CubeHash parameter tweak: 16 times faster

Daniel J. Bernstein *

Department of Computer Science
University of Illinois at Chicago
Chicago, IL 60607–7045
cubehash@box.cr.yp.to

Summary. CubeHash is a parametrized family of hash functions CubeHashr/b. The original CubeHash submission proposed CubeHash8/1 as SHA-3.

This year NIST issued some clarifications of how speed and security will be evaluated in the SHA-3 competition. The clarifications show that CubeHash8/1 is much more conservative than necessary. In response, this document proposes

- CubeHash16/32–224 for SHA-3–224,
- CubeHash16/32–256 for SHA-3–256,
- CubeHash16/32–384 for SHA-3–384–normal,
- CubeHash16/32–512 for SHA-3–512–normal,
- CubeHash16/1–384 for SHA-3–384–formal, and
- CubeHash16/1–512 for SHA-3–512–formal.

For all real-world cryptographic applications, the "formal" versions here can be ignored, and this document amounts to a proposal of CubeHash16/32 as SHA-3. CubeHash16/32 is approximately 16 times faster than CubeHash8/1, easily catching up to both SHA-256 and SHA-512 on the reference platform. Despite this speed, CubeHash16/32 is a conservative proposal with a comfortable security margin. The best attacks known against CubeHash16/32 are the attacks explained in the original CubeHash submission documents.

CubeHash parameter review. NIST's SHA-3 submission requirements stated that a submission "may include a tunable security parameter, such as the number of rounds, which would allow the selection of a range of possible security/performance tradeoffs. ... The tunable parameter may be used to produce weakened versions of the submitted algorithm for analysis, and permit NIST to select a different security/performance tradeoff than originally specified by the submitter, in light of discovered attacks or other analysis, and in light of the alternative algorithms that are available."

CubeHash is parametrized by a pair of positive integers (r, b) with $b \leq 128$. CubeHashr/b applies an r-round transformation after each b-byte message block.

* The author was supported by the National Science Foundation under grant ITR–0716498. Date of this document: 2009.07.15.

The time taken by CubeHashr/b is approximately proportional to the quotient r/b.

Extremely fast versions of CubeHashr/b, with small r and large b, are easy to break in various ways, as explained in the original CubeHash submission. However, existing security analyses confirm the intuitive idea that doubling r (and thus doubling the CubeHash time) makes attacks much more difficult; existing security analyses also confirm the intuitive idea that chopping b in half (and thus doubling the CubeHash time) makes attacks much more difficult. A few such doublings produce unbroken versions of CubeHash, and one or two additional doublings produce versions of CubeHash with a comfortable security margin.

The original CubeHash submission recommended CubeHash8/1, and stated the following justification for the recommendation:

> For most applications of hash functions, speed simply doesn't matter. High-volume network protection with HMAC is sometimes cited as an exception, but anyone who really cares about speed shouldn't be using HMAC anyway; other MACs are faster and inspire more confidence.
>
> What about the occasional applications where hashing speed *does* matter? If, after third-party cryptanalysis, the community is convinced that much faster CubeHashr/b choices are perfectly safe, then I expect those choices to be considered in speed-oriented applications.

Speed criteria. NIST organized the First SHA-3 Candidate Conference in Leuven in February 2009, including many presentations of SHA-3 candidates and several official NIST presentations. The transcript of NIST's "Security-Performance Tradeoff" presentation includes the following statement:

> It is quite natural that SHA-2 forms a very important benchmark. If we adjust tunable parameters to run an algorithm as fast as SHA-256, SHA-512, on these two specified platforms, IA-32 and AMD64, we ask is the algorithm secure if we tune the parameters to run it as fast as SHA-2? If it is not, I'm afraid that it hurts its chances to be advanced to the second round. This is our initial thinking on how to handle tunable parameters to have an impact on this next selection. Beyond that, we are not very much paranoid about performance in other platforms. But this is our sort of threshold. This we mainly focus on.

The subsequent question-and-answer session included another statement from NIST along the same lines: "If we look at that and whatever we can tune, if it looks to us like it's slow compared to SHA-2, it's going to be very hard to see how we would ever sell it to the public."

CubeHash speed review. Here are the eBASH (`supercop-20090702`) long-message timings of SHA-256, SHA-512, and CubeHash16/32 on an Intel Core 2 Duo 6f6 (`katana`) and an Intel Core 2 Duo E8400 1067a (`brick`):

- 11.47 cycles/byte: CubeHash16/32, `brick`, amd64 architecture.
- 12.60 cycles/byte: SHA-512, `brick`, amd64 architecture.
- 12.60 cycles/byte: SHA-512, `katana`, amd64 architecture.
- 12.66 cycles/byte: CubeHash16/32, `katana`, amd64 architecture.
- 12.74 cycles/byte: CubeHash16/32, `brick`, x86 architecture.
- 14.07 cycles/byte: CubeHash16/32, `katana`, x86 architecture.
- 15.43 cycles/byte: SHA-256, `brick`, x86 architecture.
- 15.53 cycles/byte: SHA-256, `brick`, amd64 architecture.
- 15.56 cycles/byte: SHA-256, `katana`, amd64 architecture.
- 17.76 cycles/byte: SHA-512, `brick`, x86 architecture.
- 20.00 cycles/byte: SHA-512, `katana`, x86 architecture.
- 22.76 cycles/byte: SHA-256, `katana`, x86 architecture.

Note that NIST has specified a 2.4GHz "Intel Core 2 Duo Processor" (without a microarchitecture specification) as the SHA-3 reference platform.

Security criteria. NIST also announced in Leuven that submissions would be considered "broken" and discarded if they were subject to attacks using

- 2^{100} "computations" and 2^{80} "memory" for 224-bit outputs;
- 2^{120} "computations" and 2^{100} "memory" for 256-bit outputs;
- 2^{180} "computations" and 2^{150} "memory" for 384-bit outputs; or
- 2^{240} "computations" and 2^{200} "memory" for 512-bit outputs.

Standard attacks produce disastrous collisions in *any* SHA-3 candidate using approximately 2^{256} hash-function evaluations, with negligible memory consumption and essentially unlimited parallelism. The same attacks have roughly one chance in a billion of succeeding within 2^{240} hash-function evaluations. NIST has not articulated any plan to rescue SHA-3 from such large attacks. Fortunately, 2^{240} hash-function evaluations are beyond any computation that will ever be carried out, according to typical estimates.

NIST also announced that submissions would be considered "wounded" if they were subject to, e.g., a 2^{128}-"computation" preimage attack on a 256-bit hash. In the absence of statements to the contrary one is forced to conclude that NIST will consider a 2^{256}-"computation" preimage attack as "wounding" a 512-bit hash, even though such large attacks will never actually be carried out. It is interesting to observe that Grover's algorithm solidly "wounds" every SHA-3 submission according to the same criterion: it computes 256-bit preimages using only 2^{128} operations on a small quantum computer, certainly less expensive than a conventional computer capable of carrying out 2^{256} operations.

CubeHash security review. The fastest attacks known against this SHA-3 proposal are attacks discussed in the original CubeHash submission documents:

- roughly 2^{112} operations for collisions in SHA-3–224,
- roughly 2^{128} operations for collisions in SHA-3–256,
- roughly 2^{192} operations for collisions in SHA-3–384–normal or formal,
- roughly 2^{224} operations for preimages in SHA-3–224,
- roughly 2^{256} operations for preimages in SHA-3–256,
- roughly 2^{256} operations for collisions in SHA-3–512–normal or formal,
- roughly 2^{384} operations for preimages in SHA-3–384–normal,
- roughly 2^{384} operations for preimages in SHA-3–384–formal,
- roughly 2^{384} operations for preimages in SHA-3–512–normal, and
- roughly 2^{512} operations for preimages in SHA-3–512–formal.

In particular, the "Complexity of generic attacks" document in the original CubeHash submission presents details of a "standard generic preimage attack" using roughly 2^{512-4b} operations for CubeHashr/b; e.g., 2^{384} operations for CubeHash16/32.

Summary of other published analyses of CubeHash:

- Aumasson, Meier, Naya-Plasencia, Peyrin, "Inside the hypercube": Variants of the standard generic preimage attack, trying to streamline the individual operations.
- Khovratovich, Nikolic, Weinmann, "Preimage attack on CubeHash512-$r/4$ and CubeHash512-$r/8$": Republication of the same attack.
- Aumasson, "Collision for CubeHash2/120-512"; Dai, "Collisions for CubeHash1/45 and CubeHash2/89"; Brier, Peyrin, "Cryptanalysis of CubeHash"; Brier, Khazaei, Meier, Peyrin, "Attack for CubeHash-2/2 and collision for CubeHash-3/64" and "Real Collisions for CubeHash-4/64": actual collisions for CubeHash2/3 and CubeHash4/48; estimated collision time slightly below 2^{200} for CubeHash2/2 and CubeHash4/4.
- Salaev, Rao, "Logical cryptanalysis of CubeHash using a SAT solver": Some automated attacks on CubeHash2/b, not as fast as previous attacks.
- Bloom, Janis, "Inference attacks on CubeHash": Attacks on CubeHash$r$/128, similar to previous attacks.
- Wang, Wilson, "Parallel collision search attack on hash function": Report of an implementation of the generic van Oorschot–Wiener attack.

None of these analyses affect the security of CubeHash8/1, CubeHash16/32, etc.

The "SHA-3–512–formal" proposal is aimed at users who are (1) concerned with attacks using 2^{384} operations, (2) unconcerned with quantum attacks that cost far less, and (3) unaware that attackers able to carry out 2^{256} operations would wreak havoc on the entire SHA-3 landscape, forcing SHA-3 to be replaced no matter which function is selected as SHA-3. The "SHA-3–512–normal" proposal is aimed at sensible users.