Inside the Hypercube

Jean-Philippe Aumasson[1*], Willi Meier[1†], María Naya-Plasencia[2‡], and Thomas Peyrin[3]

[1] FHNW, Windisch, Switzerland
[2] INRIA project-team SECRET, France
[3] Ingenico, France

Some force inside the Hypercube occasionally manifests itself with deadly results.
`http://www.staticzombie.com/2003/06/cube_2_hypercube.html`

Abstract. Bernstein's CubeHash is a hash function family that includes four functions submitted to the NIST Hash Competition. A CubeHash function is parametrized by a number of rounds r, a block byte size b, and a digest bit length h. The 1024-bit internal state of CubeHash is represented as a five-dimension hypercube. Submissions to NIST have $r = 8$, $b = 1$, and $h \in \{224, 256, 384, 512\}$.
This paper gives the first external analysis of CubeHash, with
- improved standard generic attacks for collisions and preimages
- a multicollision attack that exploits fixed points
- a study of the round function symmetries
- a preimage attack that exploits these symmetries
- a practical collision attack on a weakened version of CubeHash
- high-probability truncated differentials over the 8-round transform

Our results do not contradict the security claims about CubeHash.

1 CubeHash

Bernstein's CubeHash is a hash function family that includes four functions submitted to the NIST Hash Competition. A CubeHash function is parametrized by a number of rounds r, a block byte size b, and a digest bit length h; the 1024-bit internal state of CubeHash is viewed as a five dimensional hypercube. Submissions to NIST have $r = 8$, $b = 1$, and $h \in \{224, 256, 384, 512\}$.

CubeHash computes the digest of a message as follows:

- initialize a 1024-bit state as a function of (h, b, r)
- append to the message a 1 bit and enough 0 bits to reach a multiple of $8b$ bits
- for each b-byte message block:
  - xor the block into the first b bytes of the state
  - transform the state through the r-round T function
- xor a 1 bit with the 993th bit of the state
- transform the state through $10r$-round T
- output the first h bits of the state

Let $x[0], \ldots, x[31]$ represent the 1024-bit state as an array of 32-bit words. The transform function T makes r identical rounds, where each round computes (see also Fig. 1):

---

*Supported by the Swiss National Science Foundation under project no. 113329.
†Supported by GEBERT RÜF STIFTUNG, project no. GRS-069/07.
‡Supported in part by the French Agence Nationale de la Recherche under contract ANR-06-SETI-013-RAPIDE.

for $i = 0, \ldots, 15$: $x[i+16] = x[i+16] + x[i]$
for $i = 0, \ldots, 15$: $y[i \oplus 8] = x[i]$
for $i = 0, \ldots, 15$: $x[i] = y[i] \lll 7$
for $i = 0, \ldots, 15$: $x[i] = x[i] \oplus x[i+16]$
for $i = 0, \ldots, 15$: $y[i \oplus 2] = x[i+16]$
for $i = 0, \ldots, 15$: $x[i+16] = y[i]$
for $i = 0, \ldots, 15$: $x[i+16] = x[i+16] + x[i]$
for $i = 0, \ldots, 15$: $y[i \oplus 4] = x[i]$
for $i = 0, \ldots, 15$: $x[i] = y[i] \lll 11$
for $i = 0, \ldots, 15$: $x[i] = x[i] \oplus x[i+16]$
for $i = 0, \ldots, 15$: $y[i \oplus 1] = x[i+16]$
for $i = 0, \ldots, 15$: $x[i+16] = y[i]$

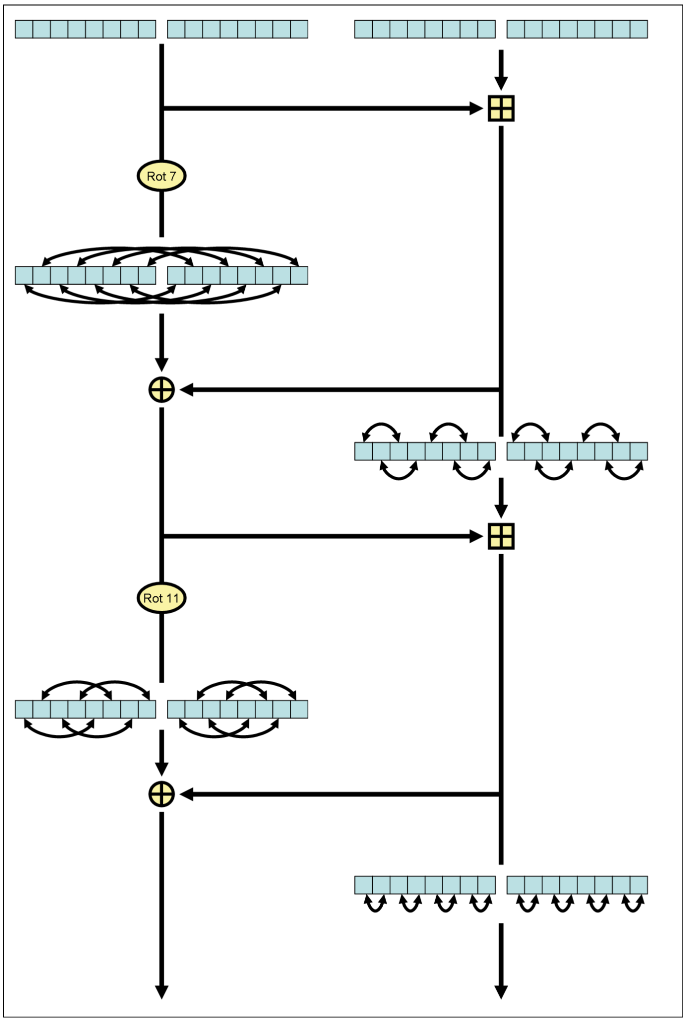

Fig. 1. Schematic view of a CubeHash round.

2 Improved standard generic attacks

The author of CubeHash presented [1] the following "standard preimage attack":

- from (h, b, r) compute the initial state S_0
- from the h-bit image plus some arbitrary $/1024 - h)$ bits, invert $10r$ rounds and the "xor 1" to get a state S_f before finalization
- find two n-block sequences that map S_0 (forward) and S_f (backward), respectively, to two states that share the last $(1024 - 8b)$ bits

There are 2^{nb} possible n-block inputs and one looks for a collision over $(1024 - 8b)$ bits. For a success chance $1 - 1/e \approx 0.63$ one thus requires 2^{512-4b} trials in each direction, that is, $2nb > 1024 - 8b$, i.e., $n > 512/b - 4$. In total the number of evaluations of T is approximately

$$2 \times \left(\frac{512}{b} - 4\right) \times 2^{512-4b} \approx 2^{522-4b-\log b} .$$

Furthermore, [1] estimates that each round of T needs 2^{11} "bit operations"; the above formula gives about $2^{533-4b-\log b+\log r}$ bit operations.

A speed-up of the above attack can be obtained by searching a collision not only in the states resulting of a n-block computation, but in every distinct state reached (i.e. also with the intermediate states). This is made possible by the absence of message length padding. Each call to T gives a new candidate for the collision search; we thus get rid of the $(512/b-4)$ multiplicative factor in the cost estimate. This gives a cost of

$$2 \times 2^{512-4b} = 2^{513-4b}$$

evaluations of T, i.e. $2^{524-4b+\log r}$ bit operations.

The proposed CubeHash-512 has $(h, b, r) = (512, 1, 8)$, our attack thus makes 2^{523} bit operations, against 2^{532} with the original attack. If $r = 8$, our attack needs $b > 3$ to make less than 2^{512} bit operations, against $b > 5$ with the original preimage attack. It is to note that these estimates exclude the nonnegligible communication costs.

One can use the same trick to speed-up the standard collision attack [1]; the cost in T evaluations then drops from $2^{521-4b-\log b}$ to 2^{512-4b}.

3 Narrow-pipe multicollisions

Based on the "narrow-pipe" attacks in [2], we show a multicollision attack on CubeHash faster than Joux's [5] or birthday [4,7] methods (for large b's). Our attack requires the same amount of computation as narrow-pipe collisions. It exploits the fact that the null state is a fixed point for the compression function T (regardless of r), and that the message padding doesn't include the message length.

Starting from an initial state S_0 derived from (h, b, r), one finds two n-block sequences m and m' that map S_0 (forward) and the zero state (backward), respectively, to two states that share the last $(1024 - 8b)$ bits. One finds a connection of the form

$$\begin{aligned} S_0 \oplus m_1 &\xrightarrow{T} S_1 \\ S_1 \oplus m_2 &\xrightarrow{T} \cdots \\ &\cdots \\ \cdots &\xrightarrow{T} S'_1 \\ S'_1 \oplus m'_2 &\xrightarrow{T} 0 \oplus m'_1 \end{aligned}$$

Once a path to the zero state is found, one can add an arbitrary number of zero message blocks to maintain a zero state. Colliding messages are of the form

$$m\|m'\|0\|0\|\dots\|0\|\bar{m},$$

where $\bar{m}$ is an arbitrary sequence of blocks.

Using the technique of §2, this multicollision attack requires approximately 2^{513-4b} evaluations of T. In comparison, a birthday attack finds a k-collision in $(k! \times 2^{n(k-1)})^{1/k}$ trials, and Joux's attacks in $\log k \times 2^{4(128-b)}$. For example, with $h = 512$ and $b = 112$, our attack finds 2^{64}-collisions within 2^{65} calls to T, against $> 2^{512}$ for a birthday attack and 2^{70} for Joux's.

4 On state symmetries

The documentation of CubeHash mentions [3, p.3] the existence of symmetries through the round function, and states that the initialization of CubeHash was designed to avoid symmetries. However [3] gives no detail on those symmetries. In this section we present five symmetry classes of 2^{512} states each, and show how to exploit them.

4.1 Symmetry classes

If a 32-word state x satisfies $x[0] = x[1]$, $x[2] = x[3]$, ..., $x[30] = x[31]$, then this property is preserved through the transformation T, for any number of rounds. One can represent this symmetry with the pattern (each letter stands for a 32-bit word):

`AABBCCDD EEFFGGHH IIJJKKLL MMNNOOPP` .

In total we found five classes of symmetry:

C_1 : `AABBCCDD EEFFGGHH IIJJKKLL MMNNOOPP`
C_2 : `ABABCDCD EFEFGHGH IJIJKLKL MNMNOPOP`
C_3 : `ABCDABCD EFGHEFGH IJKLIJKL MNOPMNOP`
C_4 : `ABCDEFGH ABCDEFGH IJKLMNOP IJKLMNOP`
C_5 : `ABBACDDC EFFEGHHG IJJIKLLK MNNMOPPO`

Each class contains 2^{512} states. If a state belongs to several classes, then its image under T also belongs to these classes; for example if $S \in (C_i \cap C_j)$, then $T(S) \in (C_i \cap C_j)$. We have

$$|C_1 \cap C_2| = |C_1 \cap C_5| = |C_2 \cap C_3| = |C_2 \cap C_5| = |C_3 \cap C_4| = 256$$
$$|C_1 \cap C_3| = |C_2 \cap C_4| = |C_3 \cap C_5| = 128$$
$$|C_1 \cap C_4| = |C_4 \cap C_5| = 64$$

We thus have $\left|\cup_{i=1}^{5} C_i\right| \geq 5 \times 2^{512} - 10 \times 2^{256} \approx 2^{514.3}$ distinct symmetric states. Note that symmetry is not preserved by the finalization procedure of CubeHash (the "xor 1" breaks any of the above symmetries).

4.2 Exploiting symmetric states

Preimages. Given a target digest, one can make a preimage attack similar to that in §2, and exploit symmetric states for the connection. The attack goes as follows:

- from the initial state, reach a symmetric state (of any class) by using $2^{1024-514-8} = 2^{502}$ message blocks
- from a state before finalization, reach (backwards) another symmetric state (not necessarily of the same class)
- from these two symmetric states in classes C_i and C_j, use null message blocks in both directions to reach two states in $C_i \cap C_j$
- find a collision by trying $\sqrt{|C_i \cap C_j|}$ messages in each direction

Complexity of steps 1 and 2 is about 2^{503} computations of T. The cost of steps 3 and 4 depends on i and j; there are three distinct cases (counting in calls to T):

1. $i = j$ (with prob. 5/25): step 3 costs 0 and step 4 costs 2×2^{256}
2. $|C_i \cap C_j| = 2^{256}$ (with prob. 10/25): step 3 costs 2×2^{256} and step 4 costs 2×2^{128}
3. $|C_i \cap C_j| = 2^{128}$ (with prob. 6/25): step 3 costs 2×2^{384} and step 4 costs 2×2^{64}
4. $|C_i \cap C_j| = 2^{64}$ (with prob. 4/25): step 3 costs 2×2^{448} and step 4 costs 2×2^{32}

In any case, the total complexity is about 2^{503} calls to T. This attack, however, finds messages of unauthorized size (more than 2^{257} bytes!).

One can find preimages of reasonable size by using a variant of the above attack: suppose $b > 4$, from the initial state reach a state in C_1, do the same backwards from a state before finalization. Then one seeks a collision within C_1 by trying messages preserving the symmetry: for example, if $b = 5$, one has to preserve the equality $x[0] = x[1]$ and shall thus pick 5-byte messages of the form `X000X` (each digit stands for a byte). The cost of reaching a C_1 state depends on b:

- if $b \equiv 0 \bmod 8$, there are $(1024 - 8b)/2 = 512 - 4b$ equations to satisfy, thus about 2^{512-4b} calls to T are necessary
- if $b \equiv 4 \bmod 8$, there are only $(1024 - 8b - 32)/2 = 496 - 4b$ equations to satisfy, because one has no condition on the first state word not xored with the message block
- generalizing, when $b \bmod 8 \leq 4$, about $2^{512-4(b+(b \bmod 4))}$ calls to T are necessary
- when $b \bmod 8 > 4$, there are $(1024 - 8b - 32 + 8(b \bmod 4))/2$ equations to satisfy, which gives a cost $2^{496-4(b-(b \bmod 4))}$

The general formula is

$$2^{512-32\lfloor b/8\rfloor-32\lfloor(b \bmod 8)/4\rfloor-[(\lfloor(b \bmod 8)/4\rfloor+1) \bmod 2]\times 8(b \bmod 4)}$$

In the best case ($b \equiv 4 \bmod 8$), the attack is 2^{15} times faster than that in §2. In the worst case ($b \equiv 0 \bmod 8$), it has the same complexity. Note that when $b = 5$, the attack makes about 2^{481} calls to T, against 2^{493} with the attack in §2.

Collisions on a weakened CubeHash. The initialization of CubeHash never leads to a symmetric initial state. Here we present a practical collision attack that would apply if the initial state were symmetric, and in $C_1 \cap C_4$.

Suppose that the initial state of CubeHashr/b-h is in $C_1 \cap C_4$, i.e. is of the form

`AAAAAAAA AAAAAAAAA BBBBBBBBB BBBBBBBB .`

If one hashes the $b2^{33}$-byte message that contain only zeros, then each of the 2^{33} intermediate states is an element of $C_1 \cap C_4$. Assuming that T acts like a random permutation of $C_1 \cap C_4$, one will find two identical states with probability about 0.63, which directly gives a collision.

5 Truncated differentials over T

We analyse linear differentials over the T transform, and use them to empirically detect high-probability truncated differentials.

We start from the input difference 80000000 in $x[16]$; $x[16]$ was chosen because words $x[16] \cdots x[31]$ diffuse less in the first rounds than $x[0] \cdots x[15]$, and to minimize the index in order to minimize b (to control $x[16]$ one needs $b \geq 68$). We chose 80000000 to minimize the impact of carries.

The weight-1 difference above gives a weight-5 difference with probability 1 after one round. Using the inverse transform function T^{-1}, we identify a difference that gives after one round a difference in 80000000 in $x[16]$ with probability 2^{-5} for random bits in $x[17] \cdots x[31]$; with probability 2^{-2} for random bits only in $x[31]$ and a particular choice of the other bits; with probability 2^{-3} for random bits in $x[28] \cdots x[31]$. To summarize, the input difference used is (printing words from left-top to right-bottom)

```
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 01000000
00000000 00002000 00000000 00002000
00000000 00000000 00000000 80000000
00000000 00000000 00000000 00100000
00000000 00000000 00000000 01000000
00000000 00002000 00000000 00006000
```

After a round this gives with some nonzero probability the weight-1 difference

```
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
80000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
```

which after another round gives with probability 1 the difference

```
80000000 00000000 80000000 00000000
00000400 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 80000000 00000000 80000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
```

In the linear model (i.e. when additions are replaced by xors), the differential path cycles over 47 rounds, that is, it comes back to the difference 80000000 in $x[16]$ after 47 rounds. In the original model, however, linear differentials are followed with negligible probability after only a few rounds. Nevertheless, one can use the 2-round differential above to empirically identify 1-to-1 truncated differentials over more rounds. We detail these results below.

We empirically looked for high-probability truncated differentials, starting from the weight-8 input difference, and applying to each output bit a frequency test similar to that in [6, §2.1], with decision treshold 0.001 and 2^{20} samples.

First, we consider as output the *first 512 state bits*, i.e. the maximum number of bits outputable by CubeHash (note that [3] defines $h \in \{8, 16, 24, \ldots, 512\}$). We initialize a message to $x[0] = \cdots = x[15] = 0$,

$$\begin{array}{llll} x[16] = \texttt{00000000} & x[17] = \texttt{4335A2F2} & x[18] = \texttt{6C2774B5} & x[19] = \texttt{184555F5} \\ x[20] = \texttt{6E359435} & x[21] = \texttt{6D8D994C} & x[22] = \texttt{0768D703} & x[23] = \texttt{16DA5B5A} \\ x[24] = \texttt{6FE4A1B6} & x[25] = \texttt{6C52326A} & x[26] = \texttt{23BEEFB7} & x[27] = \texttt{5587CDF0} \end{array}$$

and 128 random bits in $x[28] \cdots x[31]$, then apply the weight-8 difference, transform both messages with 7-round T, and collect the p-values of the frequency test for each output bit. We observe that about 30 bits have p-value less than 0.001, against none for 8 or more rounds. For example the output bits 35, 99, 498, and 499 have null p-value.

Then, we consider as output the *1024 state bits*. We set $x[0] \ldots, x[27]$ to the same values as above, and in addition set

$$x[28] = \texttt{0E22B0EE} \quad x[29] = \texttt{41F13BBA} \quad x[31] = \texttt{179C53D5}$$

and 32 random bits in $x[30]$. Over 8 T rounds, we found 5 output bits with p-value less than 0.001, at positions 579, 777, 778, 841, 842. These bits show biases about 2^{-9}. Over 9 rounds or more, no bias was detected.

These observations indicate that 8-round T does not act as a random permutation, and that 10 rounds may not be overkill, as suggested in [2]. However, the methods used don't correspond to realistic attack scenarios, since we consider differences in $x[31]$. Furthermore, if we restrict ourselves to differences in the first state byte, and put random bits in the rest of the state, then we observe nonrandomness after up to 5 rounds.

References

1. Daniel J. Bernstein. CubeHash appendix: complexity of generic attacks. Submission to NIST, 2008.
2. Daniel J. Bernstein. CubeHash attack analysis (2.B.5). Submission to NIST, 2008.
3. Daniel J. Bernstein. CubeHash specification (2.B.1). Submission to NIST, 2008.
4. Persi Diaconis and Frederick Mosteller. Methods for studying coincidences. *Journal of the American Statistical Association*, 84(408):853–861, 1989.
5. Antoine Joux. Multicollisions in iterated hash functions. application to cascaded constructions. In *CRYPTO*, 2004.
6. NIST. SP 800-22, a statistical test suite for random and pseudorandom number generators for cryptographic applications, 2001.
7. Kazuhiro Suzuki, Dongvu Tonien, Kaoru Kurosawa, and Koji Toyota. Birthday paradox for multi-collisions. In *ICISC*, 2006.