NISTIR 7620

Status Report on the First Round of the SHA-3 Cryptographic Hash Algorithm Competition

Andrew Regenscheid
Ray Perlner
Shu-jen Chang
John Kelsey
Mridul Nandi
Souradyuti Paul

National Institute of
Standards and Technology
U.S. Department of Commerce

NISTIR 7620

Status Report on the First Round of the SHA-3 Cryptographic Hash Algorithm Competition

Andrew Regenscheid
Ray Perlner
Shu-jen Chang
John Kelsey
Mridul Nandi
Souradyuti Paul
Information Technology Laboratory
National Institute of Standards and Technology
Gaithersburg, MD 20899-8930

September 2009

U.S. Department of Commerce
Gary Locke, Secretary

National Institute of Standards and Technology
Patrick D. Gallagher, Deputy Director

Abstract

The National Institute of Standards and Technology is in the process of selecting a new cryptographic hash algorithm through a public competition. The new hash algorithm will be referred to as "SHA-3" and will complement the SHA-2 hash algorithms currently specified in FIPS 180-3, Secure Hash Standard. In October, 2008, 64 candidate algorithms were submitted to NIST for consideration. Among these, 51 met the minimum acceptance criteria and were accepted as First-Round Candidates on Dec. 10, 2008, marking the beginning of the First Round of the SHA-3 cryptographic hash algorithm competition. This report describes the evaluation criteria and selection process, based on public feedback and internal review of the first-round candidates, and summarizes the 14 candidate algorithms announced on July 24, 2009 for moving forward to the second round of the competition. The 14 Second-Round Candidates are BLAKE, BLUE MIDNIGHT WISH, CubeHash, ECHO, Fugue, Grøstl, Hamsi, JH, Keccak, Luffa, Shabal, SHAvite-3, SIMD, and Skein.

KEY WORDS:

SHA-3; cryptographic hash algorithm; cryptographic hash function; cryptography

Acknowledgements

NIST is grateful for the efforts of all SHA-3 candidate submitters, who developed and analyzed new hash algorithm designs and prepared detailed submission packages describing their algorithms. The cryptographic community's response of 64 candidate algorithms submitted to the competition was very encouraging. We were also pleased about the graceful and forthright manner with which several of the submitters recognized and accepted the shortcomings of their submissions.

NIST is also grateful for the efforts of those in the cryptographic community that provided security, implementation and performance analysis of the candidate algorithms during the first round, including the e-BASH and SHA-3 Zoo organizers, those involved with papers dealing with the effects of the AES instruction, hardware implementations, and memory requirements, as well as the numerous cryptanalysts that attacked many of the submissions. NIST would not be able to select a new hash algorithm for the SHA-3 standard without the combined efforts of these individuals and the algorithm submitters.

The authors of this report are also appreciative of the efforts by the other members of NIST's SHA-3 team, who reviewed candidate algorithms, analysis and public comments; performed testing; provided technical and administrative support; and participated in numerous meetings to discuss the selection of the first- and second-round candidates. They are: Elaine Barker, Lawrence Bassham, William Burr, Sara Caswell, Lily Chen, Quynh Dang, Donna Dodson, Morris Dworkin, James Nechvatal, and Rene Peralta.

Table of Contents

1 Introduction

The National Institute of Standards and Technology (NIST) is in the process of selecting a new cryptographic hash algorithm through a public competition. The new hash algorithm will be referred to as "SHA-3" and will complement the SHA-2 hash algorithms currently specified in Federal Information Processing Standard (FIPS) 180-3, Secure Hash Standard [1]. The selected algorithm is intended to be suitable for use by the U.S. government, as well as the private sector and, at the completion of the competition, to be available royalty-free worldwide. The competition will be referred to as the SHA-3 competition hereafter in this document.

The competition is NIST's response to recent advances in the cryptanalysis of hash algorithms, including the government standard SHA-1 hash algorithm [1, 2]. An attack by Xiaoyun Wang, Yiqun Lisa Yin, and Hongbo Yu [3], and extended by many others, has seriously called into question the security of SHA-1's use in digital signatures and other applications that require collision resistance. While the SHA-2 family [1] of hash algorithms provides an immediate alternative, NIST expects the selected SHA-3 to offer security that is at least as good as the SHA-2 algorithms with significantly improved efficiency or additional features.

In preparation for the SHA-3 competition, NIST held workshops on October 31-November 1, 2005 [4] and August 24-25, 2006 [5] to discuss the status of hash algorithms and develop a path forward for developing a new hash algorithm standard. As a result, NIST instituted a public competition, similar to that used to select the Advanced Encryption Standard (AES) [6, 7].

NIST published a *Draft requirements and evaluation criteria of the SHA-3 algorithm* in a Federal Register Notice in January 2007 (FRN-Jan07) for public comment [8]; these requirements and evaluation criteria were updated, based on public feedback, and included in a later, second Federal Register Notice published on Nov. 2, 2007 (FRN-Nov07) [9], which called for a new Cryptographic Hash Algorithm (SHA-3) and marked the start of the competition.

Candidate submissions were due on October 31, 2008, at which time NIST received 64 submission packages. This was a great response from the worldwide cryptographic community, which had submitted 21 candidate algorithms for the AES competition in 1998 [10]. Of the 64 submissions, NIST announced the acceptance of 51 First-Round Candidates as meeting the minimum acceptance criteria on Dec. 10, 2008, marking the beginning of the First Round of the SHA-3 competition. Submission packages of the first-round candidates were posted online at www.nist.gov/hash-competition for public review.

NIST held the First SHA-3 Candidate Conference at K.U. Leuven, Belgium on February 25-28, 2009 [11], where submitters of the accepted first-round candidates were invited to present their algorithms. NIST also discussed the plan to narrow down the first-round candidates to a more manageable number for further studies by the summer of 2009 and start the second round of the competition. Thereafter, NIST received much feedback from the cryptographic community. Based on the public feedback and internal reviews of the first-round candidates, NIST announced the selection of 14 algorithms as Second-Round Candidates on July 24, 2009 to move forward to the second round of the competition.

Below is a timeline of major events with respect to the SHA-3 competition, up to the close of the first round.

- 10/31-11/1/2005 Cryptographic Hash Workshop, NIST, Gaithersburg, MD
- 8/24-8/25/2006 Second Cryptographic Hash Workshop, UCSB, CA
- 1/23/2007 Federal Register Notice - Announcing the Development of New Hash Algorithm(s) for the Revision of Federal Information Processing Standard (FIPS) 180–2, Secure Hash Standard
- 11/2/2007 Federal Register Notice - Announcing Request for Candidate Algorithm Nominations for a New Cryptographic Hash Algorithm (SHA-3) Family

  The SHA-3 competition began.
- 10/31/2008 SHA-3 Submission Deadline
- 11/1/2008 The First Round of the competition began.
- 12/10/2008 First-Round Candidates were announced. The public comment period on the first-round candidates began.
- 2/25-2/28/2009 First SHA-3 Candidate Conference, KU Leuven, Belgium
- 7/24/2009 The First Round ended and the Second Round began. Second-Round Candidates were announced. The public comment period on the second-round candidates began.

1.1 Purpose of this Document

The purpose of this document is to report on the **first round** of the SHA-3 competition. The document explains the evaluation and selection process for the first- and second-round candidates of the competition. This report focuses on the reasons why candidate algorithms were selected, rather than providing detailed justifications for why candidate algorithms were not selected to move to the next round.

1.2 Organization of this Document

Section 2 describes the determination of the first-round candidates using the minimum acceptability requirements defined in FRN-Nov07 for all SHA-3 submissions. It then describes the evaluation criteria and selection process used to ultimately select the second-round candidates.

Section 3 summarizes the 14 selected second-round candidates. For each candidate, we provide a brief description of the algorithm and the properties of the algorithm that interested us, as well as characteristics that caused some concern.

Section 4 describes the next steps in the competition, including provisions for allowable modifications to the second-round candidates and the evaluation process for selecting the SHA-3 finalists.

2 Evaluation Criteria and the Selection Process

2.1 Acceptance of First Round Candidates

NIST received 64 candidate algorithm submission packages by the October 31, 2008 entry deadline for the SHA-3 competition. Of these, NIST accepted 51 first-round candidates as meeting the minimum acceptance criteria for being “complete and proper submissions”, as defined in FRN-Nov07. These criteria included provisions for reference and optimized C code implementations, known-answer tests, a written specification, and required intellectual property statements. In addition, the algorithms were required to be implementable in a wide range of hardware and software platforms, support message digest sizes of 224, 256, 384, 512 bits, and support a maximum message length of at least 2^{64}–1 bits.

First Round Candidates

Abacus
ARIRANG
AURORA
BLAKE
Blender
BLUE MIDNIGHT WISH
BOOLE
Cheetah
CHI
CRUNCH
CubeHash
DCH
Dynamic SHA
Dynamic SHA2
ECHO
ECOH
EDON-R
EnRUPT
ESSENCE
FSB
Fugue
Grøstl
Hamsi
JH
Keccak
Khichidi-1
LANE
Lesamnta
Luffa
LUX
MCSSHA-3
MD6
MeshHash
NaSHA
SANDstorm
Sarmal
Sgàil
Shabal
SHAMATA
SHAvite-3
SIMD
Skein
Spectral Hash
StreamHash
SWIFFTX
Tangle
TIB3
Twister
Vortex
WaMM
Waterfall

These were the sole criteria used to judge the 64 SHA-3 entries. Other factors, such as security, cost, and algorithm and implementation characteristics of the candidates did not enter the review process prior to the first round, nor did cryptanalysis or performance data of a submission impact the acceptance of the first-round candidates.

2.2 Evaluation Criteria

FRN-Nov07 identified three broad categories of evaluation criteria that will be used to compare candidate algorithms throughout the SHA-3 competition. The three categories are: 1) security, 2) cost and performance, and 3) algorithm and implementation characteristics. These categories are described below, along with a discussion of how they impacted the first- and second-round candidate evaluations.

2.2.1 Security

As was the case for the AES competition, security is the most important factor when evaluating the candidate hash algorithms. However, there remains significant disagreement within the cryptographic community-at-large over what security definitions should be used to evaluate hash algorithms.

While initially proposed for use in digital signatures, cryptographic hash algorithms are used in a wide variety of applications, including message authentication codes, pseudorandom number generators, key derivation, and one-way functions for obfuscating password files. All of these applications have different security requirements.

A common security concept that captures the security properties desired of hash algorithms is the random oracle model [12]. It is often the assumption made in security proofs of cryptographic protocols using a hash algorithm. Unfortunately, no real hash algorithm actually acts like a random oracle [13] in all situations. Therefore, the most that could reasonably be demanded of the proposed SHA-3 algorithm is to resemble a random oracle as closely as possible, and that was the security definition, as described in FRN-Jan07, that NIST planned to use in evaluating the SHA-3 candidates. However, this security definition was strongly criticized by the cryptographic community on the grounds that it was highly subjective.

To address this concern, NIST enumerated, in the later FRN-Nov07, a number of more well-defined security properties that it expected the winning SHA-3 candidate to meet. NIST felt these security properties were sufficient to establish the security of almost all applications of cryptographic hash algorithms, and that they could be designed into most hash algorithm constructions at fairly minimal cost. These criteria were not intended as strict requirements for an algorithm to remain in the competition, but rather to give submitters suggested security targets.

At the First SHA-3 Candidate Conference in Leuven, NIST elaborated on the requirements, commenting on the comparative severity of various types of attacks. In part, the intent of this discussion was to resolve debates that were taking place on the SHA-3 e-mail discussion list, *hash-forum@nist.gov* [14], regarding how heavily to weigh the memory complexity and parallelizability of an attack in evaluating its severity. An additional intention, however, was to communicate that attacks that were significantly costlier than brute force collision searches might not disqualify a candidate, provided that there is reason to believe that these attacks could not be extended, and provided that other positive features of the algorithm were seen by NIST to outweigh any immediate questions about the algorithm’s security. This was meant to encourage further cryptanalysis of candidates that were wounded, but not clearly broken. These attacks will be reconsidered prior to selecting the SHA-3 finalists during the second round of the competition.

2.2.2 Cost and Performance

FRN-Nov07 identified cost as the second-most important criterion when evaluating candidate hash algorithms. In this case, cost includes computational efficiency and memory requirements.

Computational efficiency essentially refers to the speed of an algorithm. NIST expects SHA-3 to offer improved performance over the SHA-2 family of hash algorithms at a given security strength. Memory requirements refer both to code size and random-access memory (RAM) requirements for software implementations, as well as gate counts for hardware implementations.

FRN-Nov07 required all submitters to include performance estimates on the NIST reference platform, an Intel Core 2 Duo-based system in 32- and 64-bit modes, as well as for 8-bit processors.

Cost is of particular concern for constrained platforms. Several individuals at the First SHA-3 Candidate Conference stressed the need for supporting constrained platforms, including mobile devices, smart cards and radio-frequency identification (RFID) systems. RAM requirements are often the limiting factor on these types of platforms, but power (which usually reduces to computational efficiency) and the area required by a circuit to implement the algorithm may also be limiting factors.

2.2.3 Algorithm and Implementation Characteristics

The SHA-3 competition has received many candidate algorithms with new and interesting designs, and with unique features that are not present in the SHA-2 family of hash algorithms. Candidate algorithms with greater flexibility may be given preference over other algorithms. This includes algorithms capable of running efficiently on a wide variety of platforms, as well as algorithms that use parallelism or instruction set extensions to achieve higher performance. In addition, simple and elegant designs are preferable, in order to encourage understanding, analysis and design confidence.

2.3 Selection of Second Round Candidates

NIST selected 14 second-round candidates from the 51 first-round candidates using the evaluation criteria specified in FRN-Nov07. In relative order of importance, NIST considered the security, cost and performance, and algorithm and implementation characteristics of a candidate in selecting the second-round candidates.

For the security evaluation of an algorithm, NIST studied the security arguments presented in the submission package, as well as external cryptanalysis submitted to NIST via e-mail or posted online at websites, such as the ECRYPT SHA-3 Zoo [15]. NIST researchers also conducted internal cryptanalysis.

NIST considered not only attacks that directly demonstrated that a SHA-3 candidate fell short of NIST's stated security targets, but also attacks that have historically been precursors to more severe attacks on legacy hash algorithms. The most prominent examples are pseudo-collision attacks. Pseudo-collisions on compression functions directly violate the security assumption of the Merkle-Damgård hash construction [16, 17], which is the basis for most legacy hash algorithms and many of the SHA-3 candidates. Some designs, particularly sponge-like constructions [18, 19], have trivial pseudo-collisions which are simply implicit in their construction and would have no relevance for the security of the hash functions in question. Since these constructions are already designed to withstand trivial pseudo-collisions, nontrivial pseudo-collisions attacks on the same hash algorithms were considered less severe than similar attacks on constructions based on Merkle-Damgård.

After security, performance was the next most important criterion in selecting the second-round candidates. When evaluating the performance of the candidates, NIST considered the performance and memory estimates given by the submitters in their submission documentation and in their presentations at the First SHA-3 Candidate Conference. NIST also performed internal performance benchmarks using the 32- and 64-bit optimized code from the submission packages. In addition, NIST considered the external feedback and performance estimates provided by the cryptographic community, such as those on the eBASH website [20] or on the *hash-forum* email list. The latter includes an analysis of the memory requirements of the candidate algorithms [21], and an analysis of the impact of the AES instruction on a candidate algorithm's performance [22].

In the first round of the competition, NIST focused on the performance of software implementations on high-end platforms, with some consideration given to whether the candidate algorithms appeared to be particularly suitable or unsuitable for hardware implementations or constrained platforms.

As was NIST's stated intention at the Leuven conference, only hash algorithms that could be tuned to a level of performance roughly comparable to or better than that of the SHA-2 family of hash algorithms without compromising security were selected for the second round. In some cases, the submitted variant was too slow or required too much memory, but NIST felt that in some cases, tunable parameters could be adjusted to give acceptable performance without compromising security. In order to accept candidates on these grounds, the tunable parameter needed to be documented by the submitters, the change needed to have a reasonably predictable effect on both the security and performance of the algorithm, and variants of the algorithm with acceptable performance needed to still have enough security margin that they were not on the verge of being broken. Also relevant to performance calculations was the expectation that a number of designs could benefit greatly from features included in the current or next-generation of multi-purpose processors. These features include vector instructions, as well as instructions in upcoming Intel chips designed to optimize the performance of AES. After considering optimized implementations, including 64-bit implementations and implementations using *Single Instruction, Multiple Data* (SIMD) [23] or AES instructions, all candidate algorithms selected for the second round seem to be capable of offering performance competitive with the SHA-2 family.

Second Round Candidates

BLAKE
BLUE MIDNIGHT WISH
CubeHash
ECHO
Fugue
Grøstl
Hamsi
JH
Keccak
Luffa
Shabal
SHAvite-3
SIMD
Skein

In a few cases, a submitted design was selected in part for its uniqueness and elegance. NIST generally favored those designs that were based on clear design principles or otherwise illustrative of an innovative idea. NIST feels that the diversity of designs will provide an opportunity for cryptographers and cryptanalysts to expand the scope of ideas in their field, and it will also be less likely that a single type of attack will eliminate the bulk of the candidates remaining in the competition.

3 Summary of Second Round Candidates

Each of the candidates selected for round two is discussed below, including a summary of the basic design, performance characteristics, and known attacks. In addition, the discussion includes suggested areas that the submitters may wish to address in order to improve their candidate's chances of surviving into the final round.

3.1 BLAKE

BLAKE is a HAIFA [24] hash algorithm whose compression function is based on using a keyed permutation in a Davies-Meyer-like construction [25]. The keyed permutation is based on the internals of the ChaCha [26] stream cipher, extended over a large state, and derives its nonlinearity from the overlap of modular addition and XOR operations. The most innovative part of BLAKE is its keyed permutation.

BLAKE's performance is quite good. It has modest memory requirements and appears to be suitable for a wide range of platforms.

The most significant cryptanalytic results against BLAKE are those that attacked the reduced round versions [27, 28] and appear to pose no threat to the design.

3.2 BLUE MIDNIGHT WISH (BMW)

BMW is a wide-pipe Merkle-Damgård hash construction [29] with an unconventional compression function, where the nonlinearity is derived from the overlap of modular addition and XOR operations. The most innovative parts of the design are the compression function construction and the design of the permutations; much of the design is novel and unique amongst the second-round candidates.

BMW has very good performance and appears to be suitable for a wide range of platforms. It has modest memory requirements.

The most serious cryptanalytic results against BMW are from impractical pseudo-collision attacks, and practical near-collision attacks [30]. These results raise questions about the security of the design.

3.3 CubeHash

CubeHash is a sponge-like hash algorithm that is based on a fixed permutation. The permutation is extremely simple and elegant, using only additions, XORs, and rotations in a fixed and simple pattern. All nonlinearity in the hash algorithm is derived from the overlap of modular additions and XOR operations. The novel part of CubeHash is the fixed permutation.

CubeHash has two tunable parameters, and its original proposed set of parameters led to extremely poor performance. A consistent problem in evaluating CubeHash has been uncertainty about those parameters. The designer has now proposed a set of parameters (16, 32) which provide very good performance with the use of SIMD instructions. CubeHash has relatively modest memory requirements and appears to be suitable for implementation on a wide range of platforms.

CubeHash has received a large amount of external analysis, probably due to the simplicity of its design and the flexibility offered to attackers by the two different tunable parameters. This made a strong argument in favor of advancing CubeHash to the next round—it appeared to be the best-understood of the candidates. The best-known attacks are:

- A generic, completely impractical preimage attack on the 512-bit version, which was described by the designer [31, 32].
- A semi-free-start collision, based on easily-found fixed points in the permutation, and several related results (such as symmetries preserved by the permutation) which raise some questions about the design and might be useful in practical attacks in the future [31].
- A number of reduced-round and increased-input-size attacks, none of which appears to threaten either the currently proposed version or the originally proposed version [33, 34, 35].

Of these, we find the semi-free-start collision and the symmetry properties to be the most troubling at this time. The CubeHash submission package identifies these issues and argues that exploiting these properties, given the large state and relatively small injection of message data before each permutation, is about as hard as a brute-force collision search. Relatively simple tweaks could also remove the symmetry properties from the algorithm.

3.4 ECHO

ECHO is a wide-pipe hash algorithm following the HAIFA construction. Its compression function uses a keyed permutation; the counter and salt are used as the key, and the message and chaining value are used as inputs to the permutation. The permutation is quite novel, using a 2048-bit AES-like permutation in which the role of the substitution-box (S-box) [36] is played by a single AES round. The AES S-box provides all nonlinearity in this hash algorithm. By far the most interesting and unique part of this hash algorithm is the super-AES keyed permutation.

ECHO has acceptable performance on current high-end platforms, but requires hardware AES support to achieve impressive performance. ECHO requires a considerable amount of memory, but is expected to be otherwise suited for constrained platforms and hardware implementations.

The only known analytical result is a highly impractical distinguishing attack on the underlying permutation of a reduced round (7 out of 8) version of ECHO [37]. This attack appears to pose no threat to the overall ECHO design. We hope that the selection of ECHO as a second-round candidate will lead to more analysis of this unique hash algorithm design.

3.5 Fugue

Fugue is a variant of a sponge construction. Its compression function is based on a nonlinear shift register, maintaining a large state (thirty 32-bit words for the 256-bit version). The shift register incorporates a strengthened variant of the AES round function; all other operations are linear. Thus, all nonlinearity in this design is derived from the AES S-box. The most novel part of this design is the shift-register-based compression function, for which proofs and bounds on its differential probabilities were provided.

The performance of Fugue is acceptable, although the efficiency of current implementations is not particularly impressive. The use of SIMD instructions could yield better performance, although how much the performance could be improved is unknown at this time. Hardware support for AES may also improve its performance somewhat, but the impact of this will be limited, because of the use of the variant-AES round function. Fugue also maintains a large state, which may make it difficult to implement in constrained platforms.

We are aware of no external analysis of Fugue. We hope that its selection as a second-round candidate will lead to more analysis of this interesting hash algorithm.

3.6 Grøstl

Grøstl is a wide-pipe Merkle-Damgård hash construction with post-processing. Its compression function is a novel construction, involving two AES-like fixed permutations. All nonlinearity in the design is derived from the AES S-box. The most innovative part of Grøstl's design is the compression function construction.

Grøstl's performance is acceptable, but not especially impressive. Performance may be increased using hardware AES support, although the extent of these gains is unknown at this time. It has modest memory requirements.

The most serious attack on Grøstl is a semi-free-start collision attack on a reduced round variant that breaks 7 out of 10 rounds [37]. This attack raises some question about the security margin of the design.

3.7 Hamsi

Hamsi is a Merkle-Damgård hash construction with post-processing to block length-extension attacks. The compression function is constructed on a fixed permutation; the message is expanded using an error-detecting code to fill half the input block to the permutation, with the other half filled by the hash chaining value. The result is truncated and XORed with the hash chaining input, which is similar to the method used in Snefru. The Hamsi fixed permutation is a substitution-permutation network (SP network) [38], combining a single 4-bit S-box (taken from Serpent, and implemented using bit-slicing) with a linear mixing operation. All nonlinearity in the design is derived from that one S-box. The most innovative part of this design is the compression function construction; this is quite different from any other second-round candidate.

Hamsi requires the use of SIMD instructions to achieve acceptable performance in software. It has modest memory requirements.

The only results on Hamsi of which we are aware at present demonstrate low algebraic degree in the outputs of the compression function; whether this has any security implications for the hash algorithm is unclear.

3.8 JH

JH uses a novel construction, somewhat reminiscent of a sponge construction, to build a hash algorithm out of a single, large, fixed permutation. The fixed permutation is an SP network, combining two 4-bit S-boxes with a set of linear mixing operations and bit permutations. All nonlinearity in this design is derived from the S-boxes. The most innovative part of this design is the compression function construction, which XORs a 512-bit message block into the left half of the input of the fixed permutation, and then XORs the same message block into the right half of the output of the fixed permutation. The design of the fixed permutation is also new.

JH's performance is good, and has modest memory requirements. Unlike most second-round candidates, all output sizes of JH use the same function, but with different initial hash values and different amounts of truncation at the end.

The most serious cryptanalytic result on JH is a theoretical preimage attack on the 512-bit version [39, 40], which is barely cheaper than a brute force attack. As this attack does not appear to threaten the design, it does not concern us. However, the compression function construction of JH is not well-understood, and the submitter did not provide a great deal of analysis of this construction.

3.9 Keccak

Keccak follows the sponge construction and uses a large fixed permutation. The permutation can be seen as an SP-network with 5-bit wide S-boxes, or as a combination of a linear mixing operation and a very simple nonlinear mixing operation. The construction of the permutation is the most innovative part of the Keccak design.

Keccak performs well on high-end platforms and is expected to perform well across a wide range of platforms, as well as in dedicated hardware. The hash algorithm has modest memory requirements. Unlike most second-round candidates, Keccak uses a single design for all hash outputs.

The most significant cryptanalytic result on Keccak of which we are aware are distinguishing attacks against reduced round versions of the permutation [41]; however, these do not appear to threaten the security of the hash algorithm.

3.10 Luffa

Luffa is a variant of the sponge construction, using a linear mixing operation and several fixed 256-bit permutations in place of a single wider permutation. The fixed permutations are SP networks, which combine linear mixing operations with a single 4-bit wide S-box, and this S-box provides all nonlinearity in the design. The most innovative part of Luffa is the sponge construction.

Luffa provides good performance on high-end platforms and appears suitable for a wide variety of platforms. Substantial parts of the design are the same for different output sizes.

The most significant cryptanalytic result on Luffa of which we are aware is a pseudo-preimage attack on the squeezing steps of Luffa-384 and Luffa-512 [42]. This is a consequence of the structure of Luffa (XORing 256-bit permutation results together to generate an output) and does not appear to lead to a threat to the security of the hash algorithm.

3.11 Shabal

Shabal is a hash algorithm that is constructed using a novel chaining mode, which can be seen as a variant of a wide-pipe Merkle-Damgård hash construction. Its compression function is similarly innovative, based on a feedback shift register construction that combines the several inputs provided by the chaining mode efficiently. Nonlinearity in Shabal is derived from the overlap of XOR, modular addition, and bitwise AND operations. The entire design is very different from any other second-round candidate and has many new ideas.

Shabal's performance is good. However, it requires more working memory than most of the second-round candidates. The same internal function is used for all output sizes of Shabal.

Several observations regarding Shabal's compression function have been published, including powerful distinguishing attacks on the keyed permutation that forms its core [43, 44]. However, the attacks have not been claimed to directly threaten the security of the hash algorithm, and the submitters have modified the security proof of their chaining mode to require weaker assumptions that are not invalidated by the attacks. Nonetheless, the distinguishing attacks remain a concern.

3.12 SHAvite-3

SHAvite-3 is a HAIFA hash algorithm. The compression function is a keyed permutation that is used in the Davies-Meyer construction. The keyed permutation is a balanced Feistel network [45] (for the 256-bit case) or a pair of interwoven balanced Feistel networks (for the 512-bit case), with the F-function constructed from the AES round function. All nonlinearity in the whole construction relies upon the AES S-box. The most innovative part of the design is the decision to construct the keyed permutation in this way; however, SHAvite-3 is a conservative design, with relatively little new about it.

SHAvite-3 has acceptable performance on current high-end platforms, but hardware AES support could have a large impact on its performance, since the AES round function is used directly. Shavite-3 has modest memory requirements.

The most serious cryptanalytic results on SHAvite-3 are large numbers of zero pseudo-preimages for the compression function [46, 47]. However, these require the use of a specific counter value, which is used only for the final message block, where the pseudo-preimages apparently cannot be constructed. While this result appears to pose no direct threat to SHAvite-3, this unexpected property of the compression function is a source of concern, especially given the fact that the offending counter value is used in Shavite-3's current construction.

3.13 SIMD

SIMD is a wide-pipe Merkle-Damgård hash construction. Its compression function is constructed from a keyed permutation, in a variant of the Davies-Meyer construction. The keyed permutation is the most innovative part of this design; it uses a linear code with provable diffusion properties as the "key schedule," and uses four unbalanced Feistel networks that are reminiscent of the MD4 [48] and MD5 [49] round functions in an interleaved way as its round function. The nonlinearity in this design is provided by the overlap of modular addition and XOR operations and from the bitwise nonlinear functions.

SIMD can achieve very good performance, but only when vector instructions are available. It also has relatively large memory requirements, which raises concerns about its suitability for constrained platforms.

At present, we are aware of no analysis that raises questions about SIMD's security.

3.14 Skein

Skein is a variant of a Merkle-Damgård hash construction that is based on a novel tweakable block cipher and chaining mode. The compression function is used in a variant of the Matyas-

Meyer-Oseas [25] construction that is appropriate for a tweakable block cipher, and the submission provides proofs that the construction is secure, assuming a secure compression function and tweakable block cipher. The block cipher (called "Threefish") is constructed from a large number of very simple rounds and uses only three 64-bit operations—modular addition, bitwise XORing, and rotation. All nonlinearity in the hash algorithm is provided by the overlap of modular addition and XOR operations. The most innovative parts of Skein are the Threefish block cipher and the chaining mode.

Skein has good performance on high-end platforms, particularly in 64-bit mode, and is also expected to perform well in constrained platforms and in dedicated hardware implementations. It has modest memory requirements and benefits from the pipelining used in modern processors.

The most significant cryptanalytic results on Skein are distinguishing attacks against reduced-round versions of Threefish; these do not appear to pose a threat to the full hash algorithm at this time.

4 Conclusion and Next Steps

The announcement of the 14 second-round hash algorithm candidates, BLAKE, BLUE MIDNIGHT WISH, CubeHash, ECHO, Fugue, Grøstl, Hamsi, JH, Keccak, Luffa, Shabal, SHAvite-3, SIMD, and Skein, marks the start of the second round of the SHA-3 competition. This paper summarized the evaluation criteria used to select these candidate algorithms, and briefly described the basic design of the second-round candidates, along with advantages and disadvantages already noted in these submissions.

Submitters of the second-round candidates will be allowed to tweak their submissions to improve upon them if they wish, and fix any inconsistencies, problems or shortcomings in the specifications or source code. Any changes must be submitted to NIST by September 15, 2009 in a complete submission package, as defined in FRN-Nov07.

The next twelve months will consist of a public review on the remaining 14 second-round hash algorithm candidates. Many of the second-round candidates have little or no cryptanalysis by the cryptographic community-at-large. With the number of candidates substantially reduced from the first round, we hope that the combined efforts of the cryptographic community will evaluate the remaining candidates and provide NIST with feedback that supports or refutes the security claims of the submitters. We are also interested in additional performance data on each of the candidates. This includes optimized implementations written in assembly code or using instruction set extensions, and analyses of implementation suitability of candidate algorithms in constrained platforms, as well as performance data for hardware implementations.

NIST plans to host the Second SHA-3 Candidate Conference at the University of California-Santa Barbara on August 23-24, 2010, following Crypto 2010. Submitters of the second-round candidates will be invited to present their algorithms. Soon after the conference, in the fall of 2010, NIST plans to select approximately five finalists for the final round of the competition. Detailed plans for the final round will be provided at a later date.

5 References

[1] FIPS PUB 180-3 Secure Hash Standard (SHS), Information Technology Laboratory, National Institute of Standards and Technology, Oct. 2008, http://csrc.nist.gov/publications/fips/fips180-3/fips180-3_final.pdf

[2] Near-Collisions of SHA-0 (with Forthcoming Results on SHA-0 and SHA-1), Eli Biham, Rafi Chen, http://www.cs.technion.ac.il/users/wwwb/cgi-bin/tr-get.cgi/2004/CS/CS-2004-09.pdf

[3] Collision Search Attacks on SHA1, Xiaoyun Wang, Yiqun Lisa Yin, Hongbo Yu, February 13, 2005, http://www.c4i.org/erehwon/shanote.pdf

[4] Cryptographic Hash Workshop, October 31-November 1, 2005, NIST Gaithersburg, MD, http://csrc.nist.gov/groups/ST/hash/first_workshop.html

[5] The Second Cryptographic Hash Workshop, August 24-25, 2006, Santa Barbara, CA, http://csrc.nist.gov/groups/ST/hash/second_workshop.html

[6] Report on the Development of the Advanced Encryption Standard (AES), James Nechvatal, et al., Computer Security Division, Information Technology Laboratory, National Institute of Standards and Technology, Oct. 2000

[7] FIPS PUB 197 Announcing the ADVANCED ENCRYPTION STANDARD (AES), Information Technology Laboratory, National Institute of Standards and Technology, Nov. 2001, http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf

[8] Announcing the Development of New Hash Algorithm(s) for the Revision of Federal Information Processing Standard (FIPS) 180–2, Secure Hash Standard, Federal Register / Vol. 72, No. 14 / Tuesday, January 23, 2007 / Notices 2861, http://csrc.nist.gov/groups/ST/hash/documents/FR_Notice_Jan07.pdf

[9] Announcing Request for Candidate Algorithm Nominations for a New Cryptographic Hash Algorithm (SHA–3) Family, NIST, Federal Register / Vol. 72, No. 212 / Friday, November 2, 2007 / Notices, http://csrc.nist.gov/groups/ST/hash/documents/FR_Notice_Nov07.pdf

[10] Conference Report on the *First Advanced Encryption Standard (AES) Candidate Conference, Ventura, CA, August 20-22, 1998*, Edward Roback and Morris Dworkin, Journal of Research of the National Institute of Standards and Technology, Volume 104, Number 1, January–February 1999

[11] The First SHA-3 Candidate Conference, K.U. Leuven, Belgium, Feb. 25-28, 2009, http://csrc.nist.gov/groups/ST/hash/sha-3/Round1/Feb2009/index.html

[12] Random Oracles are Practical: A Paradigm for Designing Efficient Protocols, Mihir Bellare and Phillip Rogaway, ACM Conference on Computer and Communications Security 1993, pp. 62–73, http://cseweb.ucsd.edu/users/mihir/papers/ro.pdf

[13] The Random Oracle Methodology Revisited, Ran Canetti, Oded Goldreich and Shai Halevi, STOC 1998, pp. 209–218, http://arxiv.org/abs/cs.CR/0010019

[14] Hash Forum, http://csrc.nist.gov/groups/ST/hash/email_list.html

[15] ECRYPT SHA-3 Zoo, http://ehash.iaik.tugraz.at/wiki/The_SHA-3_Zoo

[16] One way hash functions and DES, Ralph Merkle, Advances in Cryptology - CRYPTO '89 Proceedings, Lecture Notes in Computer Science, Vol. 435, G. Brassard, ed, Springer-Verlag, 1989, pp. 428-446.

[17] A Design Principle for Hash Functions, I. Damgård, Advances in Cryptology - CRYPTO '89 Proceedings, Lecture Notes in Computer Science, Vol. 435, G. Brassard, ed, Springer-Verlag, 1989, pp. 416-427.

[18] Sponge Functions, G. Bertoni, et al, http://sponge.noekeon.org/SpongeFunctions.pdf

[19] On the Indifferentiability of the Sponge Construction, G. Bertoni, et al, EuroCrypt 2008, N. Smart (Ed.), LNCS 4965, pp. 181–197, 2008, http://sponge.noekeon.org/SpongeIndifferentiability.pdf

[20] eBASH: ECRYPT Benchmarking of All Submitted Hashes, http://bench.cr.yp.to/ebash.html

[21] A Study on RAM Requirements of Various SHA-3 Candidates on Low-cost 8-bit CPUs, Kota Ideguchi, Toru Owada, Hirotaka Yoshida, http://www.sdl.hitachi.co.jp/crypto/lesamnta/A_Study_on_RAM_Requirements.pdf

[22] The Intel AES Instructions Set and the SHA-3 Candidates, R. Benadjila, et al, http://crypto.rd.francetelecom.com/sha3/AES/paper/

[23] SIMD (Single Instruction, Multiple Data), http://en.wikipedia.org/wiki/SIMD

[24] A Framework for Iterative Hash Functions — HAIFA, Eli Biham, Orr Dunkelman, NIST Second Cryptographic Hash Workshop, Santa Barbara, August 2006, http://csrc.nist.gov/groups/ST/hash/documents/DUNKELMAN_NIST3.pdf

[25] Black-Box Analysis of the Block-Cipher-Based Hash-Function Constructions from PGV, John Black, Phillip Rogaway, and Tom Shrimpton, Advances in Cryptology - CRYPTO '02, Lecture Notes in Computer Science, vol. 2442, pp. 320-335, Springer, 2002.

[26] ChaCha, a variant of Salsa20, Daniel J. Bernstein, Jan 28, 2008, http://cr.yp.to/chacha/chacha-20080128.pdf

[27] Attacks on Round-Reduced BLAKE, Li Ji and Xu Liangyu, Sony China Research Laboratory, http://eprint.iacr.org/2009/238.pdf

[28] Round-Reduced Near-Collisions of BLAKE-32, Jian Guo and Krystian Matusiewicz, http://www.jguo.org/docs/blake-col.pdf?attredirects=0

[29] Design Principles for Iterated Hash Functions, Stefan Lucks, e-print (September 29, 2004), http://eprint.iacr.org/2004/253.pdf

[30] A near-collision attack on the *Blue Midnight Wish* compression function, Søren S. Thomsen, November 27, 2008, http://www2.mat.dtu.dk/people/S.Thomsen/bmw/nc-compress.pdf

[31] Inside the Hypercube, Jean-Philippe Aumasson, et al, http://eprint.iacr.org/2008/486.pdf

[32] Preimage attack on CubeHash512-r/4 and CubeHash512-r/8, Dmitry Khovratovich, Ivica Nikolic, and Ralf-Philipp Weinmann, University of Luxembourg, http://ehash.iaik.tugraz.at/uploads/6/6c/Cubehash.pdf

[33] Linearization Framework for Collision Attacks: Application to CubeHash and MD6, E. Brier, et al, conference paper listed on http://thomas.peyrin.googlepages.com/research.htm

[34] Collisions for CubeHash1/45 and CubeHash2/89, Wei Dai, http://www.cryptopp.com/sha3/cubehash.pdf

[35] NIST mailing list 2008-12-04, Jean-Philippe Aumasson, http://ehash.iaik.tugraz.at/uploads/a/a9/Cubehash.txt

[36] S-Box (Substitution-box), http://en.wikipedia.org/wiki/S-box

[37] Improved Cryptanalysis of the Reduced Grøstl Compression Function, ECHO Permutation and AES Block Cipher, Florian Mendel, et al, https://online.tu-graz.ac.at/tug_online/voe_main2.getVollText?pDocumentNr=106996

[38] Handbook of Applied Cryptography by Alfred J. Menezes, Paul C. van Oorschot, Scott A. Vanstone. Fifth Printing (August 2001) page 251.

[39] An Observation on JH-512, Florian Mendel and Søren S. Thomsen, http://ehash.iaik.tugraz.at/uploads/d/da/Jh_preimage.pdf

[40] The Complexity of Mendel and Thomsen's Preimage Attack on JH-512, Hongjun Wu, http://ehash.iaik.tugraz.at/uploads/6/6f/Jh_mt_complexity.pdf

[41] Cube Attacks on Cryptographic Hash Functions, Joel Lathrop, May 21, 2009, http://www.cs.rit.edu/~jal6806/thesis/thesis.pdf

[42] Practical Pseudo-Cryptanalysis of Luffa, Jia Keting, http://eprint.iacr.org/2009/224.pdf

[43] Observations on the Shabal keyed permutation, Lars R. Knudsen, Krystian Matusiewicz, Søren S. Thomsen, April 7, 2009, http://www2.mat.dtu.dk/people/S.Thomsen/shabal/shabal.pdf

[44] More on Shabal's permutation, Jean-Philippe Aumasson, Atefeh Mashatan, and Willi Meier, http://131002.net/data/papers/AMM09.pdf

[45] Handbook of Applied Cryptography by Alfred J. Menezes, Paul C. van Oorschot, Scott A. Vanstone. Fifth Printing (August 2001) page 251.

[46] Chosen-salt, chosen-counter, pseudo-collision for the compression function of SHAvite-3, Thomas Peyrin, NIST mailing list, 19 January 2009, http://ehash.iaik.tugraz.at/uploads/e/ea/Peyrin-SHAvite-3.txt

[47] OFFICIAL COMMENT: SHAvite-3, Mridul Nandi and Souradyuti Paul, http://csrc.nist.gov/groups/ST/hash/sha-3/Round1/documents/SHAvite3_Comments.pdf

[48] The MD4 Message Digest Algorithm, Rivest, R., RFC 1186, MIT, October 1990.

[49] The MD5 Message-Digest Algorithm, Rivest, R., RFC 1321, MIT and RSA Data Security, Inc, April 1992.