Demise of MD5 and SHA-1

Emerging New Hash

SHA-3

Stanisław Radziszowski

Computer Science
Rochester Institute of Technology
Rochester, New York, USA

November 2010

Abstract

A hash function $H : \{0,1\}^* \rightarrow \{0,1\}^m$ produces an m-bit digest of an arbitrary message, file, or even an entire file system. Typically, one wants hash functions to be easy to compute, but also infeasible to invert or to find collisions (pairs of inputs which hash to the same value). Hash functions are fundamental cryptographic primitives, and they are used extensively in authentication, preserving data integrity, digital signatures, and many other security applications. The two most widely used hash functions are MD5 (Message Digest, $m = 128$) and SHA-1 (Secure Hash Algorithm, $m = 160$), the latter supported by the US government as a standard FIPS-180-2. The collisions for MD5 were found four years ago, and by now they can be produced quickly by software available on the Net. The SHA-1 algorithm seems also to be in trouble (and other algorithms in the SHA family, with $m = 256, 384, 512$, might follow). No collisions for SHA-1 have been found so far, but attacks much better than the simple birthday attack approach have been designed. Breaking SHA-1 soon is a likely possibility.

On January 23, 2007, NIST (National Institute of Standards and Technology) announced an initiative to design a new hash for this century, the Advanced Hash Standard (AHS), to be dubbed SHA-3. The competition is open, submissions were due October 31, 2008, and it is planned to conclude in late 2012.

`http://www.csrc.nist.gov/pki/HashWorkshop/timeline.html`

These developments are quite similar to the recent history of symmetric block ciphers - breaking of the DES (Data Encryption Standard) and an emergence of the AES (Advanced Encryption Standard) in 2001 as the winner of a multiyear NIST competition.

These lectures give the background on hash function design, outline the attacks on MD5 and SHA-1, and overview the progress of the SHA-3 hash algorithm competition.

Hash - simple, powerful idea

anything
(email, program, document, movie, file system ...)

$$x = y$$

$$\Updownarrow^*$$

$$H(x) = H(y)$$

256 bits
(32 bytes, like this "napisze do ciebie z dalekiej pod" ... no more)

Hashes in Practice

Applications of (cryptographic) hashes

- hash then sign
- time-stamping
- data authentication
- checksumming
- PGP email
- shadow passwords
- networking: SSL, SSH, VPN
- signatures: DSA, DSS (FIPS 186)
- MACs, HMAC (FIPS 198)
- PRF, PRNG, diffusers
- stream ciphers

The Problem

Design a (cryptographic) *hash* function
$H : \{0,1\}^* \to \{0,1\}^m$ such that:

- H is *preimage resistant*, i.e. given z, it is infeasible to find any x such that $H(x) = z$
- H is *collision resistant*, i.e. it is infeasible to find any pair x and y such that $H(x) = H(y)$
- H is *resistant* to *second preimage-*, *zero preimage-* ($H^{-1}(0^m)$), *length extension-*, and other attacks.
- H is fast to compute, uses little memory
- H can operate in the streaming mode

Very LARGE bound on input length can be given, pick m as small as possible but still guaranteeing resistance properties

Merkle-Damgård iterated hash

Notation

$x \in \{0,1\}^*$ - input message
blocks $m_i(= x_i)$ all of length $|m_i| = b$
$M(x) = m_1 m_2 \cdots m_t$ - formatted input
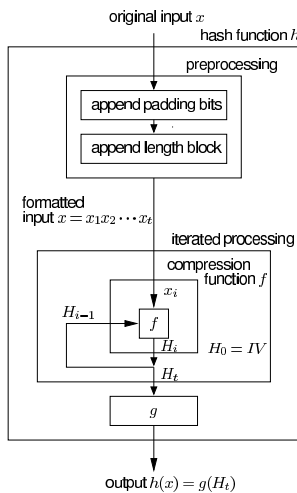m_t - padded, includes as tail $|x|$ in binary

IV - initialization vector
H_i - chaining variables
g - postprocessing function
$compress$ - a "kind" of OWF

$H(x)$
  $H_0 = IV$;
  for $i = 1, 2, \ldots, t$ do
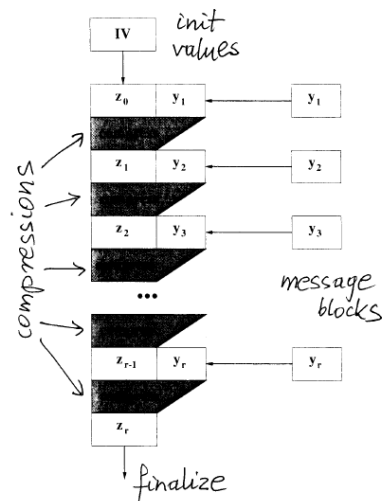 $H_i = compress(H_{i-1} \| m_i)$;
  return $H(x) = g(H_t)$;

Merkle-Damgård iterated hash

CRC Handbook of Applied Cryptography [13]

Merkle-Damgård iterated hash

Stinson textbook [19]

Merkle-Damgård iterated hash

Compression in r simpler rounds

Each m_i is "unfolded" into
message schedule m_{ij}, $1 \le j \le r$

Each round "absorbs" one m_{ij}

Finalization prevents length extension attack
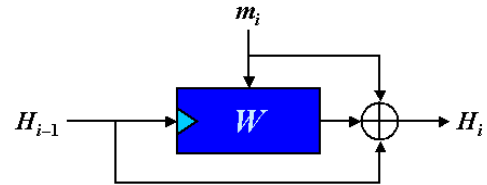
Why to design specially for hash?
Klima 2007 [8]

- all open vs. secret key in ciphers
- fixing key makes a permutation from
  ciphers, no need for this in hashes
- can get better performance

Compression types

Compression from block cipher

Using block cipher W with $key = H_{i-1}$
as compression function in Whirlpool

Miyaguchi-Preneel compression, Whirlpool [14, 18]

There are 12 other ways to safely embed
cipher into chaining, including designs by
Matyas-Meyer-Oseas and Davies-Meyer

Compression types

Chaining/collection type

- narrow pipe (MD) -
  "small" state, possibly good enough
- wide pipe (MD) -
  large internal state, prevents
  multicollision attacks, fixed points
- sponge (MD) - large internal state
  permuted after each absorption
- HAIFA (MD) - wide pipe with salt
  and bit count so far injected into each
  compression, prevents multicollision
  attacks, fixed points, herding
- hash tree collection (not MD) -
  permits natural parallelization

Compression types

Why from block cipher?
Biham 2005 [4]

- known much better than hashes
- there is no evidence that cipher design
  must lead to worse performance
- easily foil differential attacks
- no more multi-block attacks
- many rounds in hashes hide weaknesses,
  better use less but stronger rounds
- SHA-2 is just more of the same

Theory needs your help

Theorem
(most of the time - in various scenarios)
Resistant compression implies resistant hash.
Resistant hash implies resistant compression.

Problem
Find a way to study collision resistant
compression using complexity theory.
(more than in CRC Handbook 9.8.2 [13])

Characterize more formally:
"This n-to-m-bit compression needs
essentially $2^{m/2}$ tests to find a collision and
essentially 2^m effort to find any preimage."

People do it normally in random oracle model
in probabilistic combinatorics language.

Brief History (SHA-family biased)

- 1990 - **MD4**, Rivest, $m = 128$

- 1992 - **MD5**, Rivest, modified MD4

- 1993 - **SHA-0**, NIST, MD-like design

- 1995 - **SHA-1**, FIPS-180-1 $m = 160$

- 2002 - **SHA-2** family, NIST, FIPS-180-2
  for $m = 256, 384$ and 512 bit digests [10]

Brief History, contd.

- 2003 - **Whirlpool**, Rijmen-Barreto [14],
  AES-like cipher W inside, $m = 512$

- 2004-2006 Wang, Yu, Yin, et al. [22, 23,
  24] collision attacks on MD5 and SHA-1

- 2007 - NIST calls for new designs [11]

- 2012 - **SHA-3** recommended to for use

All above hashes (so far) follow
Merkle-Damgård template.

Hashes in Practice

The two most used hash functions
both of Merkle-Damgård type

- **MD5**, Rivest 1992

  128 bit hash, 512 bit blocks
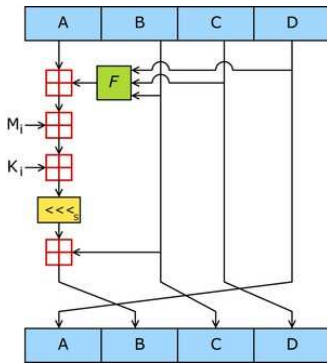 iterating 64-round compression c_{MD5}

  $$c_{MD5} : \{0,1\}^{640} \rightarrow \{0,1\}^{128}$$

- **SHA-1**, NSA/NIST 1995,
  created mainly for use in DSA

  160 bit hash, 512 bit blocks
  iterating 80-round compression c_{SHA-1}

  $$c_{SHA-1} : \{0,1\}^{672} \rightarrow \{0,1\}^{160}$$

MD5

MD5 round structure [12]
(Wikipedia)

each unit is a 32-bit word

MD5 needs MD4

MD4 definition in CRC [13]
collisions found in 1995

MD5 as an edit to MD4

MD5 as an edit of MD4 in CRC [13]

MD6 submitted to SHA-3 competition,
but no longer a candidate.

SHA-1
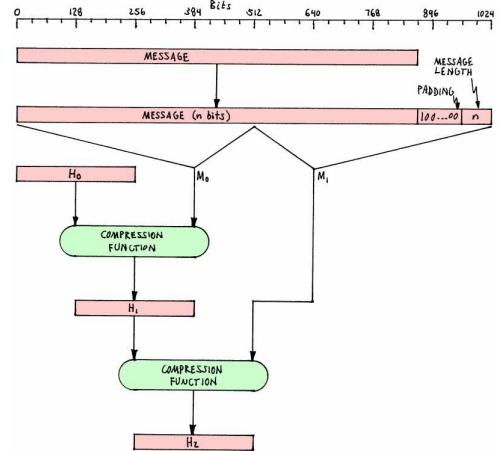
basic SHA-x structure
(Alan Kaminsky, RIT, 2004)

SHA-1 in Standards

HMAC
Keyed message authentication codes

key K and pads have 512 bits each

ipad = 3636 ... 36,
opad = 5C5C ... 5C,

$\mathrm{HMAC}_K(x) =$
   $\mathrm{SHA\text{-}1}((K \oplus opad \parallel \mathrm{SHA\text{-}1}((K \oplus ipad) \parallel x)).$

DSA and ECDSA
Elliptic Curve
Digital Signature Algorithm

...
$s = k^{-1}(\mathrm{SHA\text{-}1}(x) + mr) \pmod{q}$
...

HMAC
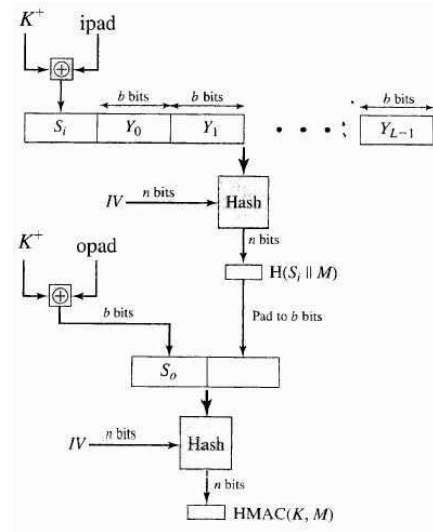
[Stallings]
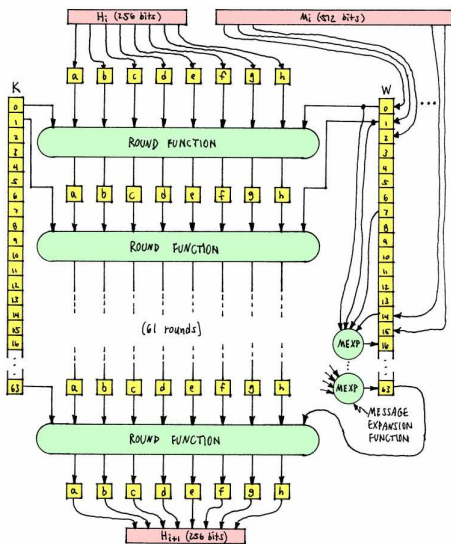
SHA-2

structure of SHA-2 compression (Alan Kaminsky)

SHA-2

FIPS 180-2, 2002
Modes for 224, 256, 384 and 512 bits

each unit is a 32-bit word
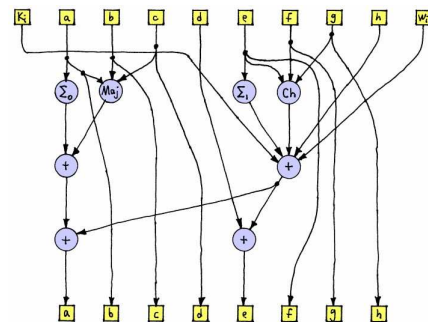
one round of SHA-256 compression
(Alan Kaminsky)

Birthday Attack

Counting fishes in a lake, Schnabel 1938 [16]

Theorem (Birthday Paradox)
Random sampling of q elements of the domain of size m will produce at least one collision with probability ϵ if

$$q \approx \sqrt{2m \ln \frac{1}{1-\epsilon}}$$

$q \approx 1.17\sqrt{m}$ for $\epsilon = 1/2$

$(m = 365, q = 23)$
Among 23 random people at least two of them have the same birthday with probability at least $1/2$.

Birthday Attack

Proof. [19]
The probability of collision ϵ satisfies

$$1 - \epsilon = \left(\frac{m-1}{m}\right)\left(\frac{m-2}{m}\right)\cdots\left(\frac{m-q+1}{m}\right)$$

$$= \prod_{i=1}^{q-1}\left(1 - \frac{i}{m}\right) \approx \prod_{i=1}^{q-1} e^{\frac{-i}{m}}$$

$$= e^{\sum_{i=1}^{q-1}\frac{-i}{m}} = e^{\frac{-q(q-1)}{2m}}.$$

Hence
$$\frac{-q(q-1)}{2m} \approx \ln(1-\epsilon).$$

After ignoring -1 in $-q(q-1)$
the theorem follows.

Generic Attack

Sheer power of computing

- 1998, effort 2^{56}, **DES**↓
- 2010, effort 2^{64} is possible
  $m = 2^{128}$, **MD5**↓
- 2020, effort 2^{80} may be feasible
  $m = 2^{160}$, **SHA-1**↓
- effort 2^{112}, won't be feasible for long

Conclusion. Requiring $m \geq 224$ for **AHS** seems reasonable (224 is the smallest multiple of 32 which prevents birthday attack well).

Preimage attacks are much more difficult, MD5 and SHA-1 are still strong.

Best potential attacks
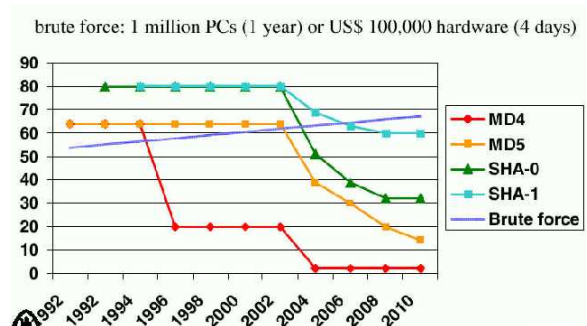
Complexity of collision attacks

(Bart Preneel, 2010)

Chinese attacks on MD5/SHA-1

Wang, Yu, Yin (1995 - 2004 - 2006)
Probabilistic differential cryptanalysis found collisions in **MD5** and other hashes [22-24].

- track simultaneously bitwise
  XOR and (mod 2^{32}) differences
- special difference bits in special rounds propagate with probability 1
- good differential paths
- multi-block manipulation
- heuristic approximation

Collisions for full 80 rounds CAN
be found (still not done) with

$$2^{80} \to 2^{69} \to 2^{63}$$

SHA-1 computations.

Vast experience and intuition were needed to develop this approach by hand.

Attacks on MD5/SHA-x (1996-2007)

- Collisions for **MD5** in seconds
- Collisions for **SHA**-1 likely soon (?)
- **SHA-2** not (yet) threatened
- Preimages almost hopeless

- No differential paths (much) better than those found in original attacks. Various attempts made to automate the search.
- Several authors (e.g. Black+ 2006 [6], Klima 2007 [8]) correct, experiment with and improve on hard to read Wang+ papers.
- Satoh, IBM Japan 2005 [15] - collisions for **SHA-1** COULD be found on a $10M special system in 127 days.

Attacks on MD5 (2008)

Chosen prefix collisions -
Stevens, Lenstra, de Weger

For any P, P' find S, S' such that

$$MD5(P \parallel S) = MD5(P' \parallel S')$$

Consecutive blocks of suffixes target to eliminate specific bit differences.

Converging paths
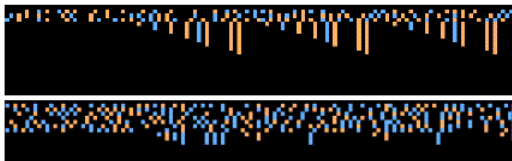
time down, black the same, colors 0-1, two cases

.

Faking documents

Lenstra, Wang, Yin (2005)
Given $x_1 \neq x_2 \wedge MD5(x_1) = MD5(x_2)$
construct two well-formed distinct RSA moduli n_1, n_2 with prefixes x_1, x_2.

This leads to two X.509 PKI certificates, differing only on the public key, but with the same MD5 hash.

Old trick on ASCII texts, philosophical
Any text has 2^k equivalent versions for any k. Thus, for any two texts there exist their equivalent versions colliding for SHA-x.

New trick on the Net, really scary
For any two texts one can effectively produce their postscript equivalents colliding for MD5 (same for pdf, WORD, tiff, ...).

Collisions abound

False Alarm?

```
                    SHA-1 Broken

SHA-1 has been broken.  Not a reduced-round version.
Not a simplified version.  The real thing.

The research team of Xiaoyun Wang, Yiqun Lisa Yin, and Hongbo Yu
(mostly from Shandong University in China) have been quietly
circulating a paper describing their results:

collisions in the full SHA-1 in 2**69 hash operations, much less than
the brute-force attack of 2**80 operations based on the hash length.

collisions in SHA-0 in 2**39 operations.

collisions in 58-round SHA-1 in 2**33 operations.
```

Still waiting for the real thing ...

Collision search for SHA-1 using the
distributed platform BOINC at GUT began
August 8, 2007, ... abandoned May 12, 2009
due to lack of progress.

Recommendations (2007)

- no more MD5
- NIST: SHA-1 out by 2010 (sic!)
- use SHA-x, $x \geq 224$
- in each case analyze which type of
  resistance is really needed, if only
  preimage then SHA-1 may stay around
  little longer
- design the new hash **AHS/SHA-3**,
  long time (30+ years) solution,
  should be parametrizable

General AHS design

- Must be resistant to all known attacks
- Small memory and long inputs seem
  to imply an iterated hash
- A compression doing less job than a block
  cipher was and can be risky
- Secure hashes from modular number
  theory are possible, but painfully slow
- Rather one parameterized hash than
  several special purpose hashes
- Take constants from math (like fractional
  part of $\sqrt[3]{p_i}$ in SHA-2, p_i the i-th prime).
  Constants in DES and SHA-1 are a
  mystery.

Block cipher based AHS

Nice properties

- Can be massively parallelized,
  NONE of the standard hashes can.
- Resistant to the length extension attack
- Immune to linear cryptanalysis
- Immune to differential cryptanalysis
- Good confusion
- Good diffusion
- Uses better understood components
- Incrementability. Small length-preserving
  message changes permit fast hash update.

Block cipher based AHS

Things to look at

- Get large blocks from ciphers like AES -
  Whirlpool, Maelstrom, Grøstl or JH.
- Compression must be fast, and so better
  be byte and word oriented, and easy to
  parallelize and pipeline in hardware.
- For software parallelizability use some tree
  result collection (MD6 or Skein).
- Tree-hashing: built-in or an afterthought
  mode of operation.

  The dilemma of
  SEQUENTIAL vs. PARALLELIZABLE
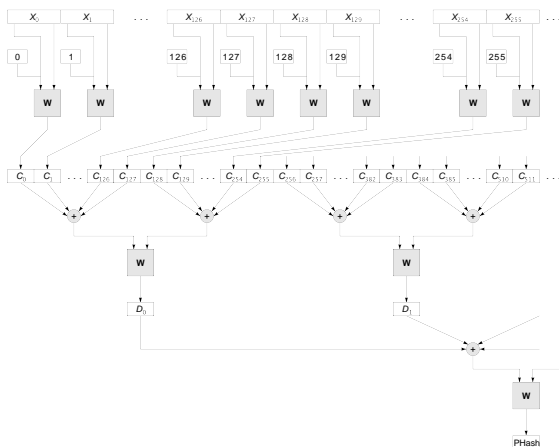
PHASH

Part of the PHASH computation tree
with cipher W, $L = 512$ and $R = 128$

(KaminskyR 2008, [7])

SHA-3 acceptability requirements

- **A.1**
  Free worldwide.

- **A.2**
  Implementable on varied hardware and
  software platforms.

- **A.3**
  Must support 224, 256, 384 and 512 bit
  digests, and messages of at least up to
  2^{64} bits.

SHA-3 submission requirements

August 31, 2008 - optional presubmission
October 31, 2008 - full submission

- **B.1** Completely specified, rationale given for choices made, attack scenarios and resistance analysis, parameterizable

- **B.2** Source in ANSI C

- **B.3** Time and space requirements for hardware and software for 8-, 32- and 64-bit platforms

- **B.4** Documentation in English

- **B.5** Issued or pending patents

- **B.6** Self-evaluation

SHA-3 evaluation criteria

- **C.1**
  Security

- **C.2**
  Cost (time and space complexity)

- **C.3**
  Algorithm and implementation characteristics (flexibility, parameterizable, easy to parallelize, and ... simplicity)

Hash Function Candidate competition timeline [11]

- 2007, 1-3Q - minimum requirements
- 2008, October 31 - submissions deadline
- 2009, public comments period
      2Q - First **HFC** Conference [26]
          Leuven, BE, February 25-28
- 2010, public comments period
      3Q - Second **HFC** Conference
          Santa Barbara, CA, August 23-24
      4Q - final round begins
- 2011, 4Q - end of public comments
- 2012, 1Q - Final **HFC** Conference [27]
      2Q - select the winner
      3Q - draft documents
          public comments, tuning up
      4Q - **SHA-3** proposed to
          the Secretary of Commerce

Hash Function Candidate competition timeline

What happened

- 2008, November-December, NIST
  64 submissions → 51 passed filtering
- 2009, February 25-28
  First HFC Conference
  Katholieke Universiteit Leuven, BE
- 2009, July 1
  MD6 by Rivest/MIT-team
  no further support, (no)-withdrawn
- 2009, July 24, NIST
  14 candidates qualified to the next round
- 2010, August 23-24
  Second HFC Conference
  U. of California Santa Barbara, CA

Current Hash Function Candidates

2010, November, 14 candidates [25]

What	Who	Where*
BLAKE	Aumasson+	CH
BMW	Knapskog	NO
CubeHash	Bernstein	IL
ECHO	Gilbert+	FR
Fugue	Jutla + IBM	NY
Grøstl	Knudsen+	DK
Hamsi	Küçük (f)	BE
JH	Wu	SG
Keccak	Daemen+ (m)	BE
Luffa	Watanabe+	JP
Shabal	Misarsky+	FR
SHAvite-3	Dunkelman+	Israel
SIMD	Leurent	FR
Skein	Schneier+	US/UK

*main countries/states of the main submitter

Hash Function Competition

What next?

- 2010, around December 15, NIST

  5-6 finalist candidates to be announced
  Big boss: Bill Burr
  Contents boss: John Kelsey

- Race is getting hot!

  Subscribe to hash-forum@nist.gov
  Email listproc@nist.gov, in the body

  > subscribe hash-forum "your name"

- 2011

  become an expert in one finalist function

- 2012

  become an expert in the winner function

SHA-3 Zoo

Hosted in Austria by
the Graz University of Technology [29]

- It aims to provide an overview of design
  and cryptanalysis of all submissions.
- Informal collection of documents, attacks,
  papers and opinions on SHA-3 hash
  candidates.
- Extensive benchmarking of SHA-3
  hardware implementations.
- NIST doesn't provide anything similar.

eBash

Hosted by Daniel Bernstein from
the University of Illinois at Chicago [30]

```
djb@cr.yp.to
```

- Part of a large system benchmarking
  cryptographic functions.
- By far the most extensive uniform
  software benchmarking of all SHA-3
  candidates, on many architectures.
- In places questionable taste:
  … suppose you want to submit your MD7 software to
  eBash, then do the following …

Hints by John Kelsey, NIST [27]

Classification

	Narrow-Pipe	MD	Wide-Pipe	MD	Sponge	Sponge-Like
Bitsliced	Hamsi		JH		Keccak	Luffa
AES	Shavite3		Echo	Grostl		Fugue
ARX	Skein	BLAKE	BMW		Cube	
Logical/ARX			SIMD	Shabal		

rows - source of nonlinearity

columns - type of design

ARX - addition, rotation, XOR

Hints by John Kelsey, NIST

Security

- no earth-shaking results
- will be used as KDF, PRF, PRNG
- Haifa designs with salt, sponges
- domain extenders
- do we count papers?
- no messagee schedule prevents message modification attacks
- rebound attacks
- do pseudocollisions matter?
- is there a "narrow pipe problem"?
- symmetries
- Alan randomness study

Hints by John Kelsey, NIST

Performance (narrow, wide, sponge)

Desktop	ASIC throughput	FPGA (ratio)
– BMW	– Luffa	– Keccak
– Shabal	– Keccak	– Cube
– Skein	– Cube	– Luffa
– SIMD	– Hamsi	– JH
– Luffa	– Blake	– Grostl
– Keccak	– Grostl	– Shabal
– Blake	– SHAvite3	– Blake
– JH	– JH	– Skein
– Cube	– BMW	– SHAvite3
– Grostl	– Shabal	– Fugue
– Hamsi	– Skein	– Hamsi
– Shavite3*	– Echo	– BMW
– Echo*	– Fugue	– Echo
– Fugue	– SIMD	– SIMD

Hints by John Kelsey, NIST

Performance

Patterns that jump out:

- ARX algorithms often optimized for S/W, not so great on H/W:

  Skein, BMW, SIMD, Shabal

- AES-based algorithms tend to be slow in S/W not so great in H/W either, but AES instruction *really* speeds up:
  SHAvite-3, Echo

- Bitsliced designs do pretty well in H/W and S/W:

  Keccak, Luffa do well, JH does okay
  Hamsi doesn't seem to do as well

  .

Tricky Comparisons 0

Performance, Daniel Bernstein
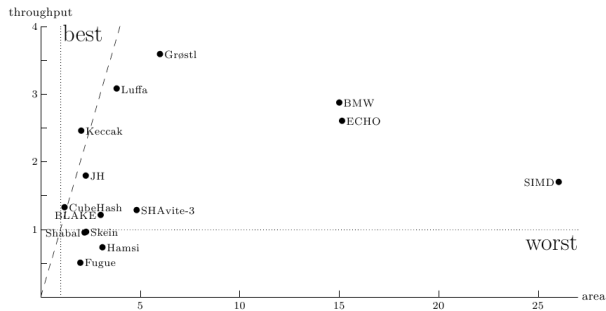
area - FPGA slices relative to SHA-2/512

throughput - Gbit/sec relative to SHA-2/512

Tricky Comparisons 1
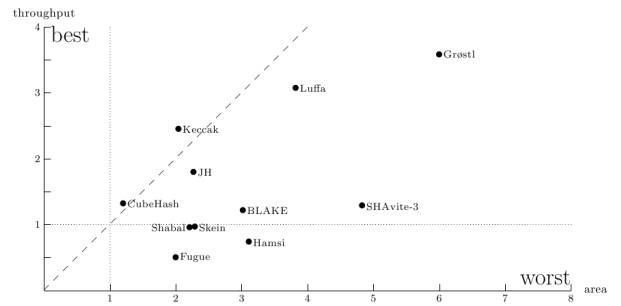
axes balanced

BMW, ECHO, SIMD now gone

Tricky Comparisons 2
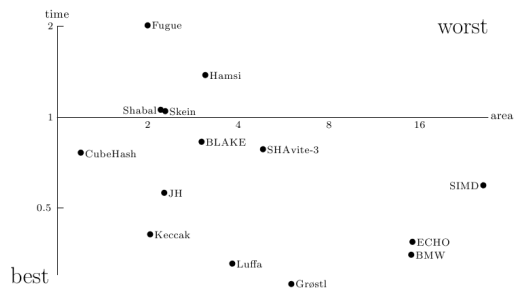
inverted cost (time)

logarithmic scale

Tricky Comparisons 3
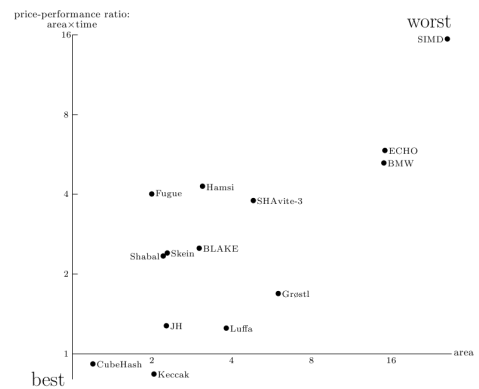
Price-performance ratio useful for

- cost of hashing heavy volumes of data

- hashing small number of extremely long messages with parallelizable hash

- hashing extremely large number of short messages with any hash

Hints by John Kelsey, NIST

What will matter for the final choice?

First call: security, cost, algorithm

Other factors

- diversity - not all eggs in one basket
- well interpreted number of papers
- 64-bit is becoming a standard
- SHA-3 will compete with SHA-2
- possible AES-chip-like solutions
- dual signatures
- updates to DSA and HMAC

Hash Summaries

The following 14 slides contain summaries of the hash function candidates, in alphabetical order, as announced by NIST in the report NISTIR 7620, September 2009 [25]

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

Use SHA-3 Zoo website to access the documentation of all candidates

`http://ehash.iaik.tugraz.at/wiki/The_SHA-3_Zoo`

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.

.

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.

.

CubeHash (5 dimensional cube)

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 ... (snip)

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.

.

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.

.

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.

.

Grøstl (austrian bigos)

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.

.

Hamsi (anchovy – sardela)

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.

.

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.

.

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.

.

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.

.

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-Damgrd 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.

.

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.

.

SIMD

SIMD is a wide-pipe Merkle-Damgrd 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.

.

Skein (flock of geese, oblong ball of yarn)

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.

.

References from hash summaries point to those listed in the NIST report [25].

References-1

1. Mihir Bellare, Roch Guérin and Philip Rogaway, XOR MACs: New Methods for Message Authentication Using Finite Pseudorandom Functions, *LNCS* **963**, (1995) 15–28.

2. Mihir Bellare and Daniele Micciancio, A new paradigm for collision-free hashing: incrementality at reduced cost, *LNCS* **1233**, (1997) 163–192.

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

4. Eli Biham, Recent Advances in Hash Functions: The Way to Go, slides from various conference presentations, 2005.

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

6. John Black, Martin Cochran and Trevor Highland, A Study of the MD5 Attacks: Insights and Improvements, *LNCS* **4047**, (2006) 262–277.

7. Alan Kaminsky and Stanisław Radziszowski, A Case for a Parallelizable Hash, *Proceedings of MILCOM'2008*, San Diego, CA.

8. Vlastimil Klima, posts at the NIST Cryptographic Hash Project forum, 2007, `hash-forum@nist.gov`.

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

References-2

10. National Institute of Standards and Technology, Cryptographic Toolkit, Secure Hashing, specification of SHA-2 standards, 2002, `http://csrc.nist.gov/CryptoToolkit/tkhash.html`

11. National Institute of Standards and Technology, Tentative Timeline of the Development of New Hash Functions, 2007, `http://www.csrc.nist.gov/pki/HashWorkshop/timeline.html`

12. MD5, `http://en.wikipedia.org/wiki/MD5`

13. Alfred J. Menezes, Paul C. van Oorschot and Scott A. Vanstone, *CRC Handbook of Applied Cryptography*, CRC Press 1996. `http://www.cacr.math.uwaterloo.ca/hac`

14. Vincent Rijmen and Paulo S. L. M. Barreto, The WHIRLPOOL Hash Function (2003), `http://paginas.terra.com.br/informatica/paulobarreto/ WhirlpoolPage.html`

15. Akashi Satoh, Hardware Architecture and Cost Estimates for Breaking SHA-1, *LNCS* **3650**, (2005) 259–273.

16. Zoe Emily Schnabel, The Estimation of the Total Fish Population of a Lake, *American Mathematical Monthly*, **45** (6), (1938) 348–352.

17. Bruce Schneier, *Applied Cryptography*, second edition, John Wiley & Sons, 1996.

18. William Stallings, The Whirlpool Secure Hash Function, *Cryptologia*, **30**, (2006) 55–67.

References-3

19. Douglas R. Stinson, *Cryptography: Theory and Practice*, third edition, CRC Press 2006.

20. Douglas R. Stinson, Some observations on the theory of cryptographic hash functions, *Designs, Codes and Cryptography* **38**, (2006) 259–277.

21. Janusz Stokłosa, Ochrona danych i zabezpieczenia w systemach teleinformatycznych, Wydawnictwo Politechniki Poznańskiej, 2005.

22. Xiaoyun Wang and Hongbo Yu, How to Break MD5 and Other Hash Functions, *LNCS* **3494**, (2005) 19–35.

23. Xiaoyun Wang, Hongbo Yu and Yiqun Lisa Yin, Efficient Collision Search Attacks on SHA-0, *LNCS* **3621**, (2005) 1–16.

24. Xiaoyun Wang, Yiqun Lisa Yin and Hongbo Yu, Finding Collisions in the Full SHA-1, *LNCS* **3621**, (2005) 17–36.

Prediction

Keccak will win

Meta References

Sources to other SHA-3 references:

25. NIST report NISTIR 7620, Status Report on the First Round of the SHA-3 Cryptographic Hash Algorithm Competition, September 2009, `http://csrc.nist.gov/groups/ST/hash`

26. 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`

27. The Second SHA-3 Candidate Conference, Santa Barbara, CA, Aug. 23-24, 2010, `http://csrc.nist.gov/groups/ST/hash/sha-3/Round2/Aug2010`

28. NIST Hash Forum `http://csrc.nist.gov/groups/ST/hash/email_list.html`

29. ECRYPT SHA-3 Zoo `http://ehash.iaik.tugraz.at/wiki/The_SHA-3_Zoo`

30. eBASH: ECRYPT Benchmarking of All Submitted Hashes, `http://bench.cr.yp.to/ebash.html`

Revisions

Revision #1, March 2007
Revision #2, August 2008
Revision #3, November 2010