CubeHash

http://cubehash.cr.yp.to

D. J. Bernstein
University of Illinois at Chicago

NSF ITR–0716498
CubeHash is the *smallest* high-security SHA-3 proposal.

Several meanings of “smallest”:
- Smallest memory use.
- Smallest description.
- Smallest code size.
- Smallest vector-code size.
- Smallest area in hardware.

Also good security/speed tradeoff. Default parameters are extremely conservative, but faster-than-MD5 parameters are still resisting all attacks.
Memory use

CubeHash-512 (512-bit output) fits into 128 bytes of RAM.

Closest competitor: Keccak; 200 bytes of RAM.

Skein-512: 224 bytes of RAM?

SHA-512: 256 bytes of RAM (plus counter if necessary).
64 for current state,
64 for beginning-of-block state,
128 for transformed block.

Most submissions: Much larger.
CubeHash $b$-byte message block is xor’ed into first $b$ state bytes. Does *not* take extra space.

Standard generic attacks: $2^{512-4b+\text{overhead}}$ bit operations to find preimages etc.

Default parameter: $b = 1$; $2^{508+\text{overhead}}$ bit operations. $b = 8$ is $\approx 8 \times$ faster; $2^{480+\text{overhead}}$ bit operations. $b = 32$ is $\approx 32 \times$ faster; $2^{384+\text{overhead}}$ bit operations.

Attack details: See submission.
Code size

After $b$-byte message block, CubeHash$\frac{r}{b}$ transforms state (as 32 little-endian 4-byte words) through $r$ identical rounds.

Finalization: Flip one bit; $10r$ extra identical rounds; output first 64 state bytes.

Initialization: Similarly easy.

SHA-3 proposal: CubeHash8/1; 8 rounds after each byte.

Faster: CubeHash8/2, CubeHash8/4, CubeHash8/8, etc.
First half of a round:

for (i = 0; i < 16; ++i)
    x[i + 16] += x[i];
for (i = 0; i < 16; ++i)
    y[i ^ 8] = x[i];
for (i = 0; i < 16; ++i)
    x[i] = ROTATE(y[i], 7);
for (i = 0; i < 16; ++i)
    x[i] ^= x[i + 16];
for (i = 0; i < 16; ++i)
    y[i ^ 2] = x[i + 16];
for (i = 0; i < 16; ++i)
    x[i + 16] = y[i];
Second half of a round:

for (i = 0; i < 16; ++i)
    x[i + 16] += x[i];
for (i = 0; i < 16; ++i)
    y[i ^ 4] = x[i];
for (i = 0; i < 16; ++i)
    x[i] = ROTATE(y[i], 11);
for (i = 0; i < 16; ++i)
    x[i] ^= x[i + 16];
for (i = 0; i < 16; ++i)
    y[i ^ 1] = x[i + 16];
for (i = 0; i < 16; ++i)
    x[i + 16] = y[i];
Reduced-round cryptanalysis

Traditional confidence-building: see how many rounds survive third-party cryptanalysis.

Culmination of attacks by Aumasson, Brier, Dai, Khazaei, Meier, Naya-Plasencia, Peyrin: collision in CubeHash3/64; \(2^{231}\) for CubeHash5/64.

Do you have an improved attack? http://cubehash.cr.yp.to/prizes.html
## Speed: cycles/byte from eBASH

<table>
<thead>
<tr>
<th>CubeHash8/32</th>
<th>CPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.42</td>
<td>64 Core i7 920</td>
</tr>
<tr>
<td>6.03</td>
<td>64 Phenom 9550</td>
</tr>
<tr>
<td>6.29</td>
<td>64 Core 2 Duo 6f6</td>
</tr>
<tr>
<td>6.44</td>
<td>32 Core 2 Duo 6f6</td>
</tr>
<tr>
<td>7.32</td>
<td>32 Phenom 9550</td>
</tr>
<tr>
<td>9.00</td>
<td>64 Atom 330</td>
</tr>
<tr>
<td>9.47</td>
<td>32 Atom 330</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SHA-512</th>
<th>CPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.41</td>
<td>64 Core i7 920</td>
</tr>
<tr>
<td>9.92</td>
<td>64 Phenom 9550</td>
</tr>
<tr>
<td>13.09</td>
<td>64 Core 2 Duo 6f6</td>
</tr>
<tr>
<td>116.61</td>
<td>32 Core 2 Duo 6f6</td>
</tr>
<tr>
<td>20.31</td>
<td>32 Phenom 9550</td>
</tr>
<tr>
<td>17.58</td>
<td>64 Atom 330</td>
</tr>
<tr>
<td>59.92</td>
<td>32 Atom 330</td>
</tr>
</tbody>
</table>