BLAKE2

https://blake2.net

Another hash again?

Why not SHA-2, SHA-3?

Why not SHA-2, SHA-3?

BLAKE, Groestl, JH, Skein?

Why not SHA-2, SHA-3?

BLAKE, Groestl, JH, Skein?

BMW, CubeHash, MD6, Shabal?

focus on

software *(speed)*

&

usability

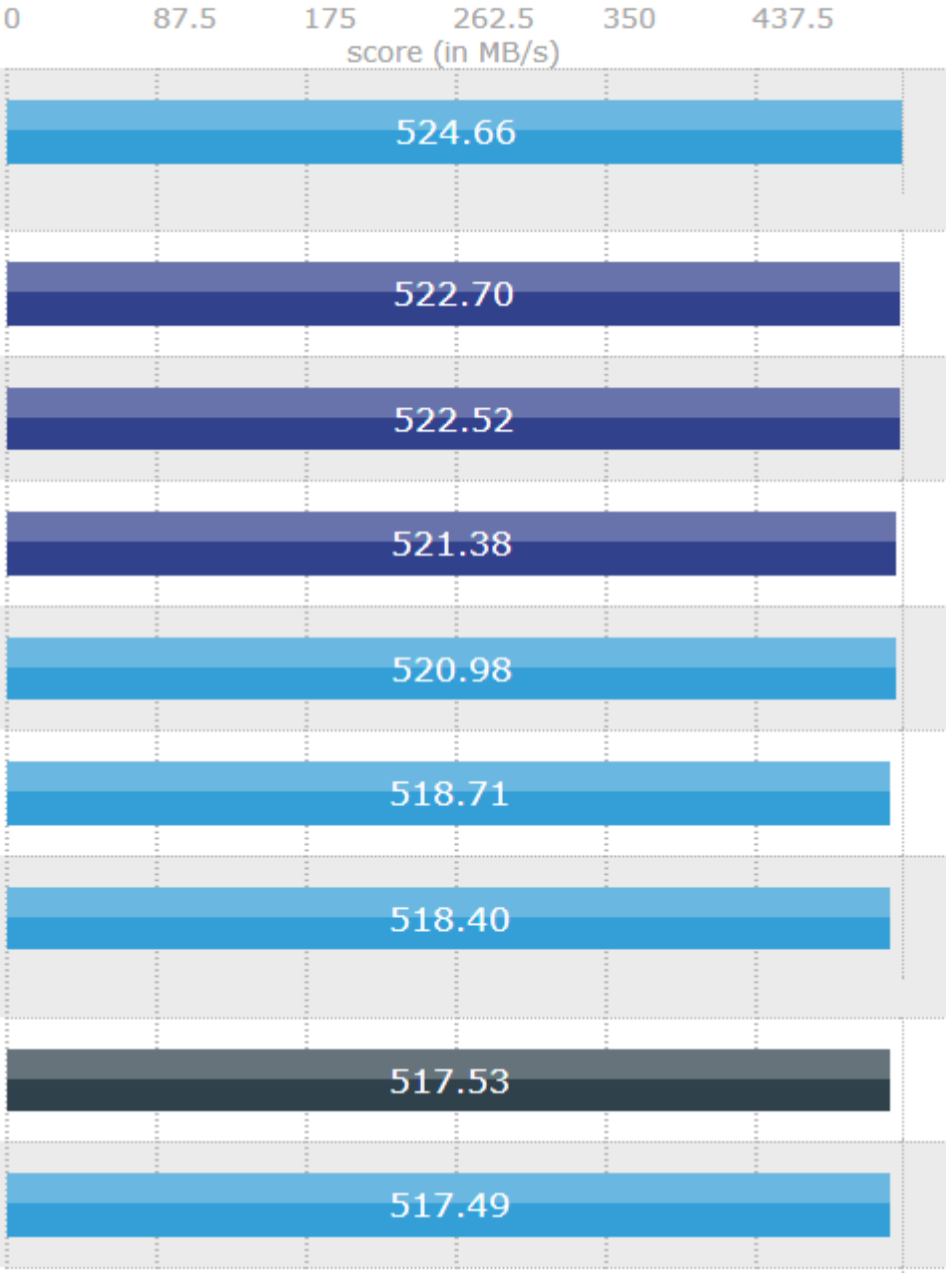
“But speed in software is limited by HDD latency”

True on legacy hardware, but recent SSDs R/W at ≈500MBps

on 2GHz Core i7 (SB)
SHA-256: 110 MiBps
SHA-512: 172 MiBps

www.tomshardware.com/charts/ssd-charts-2012/AS-SSD-Sequential-Read,2782.html

Compare



“OK, but software hashing is never a bottleneck”

Except in

cloud storage

advanced file systems

version control systems

intrusion detection systems

etc.

Example: Artillery IDS

hashes **all files** in **/etc/** and **/var/www/** every 60sec on a machine busy serving HTTP

~2000 small files in /etc/ of a basic install

but /var/www/ can be huge (GiBs)

(›200MiB on my tiny server…)

Example: ZFS (Sun/Oracle)

"Each block of data is checksummed and the checksum value is then saved in the pointer to that block (…) Next, the block pointer is checksummed (…) This checksumming continues all the way up the file system's data hierarchy to the root node, which is also checksummed, thus creating a Merkle tree."

http://en.wikipedia.org/wiki/ZFS#ZFS_Data_Integrity

Example: ZFS (Sun)

Integrity checking defaults to a weak checksum, because SHA-256 (optional) isn't fast enough

Deduplication in ZFS also requires fast hashing

Example: OpenStack Swift

Cloud storage system used by NASA, AT&T, etc.

"Objects and files are written to multiple disk drives spread throughout servers in the data center, with the OpenStack software responsible for ensuring data replication and integrity across the cluster."

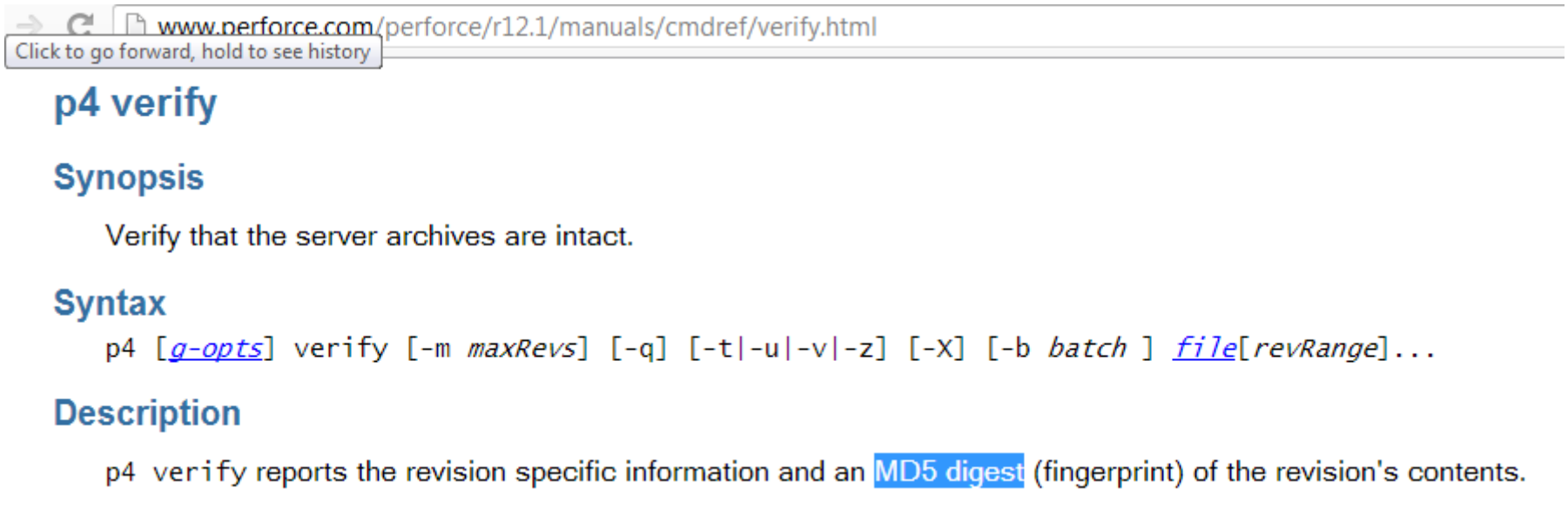
http://www.openstack.org/software/openstack-storage/

-› **MD5**

Example: Perforce

major commercial VCS

used in 5500+ orgs, including Google



reported to take several hours on large projects…

Why do people still use/recommend MD5 if it is cracked since 1996?

7

1

It's still commonly recommended way of hashing passwords, even if it's insecurity had been proven in 1996

> Therefore we suggest that in the future MD5 should no longer be implemented in applications like signa- ture schemes, where a collision-resistant hash func- tion is required. According to our present knowl- edge, the best recommendations for alternatives to MD5 are SHA-1 and RIPEMD-160.

(The Status of MD5 After a Recent Attack, CryptoBytes, RSA Laboratories, VOLUME 2, NUMBER 2 — SUMMER 1996)

MD5 2.5 times as fast as SHA-256

SHA-3 not much faster…

Need #1

at least as fast as MD5

Need more than a (fast) hash

Bring what users **need**, in a way that users **understand**

Need #2

specify “extra features”

but **avoid bloat** of design/specs

Need #3

usage-oriented specs

focus on users' needs and expectations

do it with the **right language**

Mission summary

aggressively optimize the design

identify and specify extra features

provide ready-to-use code and tools

present it in the most convincing way

Tiger team

JP Aumasson (@aumasson)

Samuel Neves (@sevenps)

Zooko Wilcox-O'Hearn (@zooko)

Chris Winnerlein (@codesinchaos)

BLAKE2

Faster than MD5 on 64-bit Intel

32% less RAM used than BLAKE

No-overhead support of

Parallel hashing (multicore, SIMD)

Tree mode (updatable, incremental)

Prefix-key, salt, personalization

BLAKE2b

for 64-bit CPUs, 1-64 byte digests
12 rounds, based on BLAKE-512

BLAKE2s

for 32-bit CPUs, 1-32 byte digests
10 rounds, based on BLAKE-256

G function: simpler, faster

$$
\begin{aligned}
a &\leftarrow a + b + m_{\sigma_r(2i)} \\
d &\leftarrow (d \oplus a) \ggg 32 \\
c &\leftarrow c + d \\
b &\leftarrow (b \oplus c) \ggg 24 \\
a &\leftarrow a + b + m_{\sigma_r(2i+1)} \\
d &\leftarrow (d \oplus a) \ggg 16 \\
c &\leftarrow c + d \\
b &\leftarrow (b \oplus c) \ggg 63
\end{aligned}
$$

G function: simpler, faster

$$
\begin{aligned}
a &\leftarrow a + b + m_{\sigma_r(2i)} \\
d &\leftarrow (d \oplus a) \ggg 32 \\
c &\leftarrow c + d \\
b &\leftarrow (b \oplus c) \ggg 24 \\
a &\leftarrow a + b + m_{\sigma_r(2i+1)} \\
d &\leftarrow (d \oplus a) \ggg 16 \\
c &\leftarrow c + d \\
b &\leftarrow (b \oplus c) \ggg 63
\end{aligned}
$$

no constants
-16% ops

saves 64 RAM bytes
in B2s (128 in B2b)

G function: simpler, faster

$$
\begin{aligned}
a &\leftarrow a + b + m_{\sigma_r(2i)} \\
d &\leftarrow (d \oplus a) \ggg 32 \\
c &\leftarrow c + d \\
b &\leftarrow (b \oplus c) \ggg 24 \\
a &\leftarrow a + b + m_{\sigma_r(2i+1)} \\
d &\leftarrow (d \oplus a) \ggg 16 \\
c &\leftarrow c + d \\
b &\leftarrow (b \oplus c) \ggg 63
\end{aligned}
$$

12% faster than 25 (pshufb…)

G function: simpler, faster

$$
\begin{aligned}
a &\leftarrow a + b + m_{\sigma_r(2i)} \\
d &\leftarrow (d \oplus a) \ggg 32 \\
c &\leftarrow c + d \\
b &\leftarrow (b \oplus c) \ggg 24 \\
a &\leftarrow a + b + m_{\sigma_r(2i+1)} \\
d &\leftarrow (d \oplus a) \ggg 16 \\
c &\leftarrow c + d \\
b &\leftarrow (b \oplus c) \ggg 63
\end{aligned}
$$

= doubling
+ shift
-> a bit faster…

Parameter block
xored to the IV

Offset	0	1	2	3
0	Digest length	Key length	Fanout	Depth
4	Leaf length			
8 12	Node offset			
16	Node depth	Inner length	RFU	
20 24 28	RFU			
32 ... 44	Salt			
48 ... 60	Personalization			

Little-endian

like MD5, unlike BLAKE or SHA-x

like Intel, AMD, ARM

Bytes, not bits

can save days of debugging (really)

Parallel hashing
×4 and ×8

BLAKE2bp(M) =

B2b(B2b(M1),B2b(M2),B2b(M3),B2b(M4))

BLAKE2sp(M) =

B2s(B2s(M1),B2s(M2),B2s(M3),…,B2s(M8))

Parallel hashing

Maximizes CPU usage

keeps cores and pipelines busy

(AVX2 will enable "2-in-1 " BLAKE2s)

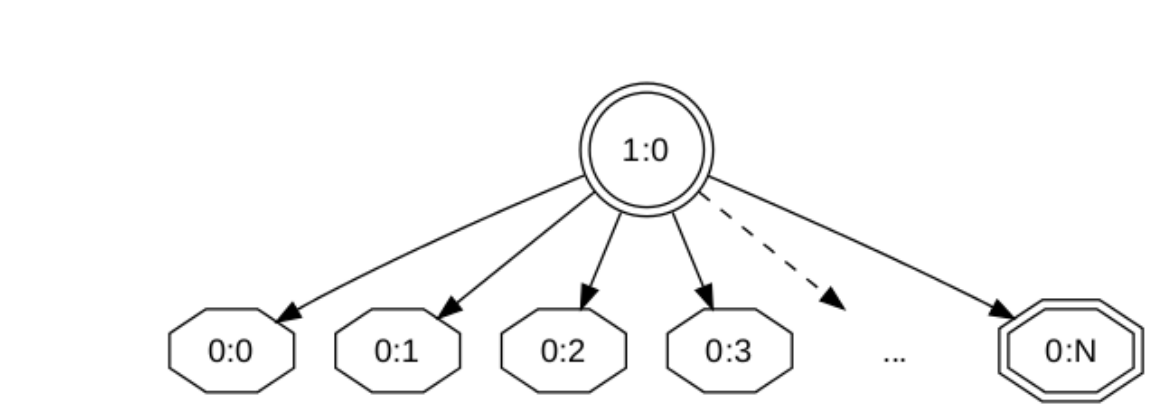
Minimizes computation overhead

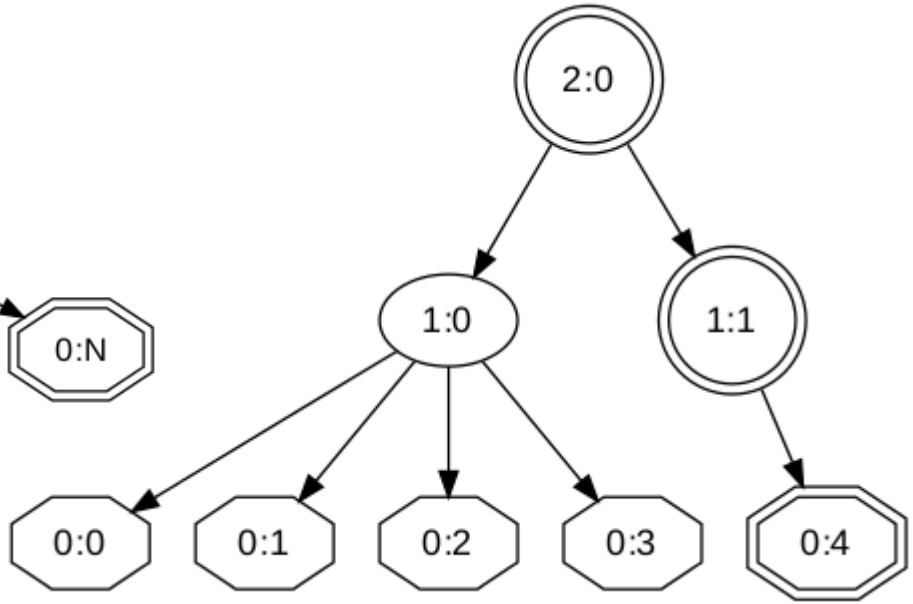
with 1 non-leaf node hashing short message

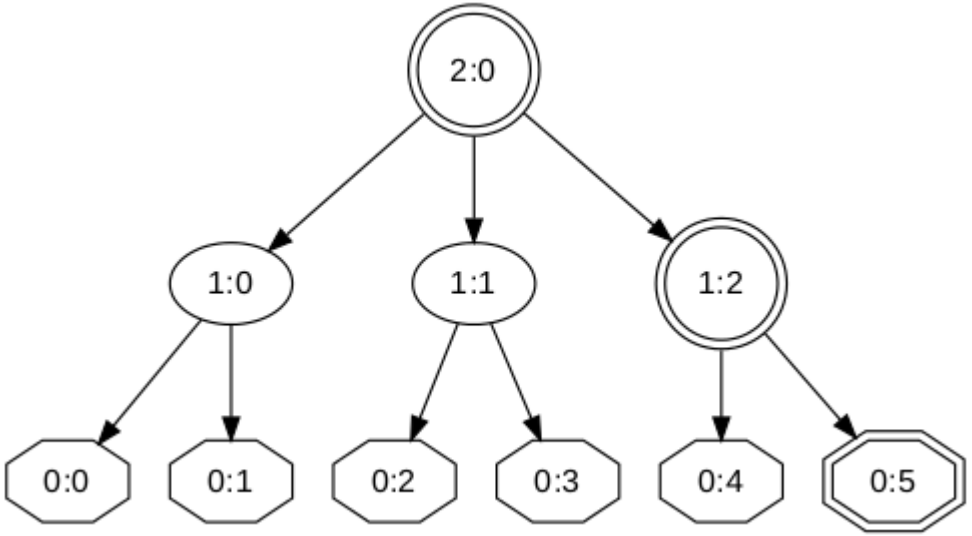
Supports streamed hashing, unknown-length messages

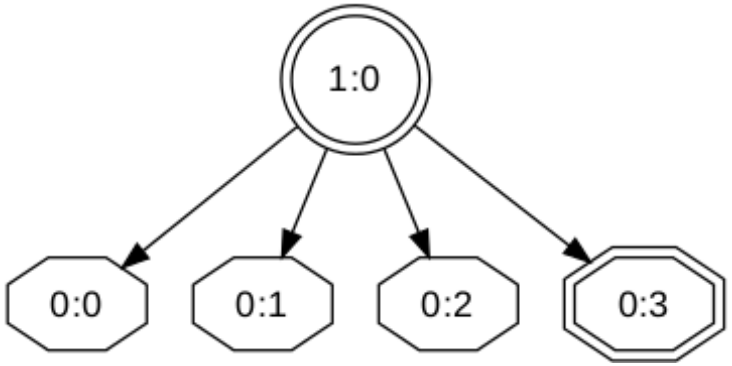
Tree hashing

Simple, yet comprehensive support









Tree hashing

"Sound tree hashing"
all nodes distinct, last node signaled, etc.

Supports unbounded fanout mode

Secure handling of "tree saturation"

Generic binary tree with 4KiB leaves

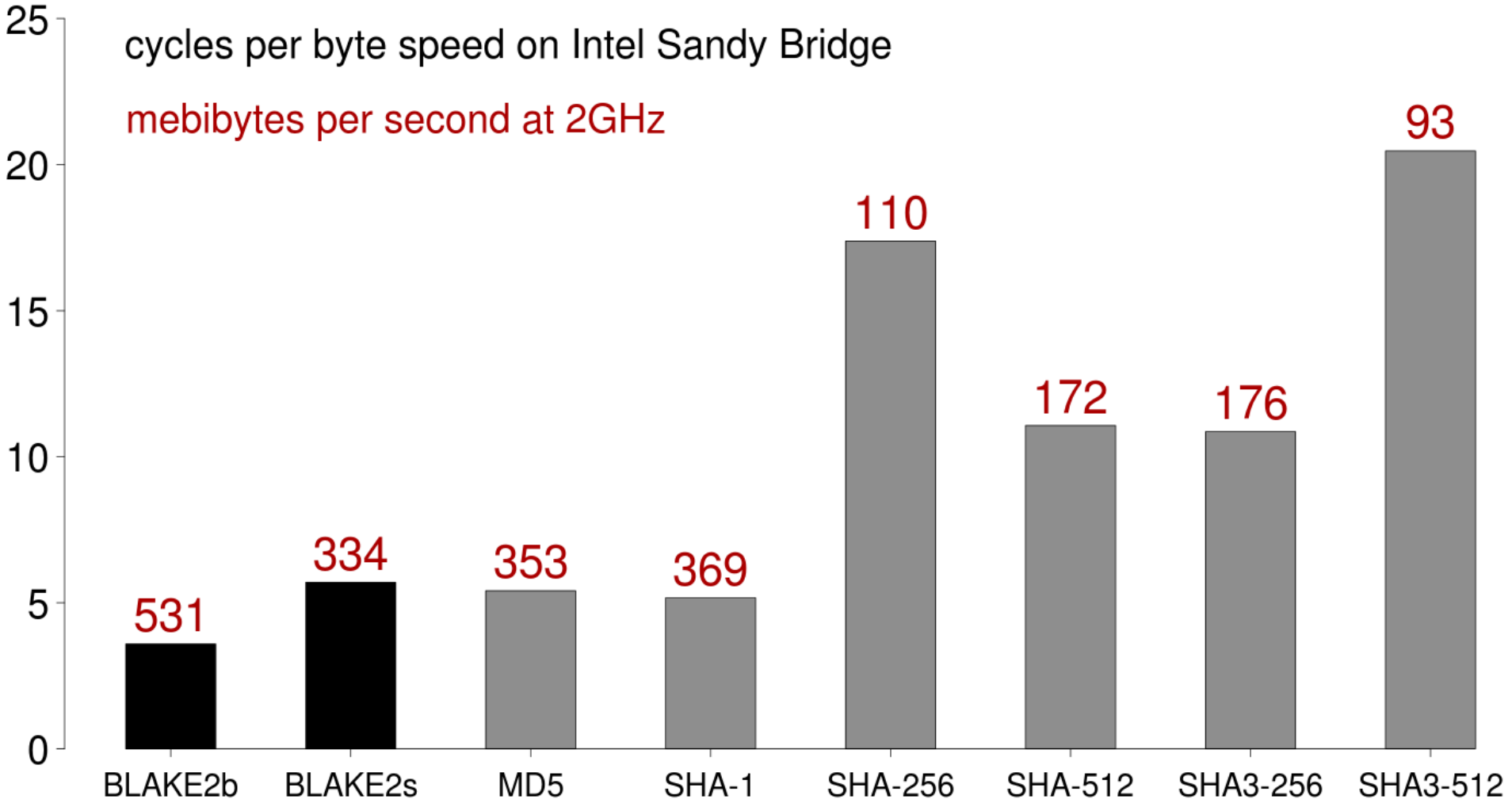
Security

High confidence, following previous analysis of BLAKE

Best hash attack on 2.5 rounds
e.g. collisions in 2^{112} instead of 2^{128}

No formal BLAKE › BLAKE2 reduction due to different compression functions

Performance



BLAKE2 designed to exploit

Multiple cores (4,8)

Instruction-level //ism

SIMD instruction sets
(including ARM's NEON, and future AVX2)

Cycles per byte

Microarchitecture	BLAKE2b			BLAKE2s		
	Long	1536	64	Long	1536	64
Sandy Bridge	3.59	3.96	8.56	5.70	5.74	6.63
Bulldozer	5.47	5.78	18.48	8.67	8.75	10.61

Mebibytes per second

Microarchitecture	BLAKE2b			BLAKE2s		
	Long	1536	64	Long	1536	64
Sandy Bridge (@2GHz)	531	482	223	335	332	288
Bulldozer (@3.6GHz)	628	330	103	220	330	180

Lesser speed-up on Bulldozer due to vector rotations…

Cycles per byte

5.35

Microarchitecture	BLAKE2b			BLAKE2s		
	Long	1536	64	Long	1536	64
Sandy Bridge	3.59	3.96	8.56	5.70	5.74	6.63
Bulldozer	5.47	5.78	18.48	8.67	8.75	10.61

Mebibytes per second

Microarchitecture	BLAKE2b			BLAKE2s		
	Long	1536	64	Long	1536	64
Sandy Bridge (@2GHz)	531	482	223	335	332	288
Bulldozer (@3.6GHz)	628	330	103	220	330	180

Lesser speed-up on Bulldozer due to vector rotations…

On Intel Sandy Bridge

BLAKE2b 59% faster than BLAKE-512

BLAKE2s 31% faster than BLAKE-256

on **long** messages

BLAKE2s faster than MD4

on ≤**64-byte** messages

On Intel Sandy Bridge

BLAKE2bp 3.51× as fast as BLAKE2b

BLAKE2sp 6.37× as fast as BLAKE2s

-› 700MB disk image hashed in ~300ms

Speed-up should converge to 4/8× if optimized

Low-end software

32% smaller than BLAKE

BLAKE2s requires 168 bytes of RAM

BLAKE2b requires 336 bytes of RAM

Hardware

speed-up only from round reduction

BLAKE2b 29% faster than BLAKE-512

BLAKE2s 25% faster than BLAKE-256

Our code package

C "ref" and "sse" of BLAKE2b/s/bp/sp
supports SSSE3, SSE4.1, AVX, XOP

C# of BLAKE2b for .NET integration

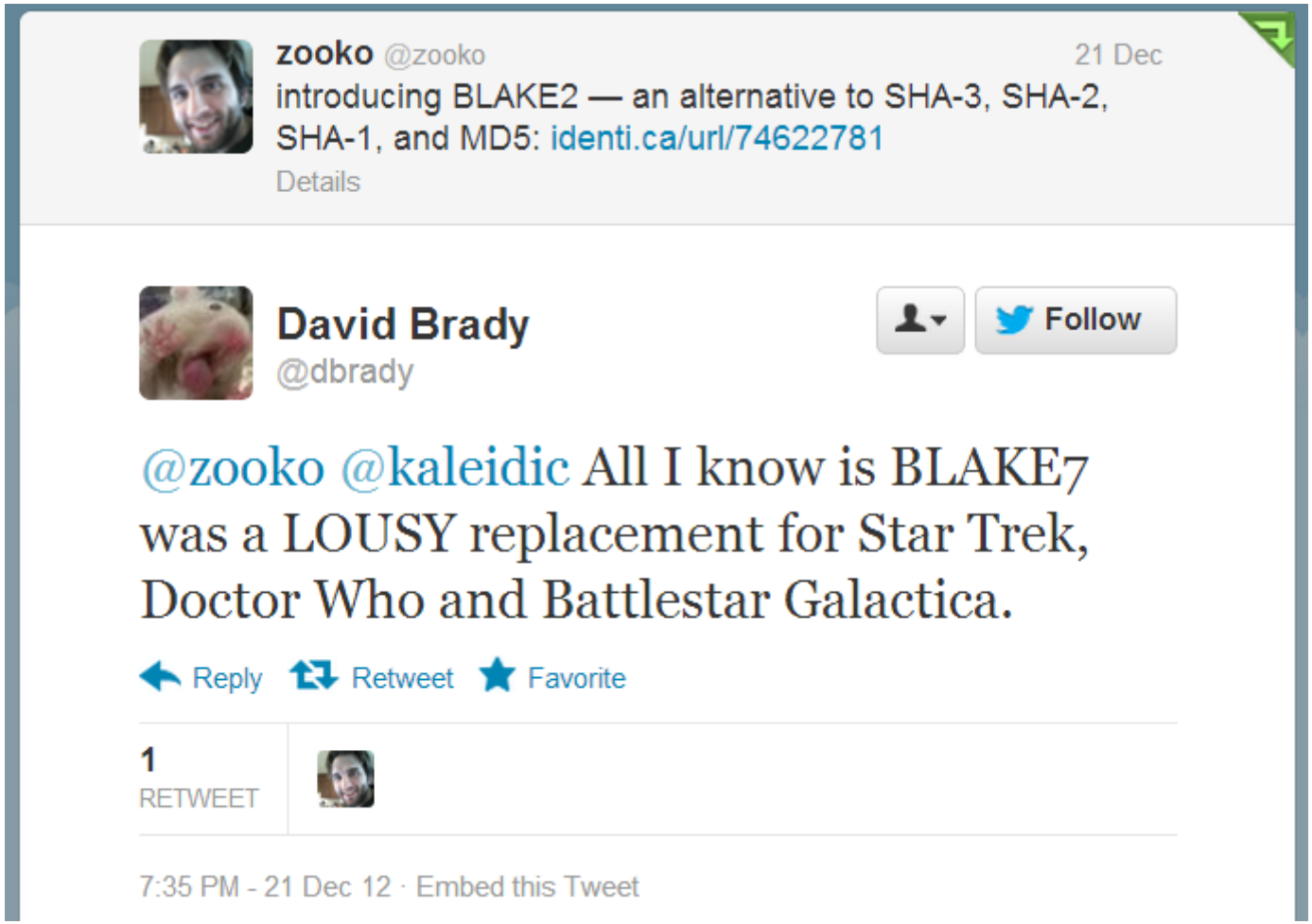
b2sum command-line tool

SUPERCOP-like benchmark tool

Released Dec 21st

Positive reception

Positive reception



zooko @zooko 21 Dec

introducing BLAKE2 — an alternative to SHA-3, SHA-2, SHA-1, and MD5: identi.ca/url/74622781

Details

David Brady

@dbrady

Follow

@zooko @kaleidic All I know is BLAKE7 was a LOUSY replacement for Star Trek, Doctor Who and Battlestar Galactica.

Reply Retweet Favorite

1 RETWEET

7:35 PM - 21 Dec 12 · Embed this Tweet

Positive reception

Slashdot

Channels

stories
submissions
popular
blog

ask slashdot
book reviews
games
idle
yro

cloud

BLAKE2 Claims Faster Hashing Than SHA-3, SHA-2 and MD5

Posted by **timothy** on Tuesday December 25, @01:18PM
from the loose-ends-may-appear-under-the-microscope dept.

hypnosec writes

> "BLAKE2 has been recently announced as a new alternative to the existing cryptographic hash algorithms MD5 and SHA-2/3. With applicability in cloud storage, software distribution, host-based intrusion detection, digital forensics and revision control tools, BLAKE2 performs a lot faster than the MD5 algorithm on Intel 32- and 64-bit systems. The developers of BLAKE2 insist that even though the algorithm is faster, there are no loose ends when it comes to security. BLAKE2 is an optimized version of the then SHA-3 finalist BLAKE."

Third-party code

BLAKE2b and BLAKE2s in Go, JavaScript
b2sum binaries for OS X, Linux, Windows
(Dmitry Chestnykh)

Libraries .a .so.* (Corey Richardson)
PHP wrapper (Craig Akimoto)
Python wrapper (Kwon-Han Bae)
Node.js bindings (Takashi Seki)
PPC Altivec C (@englabenny)

Now supported in John the Ripper

www.openwall.com/lists/john-users/2013/01/02/2

```
$ ./john testpw
Warning: detected hash type "blake2-512", but the string is also
recognized as "raw-sha512"
Use the "--format=raw-sha512" option to force loading these as that type
instead
Loaded 2 password hashes with no different salts (BLAKE2 512 [32/32])
guesses: 0  time: 0:00:00:02 0.00% (3)  c/s: 139432  trying: critas -
monniel
guesses: 0  time: 0:00:00:03 0.00% (3)  c/s: 151740  trying: 49382609 -
49184819
Session aborted

This means, the above hashes are now ambiguous.
Because there was no --format= option on the command line, john happens
to use the first format which identifies one of the hashes in the input
file as valid, and uses this format (here: blake2-512).
However, john also mentions which other hash formats would support at
least one of the hashes in the input file (here: raw-sha512).

If you want to force a specific format to be used, you'll have to use
the --format= option, either if you just want to suppress the checks
which other formats support the hashes given in the input file, or if
you want to make sure a certain format gets used:

$ ./john testpw --format=blake2-512
Loaded 2 password hashes with no different salts (BLAKE2 512 [32/32])
guesses: 0  time: 0:00:00:02 0.00% (3)  c/s: 154519  trying: meneste - saa
Session aborted
```

What's next?

SUPERCOP benchmarks

optimized NEON code

Tahoe-LAFS tests

and more...

Thank you

happy $3 \times 11 \times 61$