# SHA-3 interoperability

Daniel J. Bernstein *

Department of Computer Science (MC 152)
The University of Illinois at Chicago
Chicago, IL 60607–7053
djb@cr.yp.to

## 1 Introduction: You thought *software* upgrades were hard?

"The 8155 continues Hifn's successful and proven line of intelligent packet processing security processors," says a coprocessor data sheet issued years ago. "Supports all major security & compression protocols. IPsec performance: AES/SHA1—2.8Gbps."

"Oracle's Sun Crypto Accelerator 6000 PCI Express (PCIe) Adapter improves both network security and bottom lines without adding undue complexity and without draining system performance, resources, or budget," says another datasheet. "Hash functions: SHA1 and MD5."

"Maxim enters growing RFID market with 13.56 MHz Secure RF Keys," says a press release issued a few months ago. "The MAX66000 and MAX66100 devices employ the Secure Hash Algorithm (SHA-1), a proven technology for protecting a system's critical data without using expensive encryption techniques or an untested, proprietary protocol."

All of these devices have SHA-1 burned into ASICs. These ASICs aren't interoperable with protocols that use SHA-256. They don't know how to compute SHA-256. For IPsec and SSL there is a nearby CPU that can be programmed to understand SHA-256—but the CPU consumes much more power and much more time than the ASIC, limiting the volume of network traffic. The ASIC exists precisely because the CPU *can't* handle the expected load at an acceptable cost.

How are the users of these devices supposed to upgrade from a protocol using SHA-1 to a protocol using SHA-256? They have to buy and install new hardware. If they don't take action—or if they merely do something as easy as a software upgrade—then the new protocol doesn't work. Cryptographers and government agencies say that moving away from SHA-1 is prudent; but these users, supported by developers who haven't built SHA-256 hardware yet, kick and scream and say that SHA-1 is working just fine for them. Users who *want* protocols using SHA-256 find those protocols difficult to deploy.

## 2 Planning ahead for SHA-1/SHA-256 interoperability

SHA-256 was standardized a decade ago. It's as old as AES, and older than any of the hardware implementations mentioned above. Why didn't the hardware designers build SHA-256 into their ASICs?

* Permanent ID of this document: 087f299fbb2b92a1644d80537663c096. Date of this document: 2010.09.15. This work was supported by the National Science Foundation under grant 0716498.

The obvious answer is hardware cost. For example, Helion sells a "Fast Hash Core" that can support both SHA-1 and SHA-256, but SHA-1+SHA-256 costs many more gates than SHA-1 alone. Here's what Helion's advertising says for a 130nm ASIC:

- 12700 gate equivalents (GEs) for a 100MHz SHA-1 unit hashing at 624Mbps.
- 13200 GEs for a 200MHz SHA-1 unit hashing at 1248Mbps.
- 15700 GEs for a 350MHz SHA-1 unit hashing at 2185Mbps.
- 20100 GEs for a 100MHz combined unit that handles SHA-1 at 624Mbps and SHA-256 at 775Mbps.
- 20900 GEs for a 200MHz combined unit that handles SHA-1 at 1248Mbps and SHA-256 at 1551Mbps.
- 24600 GEs for a 300MHz combined unit that handles SHA-1 at 1872Mbps (actually, the sheet says 2185Mbps, but this was clearly a copy-and-paste error) and SHA-256 at 2327Mbps.

A typical cryptographic coprocessor has several competing demands for chip area and can't casually expand its SHA unit from 12700 GEs to 20100 GEs, or 15700 GEs to 24600 GEs, or 31400 GEs (if two parallel 300MHz units are needed to handle the hashing load) to 49200 GEs. It's completely unsurprising to see hardware designers opting for SHA-1 alone, limiting the security provided to their users but at the same time reducing cost.

Similar comments apply to RFIDs, on a smaller and slower scale. The lowest area reported for SHA-1 (O'Neill, RFIDSec08), again at 130nm, uses 5527 GEs and takes 344 cycles per block while consuming 2.32 microwatts at 100kHz. I haven't found any reports of SHA-256 implementations under 10000 GEs, never mind SHA-1+SHA-256.

## 3 History repeating itself

The upgrade from SHA-1 to SHA-256 is only the beginning. As the years pass, more and more cryptographers will realize that the $2^{128}$ security level of SHA-256 is not as comfortable as it once sounded, and will advocate an upgrade to a higher security level. The government standards are ready for this—they include SHA-512, at a very comfortable long-term $2^{256}$ security level—but typical SHA-256 hardware is *not* interoperable with SHA-512.

Why is SHA-256 hardware not interoperable with SHA-512? The obvious answer—as in the case of SHA-1 vs. SHA-256—is hardware cost. Look at how large Helion's SHA-512 units are:

- 38800 GEs for a 100MHz SHA-512 unit hashing at 1248Mbps.
- 41500 GEs for a 200MHz SHA-512 unit hashing at 2497Mbps.

A hardware designer who has been convinced to spend area on SHA-1+SHA-256 is still unlikely to spend area on SHA-1+SHA-256+SHA-512. Users of typical SHA-256 hardware will not have an easy way to upgrade to a protocol using SHA-512, and will object to the upgrade, the same way that SHA-1 users object to an upgrade today. Perhaps cryptographers' better judgment will win in the end, but only after incurring massive hardware-upgrade costs.

Is that the complete story? What about SHA-3?

If SHA-3 is going to be successful then it will have to jump into this picture somewhere. There are two obvious possibilities:

- Harder case: Upgrade from SHA-1 to SHA-256, and later from SHA-256 to SHA-3-256.
- Easier case: Upgrade from SHA-1 directly to SHA-3-256.

"Easier" is somewhat misleading terminology here. An upgrade from SHA-1 to SHA-3-256 should really be called the "hard case," and an upgrade from SHA-1 to SHA-256 to SHA-3-256 should be called the "twice-as-hard case."

Either way, we have to expect that SHA-3-256 will not be satisfactory in the long term—that's why there's also SHA-3-512! So we'll eventually have *another* painful upgrade, an upgrade from SHA-3-256 to SHA-3-512.

## 4 Planning ahead for SHA-3 interoperability

There's an obvious way to improve this situation. SHA-3-256 and SHA-3-512 will be standardized at the same time. Why don't the hardware designers build ASICs that are interoperable with *both* SHA-3-256 and SHA-3-512?

The virtues of interoperability are clear. In the long term, when cryptographers and government agencies decide that moving to SHA-3-512 is prudent, they won't have to fight against hardware that supports only SHA-3-256. Users with SHA-3-256 hardware won't have to suffer through a painful hardware upgrade. Users who want the long-term security of SHA-3-512 won't be limited by interoperability with SHA-3-256 hardware.

The obvious problem is again hardware cost: supporting SHA-3-256+SHA-3-512 costs much more hardware area than simply supporting SHA-3-256. But there are some SHA-3 candidates for which this problem disappears! Some SHA-3 candidates were designed so that SHA-3-256+SHA-3-512 would naturally fit together into the same hardware:

- CubeHash sets some IV bits to distinguish CubeHash16/32–256 from CubeHash16/32–512. Subsequent processing is identical. The output is truncated to 256 or 512 bits.
- ECHO sets some IV bits to distinguish ECHO-256 from ECHO-512. Subsequent processing has a different message-block size, and a different number of rounds, but the state size is identical and the rounds are identical, so a combined implementation should not cost much extra space. The output is truncated to 256 or 512 bits.
- JH sets some IV bits to distinguish JH-256 from JH-512. Subsequent processing is identical. The output is truncated (on the left) to 256 or 512 bits.
- Keccak uses the same IV for 256-bit Keccak[] and 512-bit Keccak[]. (I am ignoring other options such as 256-bit Keccak-c512.) Subsequent processing is identical. The output is truncated to 256 or 512 bits.
- Shabal sets some IV bits to distinguish Shabal-256 from Shabal-512. Subsequent processing is identical. The output is truncated (on the left) to 256 or 512 bits.
- Skein sets some IV bits to distinguish 256-bit Skein-512-256 from 512-bit Skein-512-512. (I am ignoring the lower-area Skein-256-256 proposal, which appears to be harder to

combine with Skein-512-512.) Subsequent processing is identical. The output is truncated to 256 or 512 bits.

With any of these choices for SHA-3, it would be perfectly reasonable for the SHA-3 standard to encourage SHA-3-256 hardware implementations to also support SHA-3-512, allowing a choice at run time between SHA-3-256 and SHA-3-512. It's easy to imagine NIST's "SHS Validation List" making a special note for these flexible implementations.

## 5 What is the cost of SHA-3 interoperability?

The above list contains the SHA-3 candidates that allow a unified SHA-3-256+SHA-3-512 implementation to be about as small as a SHA-3-256 implementation. Designers of other candidates can reasonably respond that this isn't a helpful feature when the SHA-3-256 implementation is too big to begin with!

Here is an illustrative example. As far as I know, the smallest area reported for ECHO-256 is 60000 GEs, estimated (post-synthesis) to run at 204Mbps at 90nm. ECHO-512 would fit in essentially the same space, running at about 100Mbps. BLAKE was not designed for similar unification—but it appears that *both* 256-bit BLAKE-32 and 512-bit BLAKE-64 would fit in significantly less area than ECHO while providing higher performance:

- BLAKE-32 is estimated (post-layout) to run at 125Mbps—which might not sound faster than 204Mbps, but this was much older 180nm technology!—using 13500 GEs.
- No complete ASIC implementations of BLAKE-64 have been reported, but one can extrapolate from compression-function implementations to guess that BLAKE-64 will achieve similar throughput to BLAKE-32 with about twice the area. The two together should fit into 40000 GEs.

Of course, larger implementations of ECHO can achieve higher speeds, but the same is true for BLAKE. If it is reasonable to ask hardware developers to implement ECHO-256+ECHO-512, surely it is just as reasonable to ask hardware developers to implement BLAKE-32+BLAKE-64. Both approaches eliminate the need for a subsequent hardware upgrade, and the BLAKE-32+BLAKE-64 approach uses considerably fewer gates.

There haven't been any systematic benchmarks of combined SHA-3-256+SHA-3-512 hardware units—a curious omission in a world where one can buy combined SHA-1+SHA-256 hardware units. Recently Gaj et al. benchmarked all the SHA-3-512 proposals on FPGAs, but most ASIC benchmarks have been limited to SHA-3-256. Existing reports nevertheless make clear that at least two SHA-3 candidates can fit SHA-3-256+SHA-3-512 into similar area to SHA-1+SHA-256:

- CubeHash: 22968 GEs, 1290Mbps at 130nm (post-layout, Guo et al.); or 7630 GEs, 25Mbps at 130nm (post-synthesis, Bernet et al., assuming approximately 16× speedup from CubeHash8/1 to CubeHash16/32—note that these fit in the same area).
- Shabal: 23320 GEs, 310Mbps at 130nm (post-synthesis, Bernet et al.).

I don't think that this list, CubeHash and Shabal, is complete. When I hear that the lowest-area implementation reported for CubeHash is 3× smaller than the lowest-area implementation reported for any other SHA-3-512 candidate (never mind SHA-3-256+SHA-3-512), my

presumption is that the implementors need to do more work. I prioritized smallness when I designed CubeHash, and I think I did a good job, but I also think that 3× is an exaggeration.

More data is available regarding price-performance ratio. Guo et al. identified five SHA-3 candidates for which SHA-3-256 provides better price-performance ratio than SHA-256:

- CubeHash16/32: 38184 GEs, 4624Mbps. CubeHash16/32–512 runs at the same speed as CubeHash16/32–256 in the same area, 38184 GEs. A unified implementation would run both CubeHash16/32–256 and CubeHash16/32–512 with better price-performance ratio than SHA-256.
- Grøstl–256: 110108 GEs, 9606Mbps. Grøstl–512 runs about 1.4× faster than Grøstl–256 in about twice the area, according to Gaj et al. A unified implementation would run both Grøstl–256 and Grøstl–512 with worse price-performance ratio than SHA-256.
- Hamsi–256: 29941 GEs, 3571Mbps. Hamsi–512 runs about as quickly as Hamsi–256 in about twice the area, 59882 GEs. A unified implementation would run both Hamsi–256 and Hamsi–512 with worse price-performance ratio than SHA-256.
- Keccak[]–256: 47434 GEs, 15457Mbps. Keccak[]–512 runs at the same speed as Keccak[]–256 in the same area, 47434 GEs. A unified implementation would run both Keccak[]–256 and Keccak[]–512 with better price-performance ratio than SHA-256.
- Luffa–256: 37942 GEs, 13943Mbps. Luffa–512 runs about as quickly as Luffa–256 in about 1.67× as much area, 63363 GEs. It is not clear exactly how big a combined implementation would be, but the answer is at most 2.67×, so it is clear that a unified implementation would run both Luffa–256 and Luffa–512 with better price-performance ratio than SHA-256.

All other SHA-3-256 candidates provide worse price-performance ratio than SHA-256. The bottom line is that CubeHash, Keccak, and Luffa are the only SHA-3 candidates for which a combined SHA-3-256+SHA-3-512 unit would run SHA-3-256 with better price-performance ratio than SHA-256—i.e., for which SHA-3-256 would be faster than SHA-256 on the same amount of hardware. These units consume 38184 GEs for CubeHash, 47434 GEs for Keccak, and at least 63363 GEs for Luffa.

A thorough comparison requires more information. A hardware designer normally has a particular performance target, say $M$ megabits per second, and would like to know how much hardware area has to be spent to meet this target. The graph of area as a function of $M$ becomes a line as $M \to \infty$, with slope equal to price-performance ratio, but for small $M$ the graph is above—often very far above—this line. Beating SHA-256 in price-performance ratio implies beating SHA-256 in area for *large* $M$ but does not imply beating SHA-256 in area for *small* $M$.

The CubeHash example nevertheless suggests the intriguing possibility that—for some SHA-3 candidates!—hardware implementors will be able to provide SHA-3-256+SHA-3-512 at similar cost to providing *just* SHA-256, much lower cost than SHA-256+SHA-512. If users are given SHA-3-512 without having to pay for it then they will easily be convinced that 512-bit support is a useful long-term feature, a reason to support upgrading protocols from SHA-1 to SHA-3 rather than to SHA-2.