Available online, 2008.
[42] National Institute of Standards and Technology. Cryptographic Hash Project. See http://csrc.nist.gov/groups/ST/hash/index.html.
[43] National Institute of Standards and Technology. FIPS 180-1: Secure Hash Standard. April 1995. See http://csrc.nist.gov.
[44] National Institute of Standards and Technology. FIPS 180-2: Secure Hash Standard. August 2002. See http://csrc.nist.gov.
[45] National Institute of Standards and Technology. FIPS 180: Secure Hash Standard. 1993. See http://csrc.nist.gov.
[46] Sean O'Neil, Karsten Nohl, and Luca Henzen. Enrupt hash function specification. Submission to NIST, 2008.
[47] Geoffrey Park. NKS 2D Cellular Automata Hash. Submission to NIST, 2008.
[48] R. Rivest. The MD5 Message-Digest Algorithm, 1992.
[49] Ronald L. Rivest. The MD6 hash function – A proposal to NIST for SHA-3. Submission to NIST, 2008.
[50] Gregory G. Rose. Design and Primitive Specification for Boole. Submission to NIST, 2008.
[51] Joachim Rosenthal and Roxana Smarandache. Maximum distance separable convolutional codes. *Applicable Algebra in Engineering, Communication and Computing*, 10(1):15–32, 1999.
[52] Gokay Saldaml, Cevahir Demirkran, Megan Maguire, Carl Minden, Jacob Topper, Alex Troesch, Cody Walker, and etin Kaya Ko. Spectral Hash. Submission to NIST, 2008.
[53] Peter Schmidt-Nielsen. The Ponic Hash Function. Submission to NIST, 2008.
[54] Bruce Schneier and John Kelsey. Unbalanced feistel networks and block cipher design. In *Fast Software Encryption, 3rd International Workshop Proceedings*, pages 121–144. Springer-Verlag, 1996.
[55] Sren S. Thomsen. A near-collision attack on the Blue Midnight Wish compression function. Version 2.0, available online, 2008.
[56] Sren S. Thomsen. Second preimage attack on MeshHash. Available online, 2008.
[57] Kerem Varc, Onur zen, and elebi Kocair. Sarmal: SHA-3 Proposal. Submission to NIST, 2008.
[58] John Washburn. WAMM: A CANDIDATE ALGORITHM FOR THE SHA-3 COMPETITION. Submission to NIST, 2008.
[59] David A. Wilson. Constructing Second Preimages in the WaMM Hash Algorithm. Available online, 2008.
[60] David A. Wilson. The DCH Hash Function. Submission to NIST, 2008.
[61] Hongjun Wu. The Hash Function JH. Submission to NIST, 2008.