Comparative Performance Review of the SHA-3 Second-Round Candidates

Thomas Pornin

Cryptolog International

Second SHA-3 Candidate Conference

Outline

sphlib

sphlib is an open-source implementation of many hash functions:

- includes SHA-2 and the 14 SHA-3 second-round candidates (and many older hash functions)
- optimized, portable C code for “large systems”
- optimized, portable C code for “small embedded systems”
- optimized Java code
- all functions were implemented by the same developer with similar optimization techniques and efforts

sphlib: why ?

sphlib was initiated in 2007 with the following goals:

- to benchmark hash functions on many platforms, with an emphasis on embedded systems;
- to make a fair comparison between hash functions;
- to explore performance of Java implementations of hash functions.

On PC systems, **sphlib** implementations are not the fastest available, but measures on PC can be used to *validate* optimization strategies, and to explore some architecture effects.

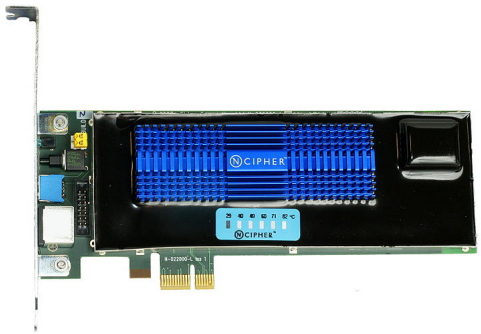
Comparison with eBASH

eBASH relies on externally provided implementations:

1. **eBASH** publishes results on some platforms;
2. implementers try to beat current records and submit new code;
3. loop to 1.

This development process appears to be slow on non-PC platforms, and has not yet created enough momentum on embedded systems. In the meantime, **sphlib** provides measures.
Also, **sphlib** has Java code.

Embedded systems



Embedded systems

Embedded systems which we consider here have the following characteristics:

- 32-bit registers
- "raw" RISC with no extra unit (no SSE2, often no FPU either): optimal for "portable" C programming
- small L1 cache RAM for instruction (no more than 16 kB)
- low operating frequency (less than 200 MHz)

Embedded systems are *specialized* and CPU-starved.

Embedded systems constraints

- cannot fully unroll all loops (small L1 cache): implies extra indirections
- 64-bit operations are slow
- no superscalar execution: parallelism is expensive
- some do not have 32-bit rotations

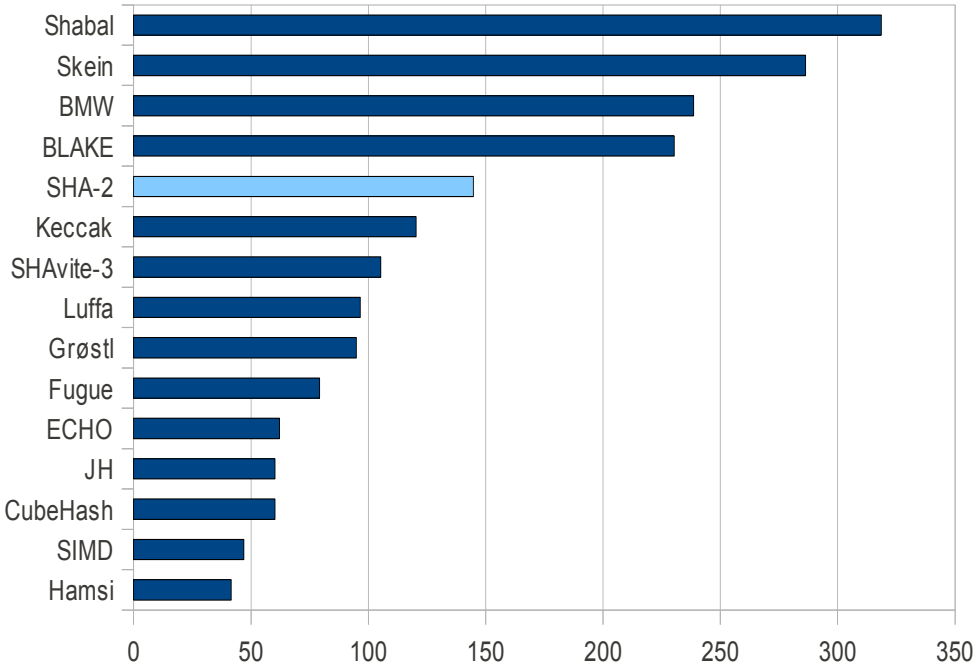
However, embedded systems often have plenty of registers. Endianness appears to be mostly irrelevant to performance.

Java constraints

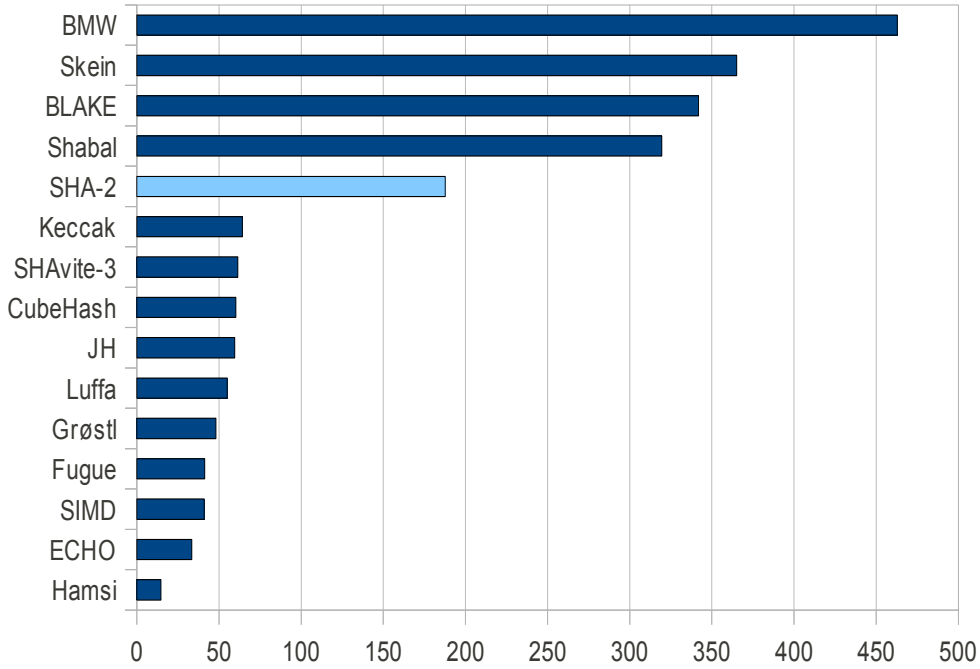
Virtual Machines (in particular Java) offer portability, ease of development and distribution, and safety against buggy or hostile code.

- no special operations, only generic arithmetic code
- code cannot be adjusted for register size
- the JIT compiler produces fat code $\longrightarrow$ severe L1 cache issues
- table accesses are more expensive
- the JIT compiler must work fast

x86-64: Intel Q6600, 64-bit, 2.4 GHz (MBytes/s)

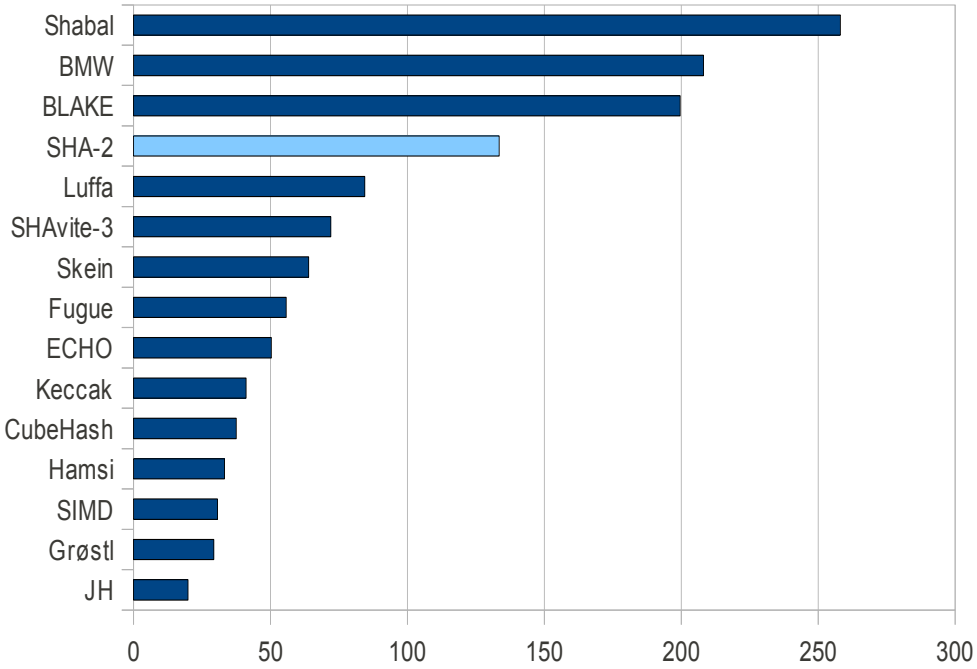


256-bit output

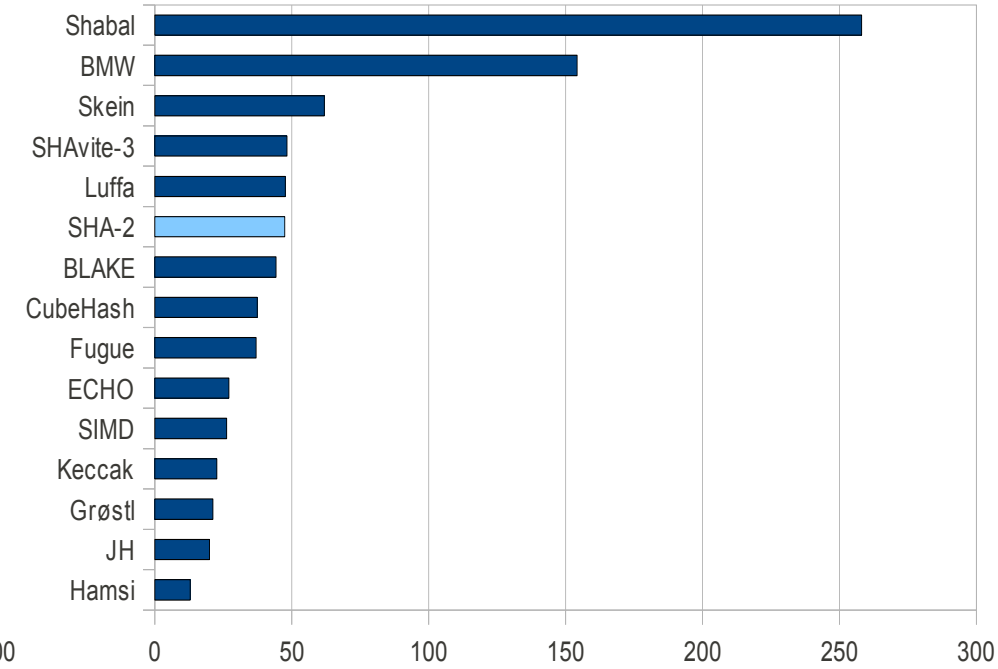


512-bit output

i386: Intel Q6600, 32-bit, 2.4 GHz (MBytes/s)

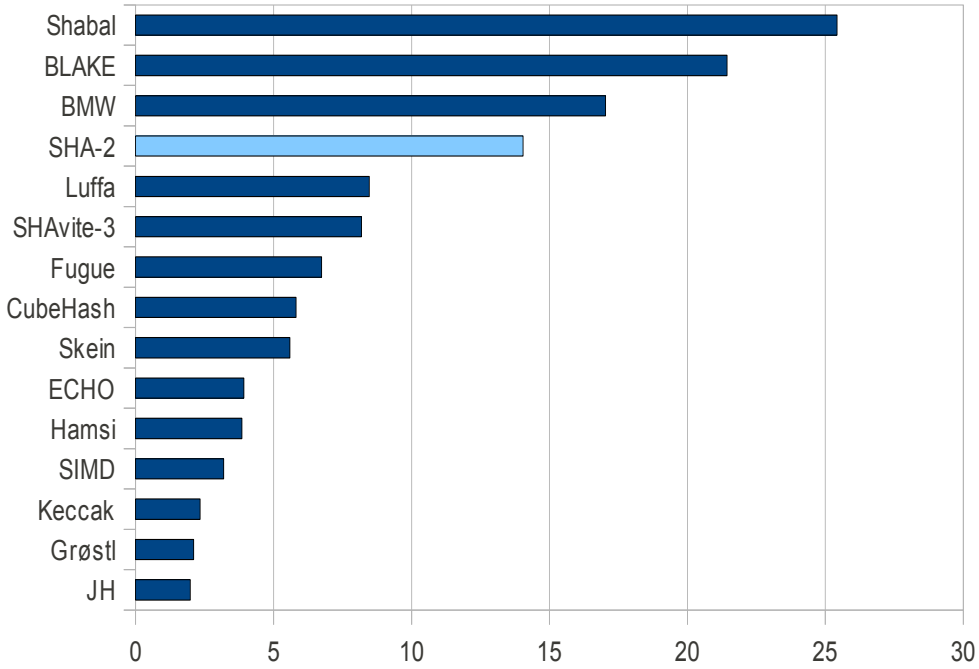


256-bit output

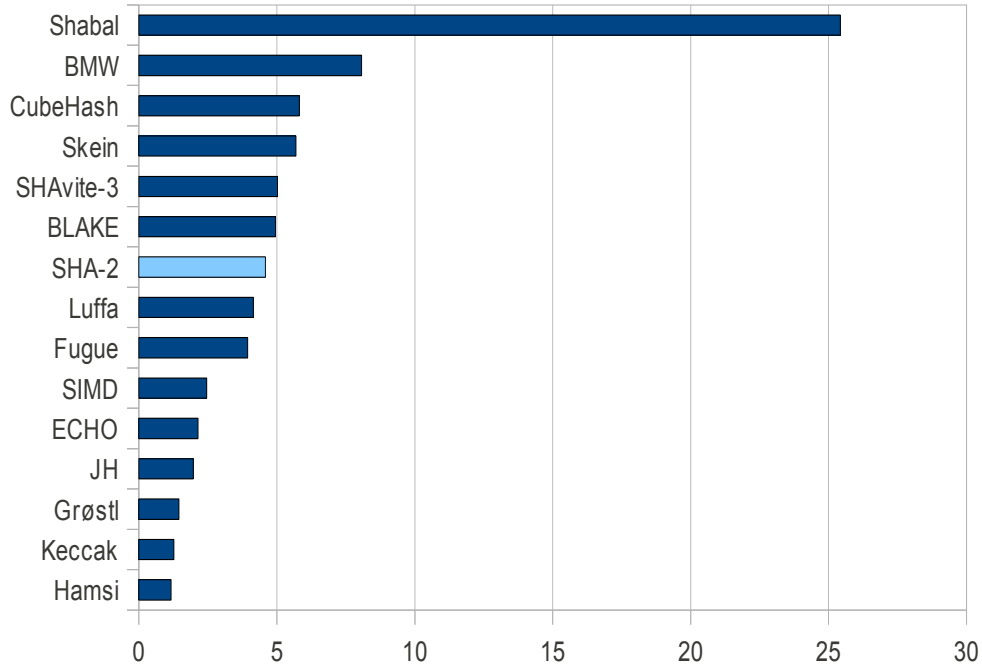


512-bit output

G3: PowerPC 750, 32-bit, 300 MHz (MBytes/s)

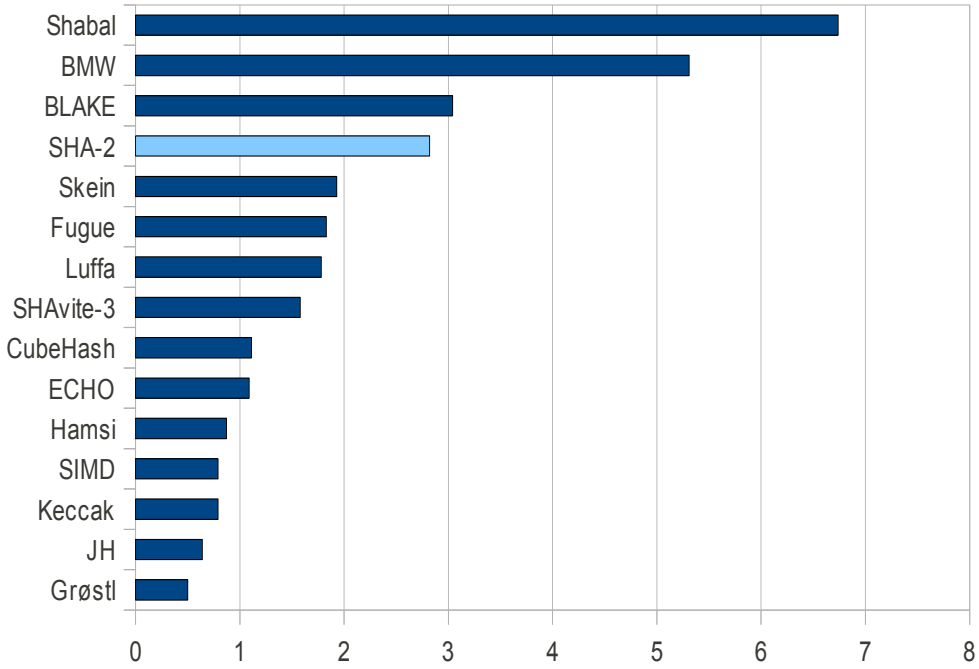


256-bit output

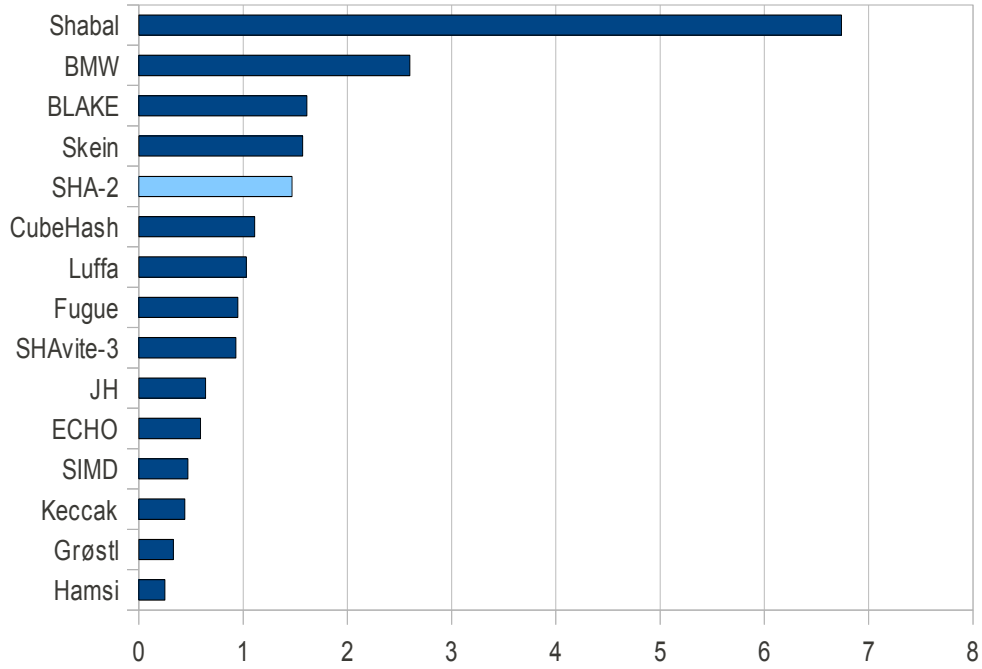


512-bit output

MIPS: BCM3302, 32-bit, 200 MHz (MBytes/s)

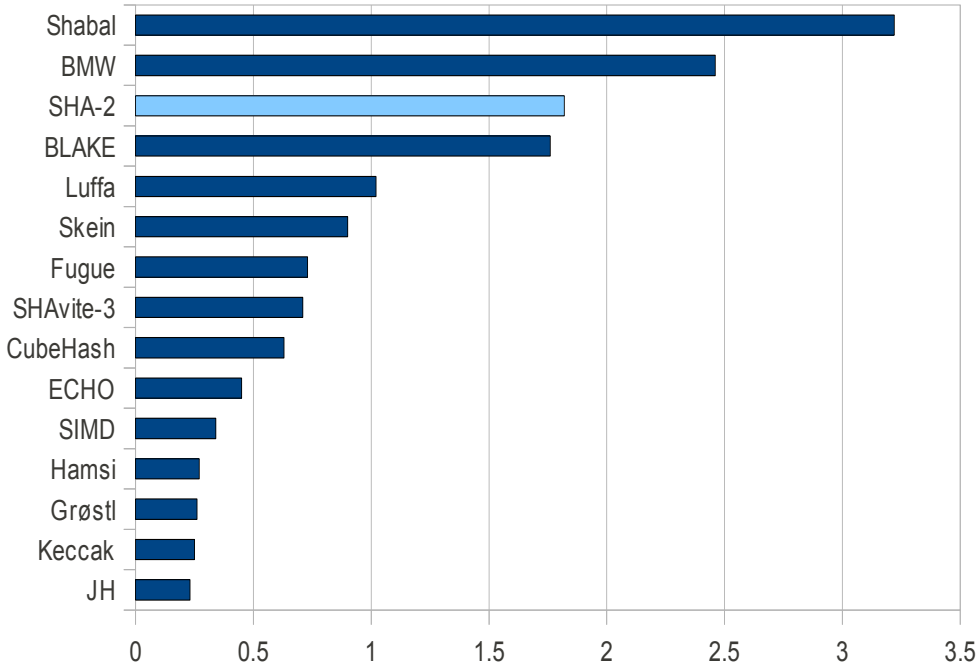


256-bit output

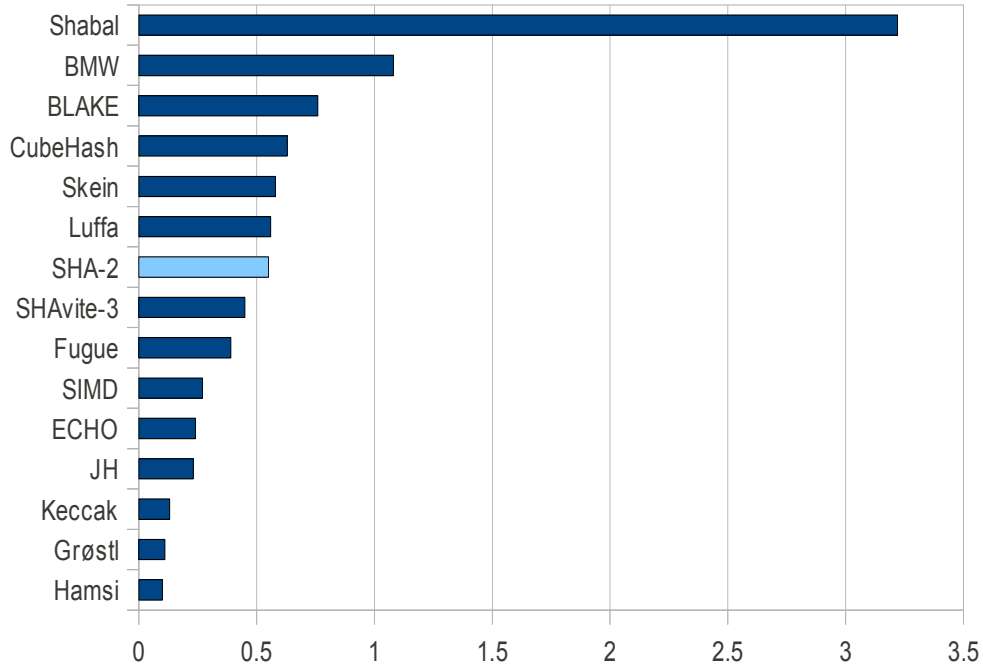


512-bit output

ARMv4: ARM920T, 32-bit, 75 MHz (MBytes/s)

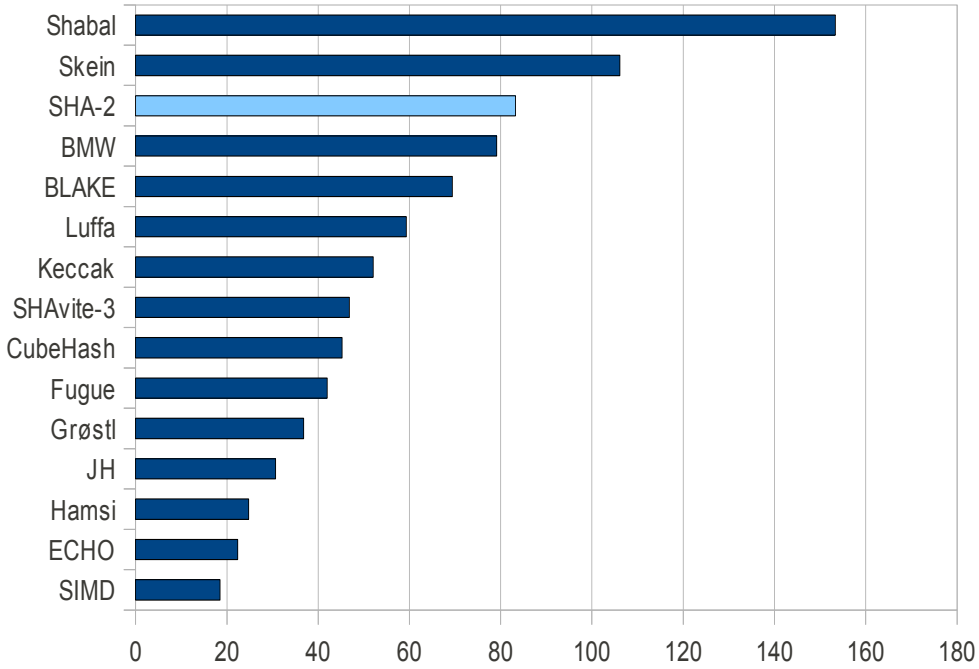


256-bit output

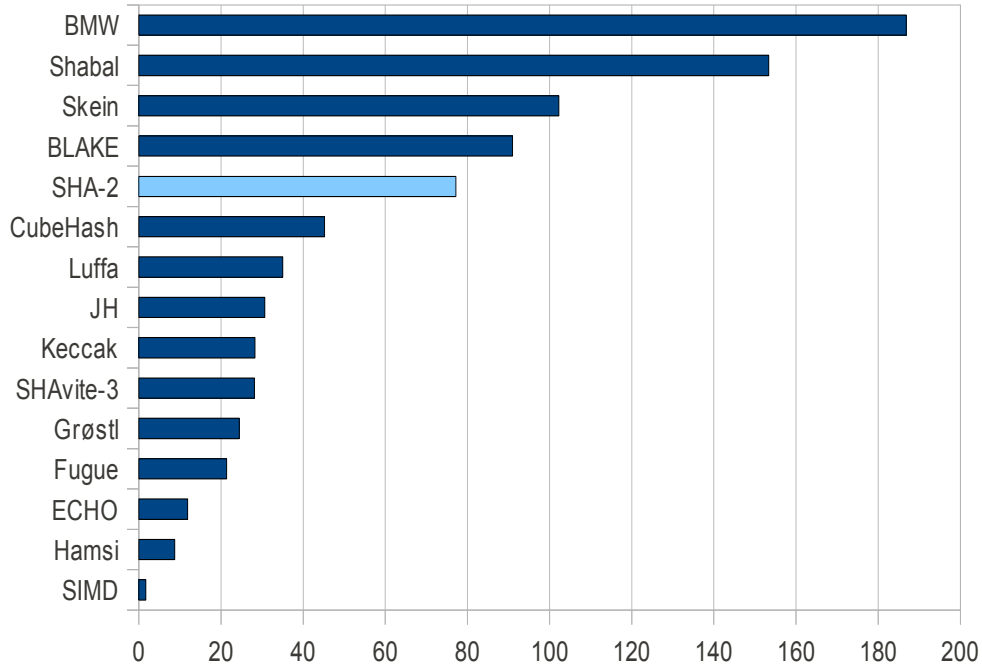


512-bit output

Java: Intel Q6600, 64-bit, 2.4 GHz (MBytes/s)

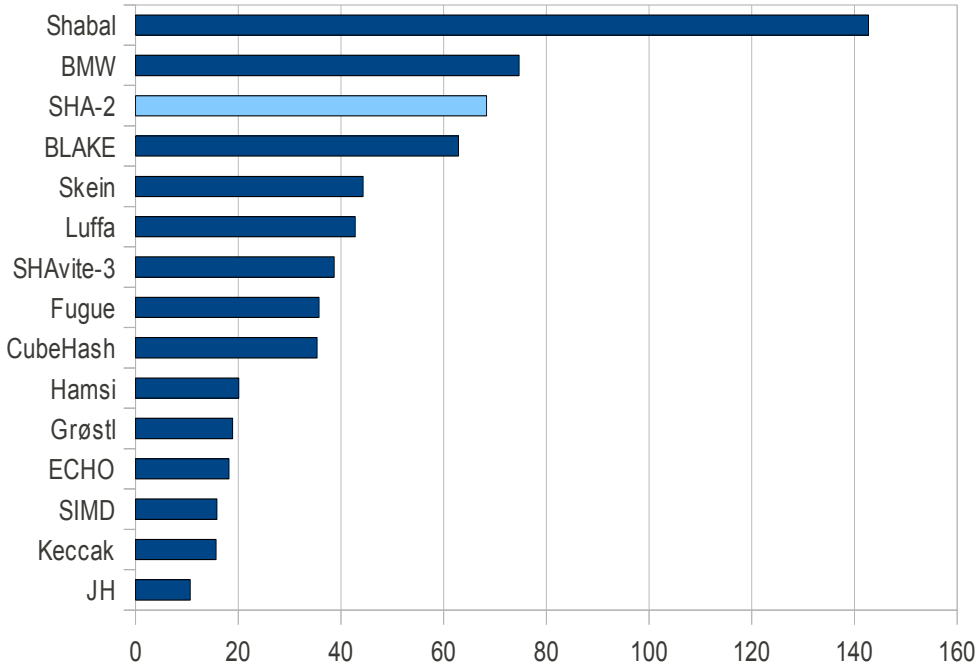


256-bit output

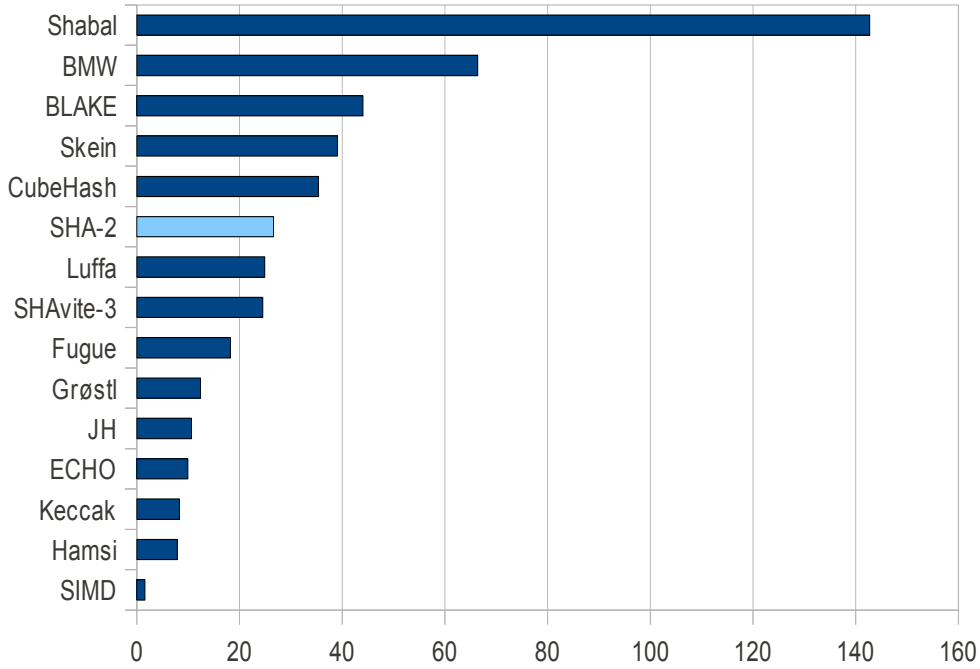


512-bit output

Java: Intel Q6600, 32-bit, 2.4 GHz (MBytes/s)



256-bit output



512-bit output

Conclusions

Hash function performance is important on embedded systems, much less so on large systems.

Most SHA-3 second round candidates (and SHA-2) appear to be optimized for large systems and/or 64-bit platforms.

CubeHash, JH and Shabal are *consistent* (performance does not depend on output size). Skein is mostly consistent.

Shabal was designed to run well on embedded systems.

Unexplored questions

Code footprint: often crucial in embedded systems. **sphlib** limits itself to the L1 cache size (8 kB on MIPS) but does not reduce code further.

Short messages: **sphlib** optimizes code for long messages, with buffers. Meaningful benchmarks are harder for short messages.

Script languages (e.g. Javascript).

`http://www.saphir2.com/sphlib/`