State of the hash: SHA-3 and beyond

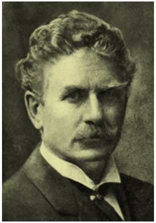
Jean-Philippe Aumasson

NAGRAVISION
KUDELSKI GROUP

Agenda

- Background material
- NIST's SHA-3 competition
- The SHA-3 candidate BLAKE
- Updated cryptanalysis of Skein
- On lightweight hashing (QUARK)
- Conclusions

What is a crypto hash?



HASH, x. There is no definition for this word—nobody knows what hash is.
Ambrose Bierce, The Devil's Dictionary

Arbitrary long string $\stackrel{H}{\rightarrow}$ Short random-looking string

a.k.a.

- Modification detection codes
- Message authentication codes (when keyed)
- Cryptographers' Swiss Army Knives

Applications of hash functions

Generation of secure keys

$$H(\text{physical entropy})$$

Key derivation

$$H(\text{salt, password})$$

Digital signatures

$$\text{Sign}\big(H(\text{salt, message})\big)$$

MAC's

$$H(\text{key, message})$$

Applications of hash functions

Passwords storage

$$H(\text{salt, password})$$

Forensics, e.g., proofs of non-modification

$$H(\text{key, evidence})$$

Random oracles in protocols, e.g. challenge-response

$$H(\text{key, random challenge})$$

Construction of pseudorandom generators

$$H(\text{key,nonce,1}), H(\text{key,nonce,2}), \ldots$$

Hash functions in standards: DSS, PKCS #1, NIST SP 800-108 (HMAC), -56a (key derivation), -106 (randomized hashing), etc.

Hash functions are ubiquitous and thus difficult to replace ($\approx$850 uses of MD5 in Windows [Ferguson, 2006])

Currently deployed hashes suffer weaknesses

Hash functions in standards: DSS, PKCS #1, NIST SP 800-108 (HMAC), -56a (key derivation), -106 (randomized hashing), etc.

Hash functions are ubiquitous and thus difficult to replace ($\approx$850 uses of MD5 in Windows [Ferguson, 2006])

Currently deployed hashes suffer weaknesses

$\Rightarrow$ we need good hash algorithms!

Difficult to define "goodness":

- Many different security requirements, sometimes difficult to formalize and weigh
- Many performance metrics and platforms (HW vs SW; speed vs. space, etc.)

Security requirements

- **Collision resistance** [*it should be hard to…*]
  find $M \neq M'$ s.t. $H(M) = H(M')$
- **Second preimage resistance**
  given M find $M' \neq M$ s.t. $H(M) = H(M')$
- **Preimage resistance**
  given $H(M)$ (but not M) find M' s.t. $H(M) = H(M')$

Security requirements

- **Collision resistance** [*it should be hard to...*]
  find $M \neq M'$ s.t. $H(M) = H(M')$
- **Second preimage resistance**
  given M find $M' \neq M$ s.t. $H(M) = H(M')$
- **Preimage resistance**
  given $H(M)$ (but not M) find M' s.t. $H(M) = H(M')$
- **Pseudorandomness**
  distinguish $H(K, \cdot)$ from a random function
- **Unpredictability**
  predict $H(K, M)$ for unqueried M's
- **Indifferentiability**
  find "related" sets of input/output values
- Etc.

Generic methods

i.e. that work for any n-bit hash function

- **Collision**: birthday search in $O(2^{n/2})$
- **(Second) preimage**: bruteforce search in $O(2^n)$
- **Pseudorandomness**: exhaustive search in $O(2^{|K|})$
- **Unpredictability**: exhaustive search in $O(2^{|K|})$
- **Indifferentiability**: depends on the relation

If a hash admits a method substantially faster than the best generic attack then it's "theoretically unideal"

An ideal hash function

SHA-2 (2002)

NSA design, 224-, 256-, 384-, or 512-bit digests

Only attacks on reduced versions, but suffers “length extensions”, “fixed points”, “multicollisions”, etc.

Current NIST recommendation

SHA-1 (1995)

NSA design, 160-bit digests

Almost practical collision attacks ($\approx 2^{60}$ vs. 2^{80} ideally)

"Federal agencies should stop using SHA-1 for digital signatures, digital time stamping and other applications that require collision resistance" [NIST]

MD5 (1991)

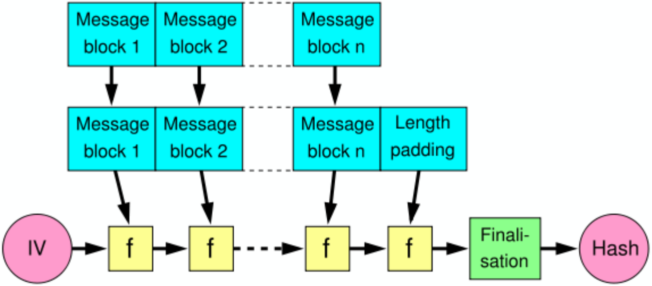
Ron Rivest design, 128-bit digests

Collisions can be found in milliseconds

Can find colliding executables, colliding public-keys, etc.

Should now be avoided

The Merkle-Damgård construction



Length extension: can find $H(M \| \text{padding} \| M')$ given only $H(M)$ (application: forgery of MAC's).

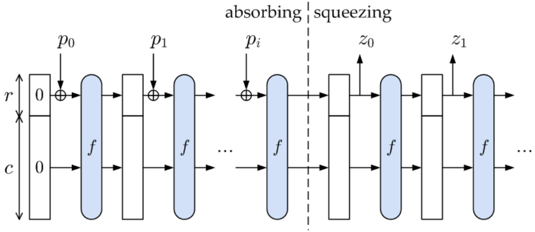
Beyond Merkle-Damgård

HAIFA [Biham-Dunkelman, 2006]

- Augmented version of Merkle-Damgård
- Counter to differentiate compression functions
- Proven "indifferentiable from a random oracle"

Sponge functions [Bertoni-Daemen-Peeters-Van Assche, 2007]

- Use a permutation rather than compression function
- Proven "indifferentiable from a random oracle"



NIST's SHA-3 competition

$\approx$ AES competition for hash functions

`http://nist.gov/hash-competition`

NIST's SHA-3 competition

- Nov 2007: SHA-3 competition announced
- Oct 2008: 64 submissions received
- Dec 2008: 51 accepted for first round
- Feb 2009: 1st SHA-3 Conference
- Jul 2009: 14 semifinalists announced
- Aug 2010: 2nd SHA-3 Conference
- end 2010: 4-6 finalists announced
- spring 2012: 3rd SHA-3 Conference
- sometime in 2012: winner announced

Formal requirements for SHA-3

- Support digests of 224, 256, 384, and 512 bits
- Support messages of at least 2^{64} bits
- Support HMAC, randomized hashing, PRFs
- Resistance to length extension
- One-pass mode

"NIST expects SHA-3 to have a security strength that is at least as good as the hash algorithms currently specified in FIPS 1802, and that this security strength will be achieved with significantly improved efficiency." [NIST]

64 submissions from all around the world

From both industry, academia, and government agencies

- EADS, Gemalto, Hitachi, IBM, Intel, Microsoft, Orange, Qualcomm, Sagem, Sony, STμ, etc.
- ENS (fr), EPFL, ETHZ (ch), KU Leuven (be), METU (tr), MIT (us), Weizmann Institute (il), etc.
- US Sandia Labs, French InfoSec Agency (DCSSI)

Great variety of designs

- AES-based
- ARX (Add, Rotate, Xor)
- Elliptic curve-based
- Parallel tree hashing (MD6)
- “Provably secure” (lattices, coding theory)

Many candidates broken

Abacus	Neil Sholer	in round 1	2nd-preimage	
ARIRANG	Jongin Lim	in round 1		
AURORA	Masahiro Fujita	in round 1	2nd preimage	
Blender	Colin Bradbury	in round 1	collision, preimage	near-collision
Boole	Greg Rose	in round 1	collision	
Cheetah	Dmitry Khovratovich	in round 1		length-extension
CHI	Phillip Hawkes	in round 1		
CRUNCH	Jacques Patarin	in round 1		length-extension
DCH	David A. Wilson	in round 1	collision	
Dynamic SHA	Xu Zijie	in round 1	collision	length-extension
Dynamic SHA2	Xu Zijie	in round 1	collision	length-extension
ECOH	Daniel R. L. Brown	in round 1	2nd preimage	
Edon-R	Danilo Gligoroski	in round 1	preimage	
EnRUPT	Sean O'Neil	in round 1	collision	
ESSENCE	Jason Worth Martin	in round 1	collision	
FSB	Matthieu Finiasz	in round 1		
HASH 2X	Jason Lee	not in round 1	2nd-preimage	
Khichidi-1	M. Vidyasagar	in round 1	collision	
LANE	Sebastiaan Indesteege	in round 1		
Lesamnta	Hirotaka Yoshida	in round 1		
LUX	Ivica Nikolić	in round 1	collision, 2nd preimage	DRBG,HMAC

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

Many candidates broken

Maraca	Robert J. Jenkins	not in round 1	preimage	
MCSSHA-3	Mikhail Maslennikov	in round 1	2nd preimage	
MD6	Ronald L. Rivest	in round 1		
MeshHash	Björn Fay	in round 1	2nd preimage	
NaSHA	Smile Markovski	in round 1	collision	
NKS2D	Geoffrey Park	not in round 1	collision	
Ponic	Peter Schmidt-Nielsen	not in round 1	2nd-preimage	
SANDstorm	Rich Schroeppel	in round 1		
Sarmal	Kerem Varıcı	in round 1	preimage	
Sgàil	Peter Maxwell	in round 1	collision	
SHAMATA	Orhun Kara	in round 1	collision	
Spectral Hash	Çetin Kaya Koç	in round 1	collision	
StreamHash	Michal Trojnara	in round 1	collision	
SWIFFTX	Daniele Micciancio	in round 1		
Tangle	Rafael Alvarez	in round 1	collision	
TIB3	Daniel Penazzi	in round 1	collision	
Twister	Michael Gorski	in round 1	preimage	
Vortex	Michael Kounavis	in round 1	preimage	
WaMM	John Washburn	in round 1	collision	
Waterfall	Bob Hattersley	in round 1	collision	
ZK-Crypt	Carmi Gressel	not in round 1		

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

The 14 round-2 candidates

Name	Submitter	Origin	Type
BLAKE	Aumasson		ARX
Blue Midnight Wish	Knapskog		ARX
CubeHash	Bernstein		ARX
ECHO	Gilbert		AES
Fugue	Jutla		AES
Groestl	Knudsen		AES
Hamsi	Küçük		S-box
JH	Wu		S-box
Keccak	Daemen		S-box
Luffa	Watanabe		S-box
Shabal	Misarsky		mix
SHAvite-3	Dunkelman		AES
SIMD	Leurent		mix
Skein	Schneier		ARX

Security

All round-2 candidates are yet unbroken

But some are less unbroken than others, due to

- Attacks for a large fraction of the total #rounds
- Attacks on building blocks (compression functions)
- Thin security margins

Impact of such results unclear, but reduces confidence. . .

(SHA-3 should not “look” weaker than SHA-2!)

Software benchmarks on eBASH

VAMPIRE

eBACS: ECRYPT Benchmarking of Cryptographic Systems

ECRYPT II

General information:	Introduction		eBASH	eBASC	eBATS	SUPERCOP	Computers
How to submit new software:	Hash functions		Stream ciphers	DH functions	Public-key encryption	Public-key signatures	
List of primitives measured:	SHA-3 candidates	All hash functions	Stream ciphers	DH functions	Public-key encryption	Public-key signatures	
Measurements indexed by machine:	SHA-3 candidates	All hash functions	Stream ciphers	DH functions	Public-key encryption	Public-key signatures	

eBASH: ECRYPT Benchmarking of All Submitted Hashes

The eBASH (ECRYPT Benchmarking of All Submitted Hashes) project, part of eBACS, measures hash functions according to the following criteria:

- Time to hash a 0-byte message.
- Time to hash a 1-byte message.
- Time to hash a 2-byte message.
- ...
- Time to hash a 4096-byte message. (Of course, longer messages are also of interest; for typical hash functions one can reasonably extrapolate to long messages by subtracting 2048-byte timings from 4096-byte timings.)
- Length of the hash output.

"Time" refers to time on real computers: time on an Intel Core 2 Quad, time on an AMD Athlon 64 X2, time on an IBM PowerPC G5 970, etc. The point of these cost measures is that they are directly visible to the cryptographic user. eBASH times each hash function on a wide variety of computers, ensuring direct comparability of all systems on whichever computers are of interest to the users.

There are separate pages explaining how to submit hash functions to eBASH, listing the hash functions already submitted to eBASH, and presenting the latest eBASH measurements.

Version

This is version 2010.09.03 of the ebash.html web page. This web page is in the public domain.

Maintained by Daniel J. Bernstein and Tanja Lange

`http://bench.cr.yp.to/`

Software benchmarks on eBASH

Based on the **SUPERCOP** toolkit (**S**ystem for **U**nified **P**erformance **E**valuation **R**elated to **C**ryptographic **O**perations and **P**rimitives)

Implementations are collected and included in the latest SUPERCOP release

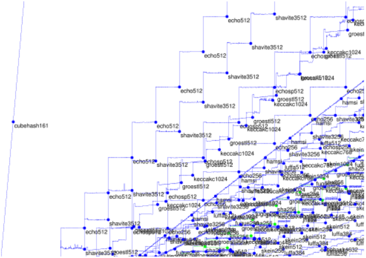
When run on your machine SUPERCOP does:

```
for each hash function H
  for each implementation of H
    for each relevant architecture (x86, amd64, etc.)
      for each compiler available (gcc, icc, xlc, etc.)
        for each compiler options set (from a defined set)
          measure speed for various message lengths
```

The fastest combinations are reported on
`http://bench.cr.yp.to/results-sha3.html`

eBASH example

x86, 2394MHz, Intel Core 2 Quad Q6600 (6fb), 2007, latour, supercop-20101002



Cycles/byte for long messages			
quartile	median	quartile	hash
4.91	4.94	5.00	bmw512
7.32	7.44	7.58	bmw256
9.70	9.99	10.41	shabal512
9.93	10.06	10.11	blake32
12.08	12.16	12.25	simd256
12.49	12.65	12.88	blake64
13.00	13.02	13.03	cubehash1632
13.46	13.59	13.68	simd512
14.08	14.15	14.24	luffa256
15.91	15.93	15.94	luffa384
17.10	17.12	17.14	skein512
18.00	18.00	18.01	skein256
19.21	19.29	19.33	keccakc448
18.93	19.32	20.04	jh224
19.18	19.34	19.62	jh512
19.30	19.41	19.56	jh256
18.98	19.49	20.06	jh384
19.59	19.70	19.76	fugue256
19.97	19.99	20.01	sha512
19.99	20.00	20.03	sha384
20.34	20.44	20.91	keccakc512
21.93	21.95	21.99	keccak

Cycles/byte for 4096 bytes			
quartile	median	quartile	hash
5.37	5.38	5.40	bmw512
7.72	7.76	7.82	bmw256
10.23	10.27	10.30	blake32
10.62	10.68	10.86	shabal512
12.49	12.51	12.54	simd256
13.14	13.15	13.27	blake64
14.18	14.19	14.19	cubehash1632
14.19	14.22	14.25	simd512
14.46	14.48	14.51	luffa256
16.42	16.42	16.43	luffa384
17.49	17.49	17.51	skein512
18.26	18.27	18.27	skein256
19.66	19.69	20.04	jh224
19.68	19.70	19.82	jh512
19.73	19.78	19.82	jh256
19.70	19.87	20.05	jh384
20.10	20.11	20.12	keccakc448
20.95	20.96	20.96	sha512
20.96	20.96	20.96	sha384
21.40	21.44	21.51	keccakc512
21.45	21.48	21.50	fugue256
22.78	22.78	22.80	keccak

Cycles/byte f		
quartile	median	qua
6.11	6.12	
8.26	8.27	
10.66	10.67	1
11.75	11.81	1
12.92	12.94	1
13.99	14.02	1
15.01	15.04	1
15.23	15.28	1
16.11	16.12	1
17.25	17.26	1
18.12	18.13	1
18.70	18.70	1
20.25	20.29	2
20.26	20.29	2
20.33	20.37	2
20.31	20.45	2
20.59	20.61	2
22.22	22.52	2
22.50	22.54	2
22.54	22.54	2
24.16	24.18	2
24.29	24.35	2

eBASH example

Implementation notes: x86, latour, crypto_hash

Computer: latour
Architecture: x86
CPU ID: GenuineIntel-000006fb-bfebfbff
CPU cycles/second: 2394000000...2394000000 (x86cpuinfo)
SUPERCOP version: 20101002
Benchmark dates: 20100903...20101003

crypto_hash

Time	Relative time	Primitive	Implementation	Compiler
16380	1.00	blake32	crypto_hash/blake32/sse2	gcc -funroll-loops -m32 -march=pentium-m -O2 -fomi frame-pointer (4.3.3)
17964	1.10	blake32	crypto_hash/blake32/ssse3	gcc -m32 -march=native -mtune=native -O2 -fomit-fr pointer
21006	1.28	blake32	crypto_hash/blake32/sphlib	gcc -m32 -march=pentiumpro -O2 -fomit-frame-pointe
30195	1.84	blake32	crypto_hash/blake32/ref	gcc -m32 -march=barcelona -O2 -fomit-frame-pointer
31338	1.91	blake32	crypto_hash/blake32/sphlib-small	gcc -funroll-loops -m32 -march=athlon -O3 -fomit-i pointer
21555	1.00	blake64	crypto_hash/blake64/sse2	gcc -funroll-loops -m32 -march=barcelona -O3 -fomi frame-pointer (4.3.3)
25776	1.20	blake64	crypto_hash/blake64/ssse3	gcc -m32 -march=core2 -msse4 -O -fomit-frame-point
74394	3.45	blake64	crypto_hash/blake64/sphlib	gcc -m32 -march=pentiumpro -O -fomit-frame-pointer
74907	3.48	blake64	crypto_hash/blake64/sphlib-small	gcc -m32 -march=native -mtune=native -O -fomit-fra pointer
80127	3.72	blake64	crypto_hash/blake64/ref	gcc -m32 -march=pentiumpro -O -fomit-frame-pointer
12636	1.00	bmw256	crypto_hash/bmw256/opt24ssse3_asm32 (Optimized_ICC_11.1_raw_asm_32bit_-_BMW256,_optc24ssse3)	gcc -m32 -march=core2 -Os -fomit-frame-pointer (4
12915	1.02	bmw256	crypto_hash/bmw256/opt31ssse3_asm32	gcc -m32 -march=core2 -msse4 -Os -fomit-frame-poi

Many open issues

How to make fair performance comparisons?

How important is each platform?

When is hash performance critical?

What's the impact of unexploited “vulnerabilities”?

Announcement of 4-6 finalists due in December 2010

BLAKE

Co-designed with Luca Henzen (ETHZ), Willi Meier (FHNW), Raphael C.-W. Phan (Uni Loughborough, UK)

Our design goals:

- Be faster and more secure than SHA-2
- Make the specs readable by non-experts
- Allow implementation space/time trade-offs
- Do not reinvent the wheel (build on previous designs)
  - Bernstein's ChaCha stream cipher
  - Biham and Dunkelman's HAIFA construction

How BLAKE works (compression function)

Initialize an internal state of 16 words

$$\begin{pmatrix} v_0 & v_1 & v_2 & v_3 \\ v_4 & v_5 & v_6 & v_7 \\ v_8 & v_9 & v_{10} & v_{11} \\ v_{12} & v_{13} & v_{14} & v_{15} \end{pmatrix}$$

Different inputs give different states

Round function: use of a bijective mapping **G** to transform each column, then each diagonal

$\mathbf{G}_0(v_0, v_4, v_8, v_{12})$ $\mathbf{G}_1(v_1, v_5, v_9, v_{13})$ $\mathbf{G}_2(v_2, v_6, v_{10}, v_{14})$ $\mathbf{G}_3(v_3, v_7, v_{11}, v_{15})$

$\mathbf{G}_4(v_0, v_5, v_{10}, v_{15})$ $\mathbf{G}_5(v_1, v_6, v_{11}, v_{12})$ $\mathbf{G}_6(v_2, v_7, v_8, v_{13})$ $\mathbf{G}_7(v_3, v_4, v_9, v_{14})$

How BLAKE works (compression function)

Initialize an internal state of 16 words

$$\begin{pmatrix} v_0 & v_1 & v_2 & v_3 \\ v_4 & v_5 & v_6 & v_7 \\ v_8 & v_9 & v_{10} & v_{11} \\ v_{12} & v_{13} & v_{14} & v_{15} \end{pmatrix}$$

Different inputs give different states

Round function: use of a bijective mapping **G** to transform each column, then each diagonal

$\mathbf{G}_0(v_0, v_4, v_8, v_{12})$ $\mathbf{G}_1(v_1, v_5, v_9, v_{13})$ $\mathbf{G}_2(v_2, v_6, v_{10}, v_{14})$ $\mathbf{G}_3(v_3, v_7, v_{11}, v_{15})$

$\mathbf{G}_4(v_0, v_5, v_{10}, v_{15})$ $\mathbf{G}_5(v_1, v_6, v_{11}, v_{12})$ $\mathbf{G}_6(v_2, v_7, v_8, v_{13})$ $\mathbf{G}_7(v_3, v_4, v_9, v_{14})$

How BLAKE works (compression function)

Initialize an internal state of 16 words

$$\begin{pmatrix} v_0 & v_1 & v_2 & v_3 \\ v_4 & v_5 & v_6 & v_7 \\ v_8 & v_9 & v_{10} & v_{11} \\ v_{12} & v_{13} & v_{14} & v_{15} \end{pmatrix}$$

Different inputs give different states

Round function: use of a bijective mapping **G** to transform each column, then each diagonal

$\mathbf{G}_0(v_0, v_4, v_8, v_{12})$ $\mathbf{G}_1(v_1, v_5, v_9, v_{13})$ $\mathbf{G}_2(v_2, v_6, v_{10}, v_{14})$ $\mathbf{G}_3(v_3, v_7, v_{11}, v_{15})$

$\mathbf{G}_4(v_0, v_5, v_{10}, v_{15})$ $\mathbf{G}_5(v_1, v_6, v_{11}, v_{12})$ $\mathbf{G}_6(v_2, v_7, v_8, v_{13})$ $\mathbf{G}_7(v_3, v_4, v_9, v_{14})$

How BLAKE works (G function)

For **BLAKE-32** — version with 32-bit words, 32-byte digests

$$
\begin{array}{ll}
\texttt{a += m}_i \oplus \texttt{const}_i & \\
\texttt{a += b} & \texttt{d = (a} \oplus \texttt{d)} \ggg 16 \\
\texttt{c += d} & \texttt{b = (b} \oplus \texttt{c)} \ggg 12 \\
\texttt{a += m}_j \oplus \texttt{const}_j & \\
\texttt{a += b} & \texttt{d = (a} \oplus \texttt{d)} \ggg 8 \\
\texttt{c += d} & \texttt{b = (b} \oplus \texttt{c)} \ggg 7
\end{array}
$$

How BLAKE works (G function)

For **BLAKE-64** — version with 64-bit words, 64-byte digests

$$
\begin{array}{ll}
\texttt{a += m}_i \oplus \texttt{const}_i & \\
\texttt{a += b} & \texttt{d = (a} \oplus \texttt{d)} \ggg 32 \\
\texttt{c += d} & \texttt{b = (b} \oplus \texttt{c)} \ggg 25 \\
\texttt{a += m}_j \oplus \texttt{const}_j & \\
\texttt{a += b} & \texttt{d = (a} \oplus \texttt{d)} \ggg 16 \\
\texttt{c += d} & \texttt{b = (b} \oplus \texttt{c)} \ggg 11
\end{array}
$$

BLAKE in software

Straightforward to implement (chain of $+, \oplus, \ggg$)

Speed-up with SIMD instructions (SSE2)

On Intel Core 2 Duo (eBASH's `katana`)

- BLAKE-32: 10.21 cycles/byte
  with `gcc -march=nocona -O -fomit-frame-pointer`
- BLAKE-64: 7.04 cycles/byte
  with `gcc -m64 -march=K8 -O2 -fomit-frame-pointer`
- vs. 15.32 (SHA-256), 11.63 (SHA-512)

268 bytes of RAM in ATmega1281 (8-bit, 8 Kb RAM)

BLAKE in hardware

Space/time trade-offs with 1, 2, 4, or 8 functions G

Own VHDL implementations, synthesis, chips fabrication

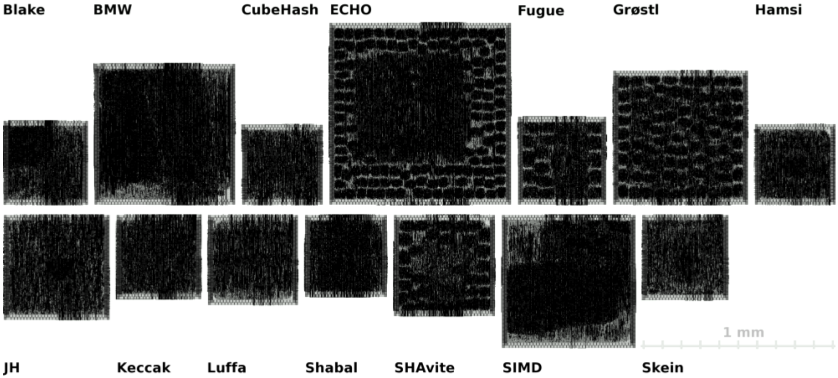
- BLAKE-32 in 13.5 kGE, 135 Mbps (180 nm)
- BLAKE-32 in 38 kGE, 15 Gbps (90 nm)
- BLAKE-64 in 79 kGE, 19 Gbps (90 nm)

Many third-party implementations, e.g. in FPGA

- BLAKE-32 with 56 Virtex 5 slices, 225 Mbs
- BLAKE-64 with 108 Virtex 5 slices, 314 Mbps

[Beuchat et al.]

SHA-3 candidates in hardware

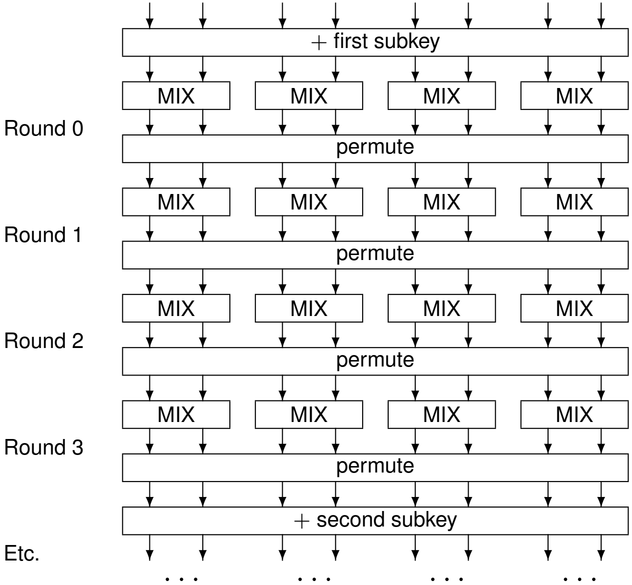


See SHA-3 hardware evaluation project by Henzen et al.
`http://www.iis.ee.ethz.ch/∼sha3/`

Skein

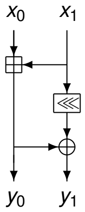
Design by Niels Ferguson, Stefan Lucks, Bruce Schneier, Doug Whiting, Mihir Bellare, Tadayoshi Kohno, Jon Callas, Jesse Walker

- One of the 14 round-2 candidates
- ARX (Add, Rotate, Xor) algorithm
- Only works with 64-bit words
- Based on the “Threefish” block cipher
  (“Twofish” was the team’s AES candidate)


+ first subkey
MIX MIX MIX MIX
Round 0
permute
MIX MIX MIX MIX
Round 1
permute
MIX MIX MIX MIX
Round 2
permute
MIX MIX MIX MIX
Round 3
permute
+ second subkey
Etc.
. . . .

Threefish's MIX function

At round $r \in \{0, 1, \ldots, 71\}$ and position $s \in \{0, 1, 2, 3\}$:



$$
\begin{aligned}
\mathrm{MIX}_{r,s}(x_0, x_1) &= (y_0, y_1) \\
y_0 &= x_0 + x_1 \\
y_1 &= y_0 \oplus (x_1 \lll R_{r,s})
\end{aligned}
$$

Skein's round-2 tweak

Skein was "tweaked" for round 2

- New rotation constants in MIX optimizing diffusion
- 2-day computation with a genetic algorithm

Expected to improve resistance to differential attacks as

- *Improved cryptanalysis of Skein*
  Asiacrypt '09, w/ Calik, Meier, Özen, Phan, Varıcı

"It is not clear to us whether the impossible differential can be modified for the new rotation constants."

How is our analysis affected?

Impossible differentials

The **miss-in-the-middle** technique:

Proof by contradiction that ($\alpha \to \gamma$) cannot occur

$$\alpha \xrightarrow{\textit{prob.}1} \beta \neq \delta \xleftarrow{\textit{prob.}1} \gamma$$

In practice, β and δ are differences over a subset of the internal state (that is, truncated differentials)

Impossible differentials were previously found for

- 8 rounds of AES-192 (of 12)
- 5 rounds of Twofish (of 16)
- 21 rounds of Threefish (of 72, round-1 version)

New results on the round-2 Threefish

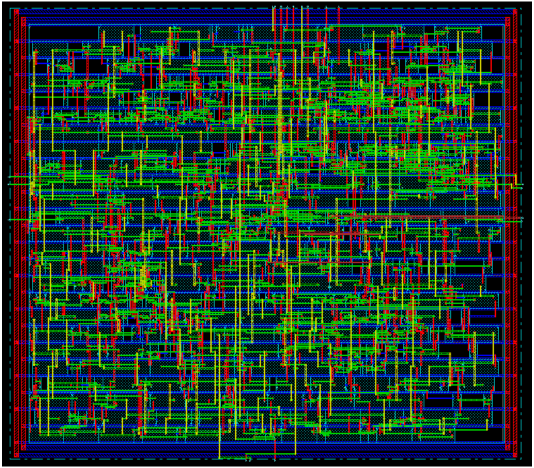
- Probability-1 truncated differential for 14 rounds
  (against 13 for the round-1 version!)
- Impossible differential for 21 rounds
  (same as for the round-1!)
- More efficient linearization than for round-1 version

⇒ Unclear whether the “improved” constants are better than the original ones...

Improved protecting against attack **A** may weaken resistance to attack **B**...

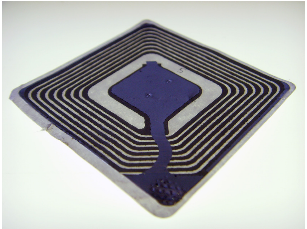
Work in progress with Raphael Phan...

And now for something completely different. . .



Towards lightweight hashing

Hashing in dedicated IC's (as RFID tags' chips)



- Identification protocols
- Message authentication

MD5 and SHA-1 generally too big (5000+ GE)

Smallest known proposal: PRESENT-based hashes by Bogdanov et al. (CHES 2008)

- 64-bit hash: 1600 GE
- 128-bit hash: 2330 GE

Our new hashes: the QUARK family

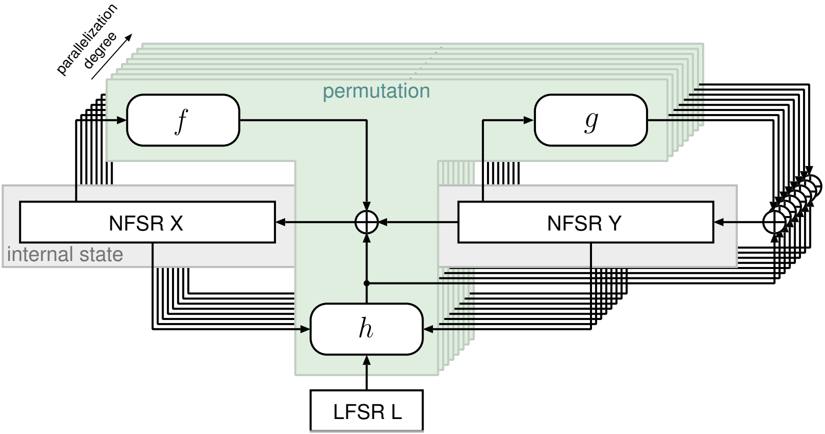
- *Quark: a lightweight hash*
  CHES '10, w/ Henzen, Meier, Naya-Plasencia

Design philosophy:

- Consider security as a parameter independent of the digest length
- Do not make “light versions” of known constructions, but use a design intrinsically “lightweight”

Lineage:

- Sponge functions
- Lightweight stream cipher Grain
- Lightweight block cipher KATAN


parallelization
degree
permutation
f
g
NFSR X
internal state
NFSR Y
h
LFSR L

Hardware performance of QUARK

	Security			Area	Thr.	Power [µW]	
	Pre	2nd	Col	[GE]	[kbps]	Mean	Peak
U-QUARK	128	64	64	1379	1.47	2.44	2.96
D-QUARK	160	80	80	1702	2.27	3.10	3.95
S-QUARK	224	112	112	2296	3.13	4.35	5.53
U-QUARK×8	128	64	64	2392	11.76	4.07	4.84
D-QUARK×8	160	80	80	2819	18.18	4.76	5.80
S-QUARK×16	224	112	112	4640	50.00	8.39	9.79

vs. 5000+ GE for SHA-1

Smaller than previous lightweight hashes

Straightforward speed/confidence trade-offs by varying #rounds

Summary

SHA-3 will augment the SHA-2 hash standard in 2012

- Hopefully faster and more confidence-inspiring
- 14 candidates left, 4-6 finalists in Dec 2010

SHA-2 still okay for most applications

- Theoretical attack on 43 rounds (of 64)
- Insecure in prefix-MAC

Future challenges:

- Ultra lightweight hashes
- Efficient “provably secure” hashes

Further reading

- NIST's Hash Competition
  http://nist.gov/hash-competition
- eBACS (benchmarks of crypto implementations)
  http://bench.cr.yp.to/
- The ECRYPT SHA-3 Zoo
  (submission packages, latest attacks/implementations)
  http://ehash.iaik.tugraz.at/wiki/The_SHA-3_Zoo
- BLAKE's webpage http://131002.net/blake/
- QUARK's webpage http://131002.net/quark/

PDF of these slides available at
http://131002.net/talks.html

State of the hash: SHA-3 and beyond

Jean-Philippe Aumasson

NAGRAVISION
KUDELSKI GROUP