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.

ash

//cubehash.cr.yp.to

Bernstein

sity of Illinois at Chicago

TR–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.

Memor

CubeH
fits int

Closest
Keccak

Skein-5

SHA-5
(plus c
64 for
64 for
128 for

Most s

sh.cr.yp.to

ois at Chicago

98

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 (5
fits into 128 byte

Closest competit
Keccak; 200 byte

Skein-512: 224 b

SHA-512: 256 by
(plus counter if n
64 for current sta
64 for beginning-
128 for transform

Most submissions

.to

cago

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 ou
fits into 128 bytes of RAM

Closest competitor:
Keccak; 200 bytes of RAM

Skein-512: 224 bytes of RA

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

Most submissions: Much la

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.

ash is the *smallest*
ecurity SHA-3 proposal.

l meanings of "smallest":
lest memory use.
lest description.
lest code size.
lest vector-code size.
lest area in hardware.

ood security/speed tradeoff.
t parameters are
ely conservative,
ster-than-MD5 parameters
l 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.

CubeH
is xor'e
Does *n*

Standa
2^{512-4}

to find

Default
2^{508+o}

$b = 8$ i
2^{480+o}

$b = 32$
2^{384+o}

Attack

smallest

A-3 proposal.

s of "smallest":

ory use.

ption.

size.

r-code size.

n hardware.

ty/speed tradeoff.

ers are

vative,

MD5 parameters

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
is xor'ed into firs
Does *not* take ex

Standard generic
$2^{512-4b+\text{overhead}}$

to find preimages

Default paramete
$2^{508+\text{overhead}}$ bit

$b = 8$ is $\approx 8\times$ fa
$2^{480+\text{overhead}}$ bit

$b = 32$ is $\approx 32\times$
$2^{384+\text{overhead}}$ bit

Attack details: S

sal.

lest":

e.

e.

radeoff.

meters

s.

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
is xor'ed into first b state b
Does *not* take extra space.

Standard generic attacks:
$2^{512-4b+\text{overhead}}$ bit operat
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 submis

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.

ry use

ash-512 (512-bit output)
o 128 bytes of RAM.

t competitor:
k; 200 bytes of RAM.

512: 224 bytes of RAM?

12: 256 bytes of RAM
ounter if necessary).
current state,
beginning-of-block state,
r transformed block.

ubmissions: 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 s

After b
CubeH
(as 32
throug

Finaliz
$10r$ ex
output

Initializ

SHA-3
8 roun

Faster:
CubeH

512-bit output)
es of RAM.

or:
es of RAM.

ytes of RAM?

ytes of RAM
necessary).
ate,
-of-block state,
ned block.

s: 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 mess
CubeHashr/b tra
(as 32 little-endia
through r identic

Finalization: Flip
$10r$ extra identic
output first 64 st

Initialization: Sim

SHA-3 proposal:
8 rounds after ea

Faster: CubeHas
CubeHash8/4, C

tput)
.
.
AM?
AM
state,
arger.

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 CubeHashr/b transforms s (as 32 little-endian 4-byte through r identical rounds.

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

Initialization: Similarly eas

SHA-3 proposal: CubeHas 8 rounds after each byte.

Faster: CubeHash8/2, CubeHash8/4, CubeHash8

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, CubeHashr/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.

ash b-byte message block
ed into first b state bytes.
ot take extra space.

rd generic attacks:
$^{b+\text{overhead}}$ bit operations
preimages etc.

t parameter: $b = 1$;
$^{\text{verhead}}$ bit operations.
s $\approx 8\times$ faster;
$^{\text{verhead}}$ bit operations.
is $\approx 32\times$ faster;
$^{\text{verhead}}$ bit operations.

details: See submission.

Code size

After b-byte message block,
CubeHashr/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 ha

```
for (i
  x[i
for (i
  y[i
for (i
  x[i]
for (i
  y[i
for (i
  x[i
```

e message block
t b state bytes.
xtra space.

attacks:
bit operations
s etc.

er: $b = 1$;
operations.
ster;
operations.
faster;
operations.

See submission.

Code size

After b-byte message block,
CubeHashr/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 ro

```
for (i = 0;i <
  x[i + 16] +=
for (i = 0;i <
  y[i ^ 8] = x
for (i = 0;i <
  x[i] = ROTA
for (i = 0;i <
  x[i] ^= x[i
for (i = 0;i <
  y[i ^ 2] = x
for (i = 0;i <
  x[i + 16] =
```

block
ytes.

ions

s.

s.

s.

sion.

Code size

After b-byte message block,
CubeHashr/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],
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];
```

Code size

After b-byte message block,
CubeHashr/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];
```

ize

-byte message block,
ashr/b transforms state
little-endian 4-byte words)
h r identical rounds.

ation: Flip one bit;
tra identical rounds;
first 64 state bytes.

zation: Similarly easy.

proposal: CubeHash8/1;
ds after each byte.

CubeHash8/2,
ash8/4, 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

```
for (i
  x[i
for (i
  y[i
for (i
  x[i]
for (i
  x[i]
```

sage block,
ansforms state
an 4-byte words)
cal rounds.

one bit;
al rounds;
tate bytes.

milarly easy.

CubeHash8/1;
ch byte.

h8/2,
ubeHash8/8, 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

```
for (i = 0;i <
  x[i + 16] +=
for (i = 0;i <
  y[i ^ 4] = x
for (i = 0;i <
  x[i] = ROTA
for (i = 0;i <
  x[i] ^= x[i
for (i = 0;i <
  y[i ^ 1] = x
for (i = 0;i <
  x[i + 16] =
```

k,
tate
words)
.

y.

h8/1;

/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],
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];
```

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];
```

alf of a round:

```
i = 0;i < 16;++i)
 + 16] += x[i];
i = 0;i < 16;++i)
 ^ 8] = x[i];
i = 0;i < 16;++i)
] = ROTATE(y[i],7);
i = 0;i < 16;++i)
] ^= x[i + 16];
i = 0;i < 16;++i)
 ^ 2] = x[i + 16];
i = 0;i < 16;++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];
```

Reduce

Traditi
see hov
third-p

Culmin
Aumas
Meier,
collisio
2^{231} fo

Do you

http:/
/prize

und:

```
< 16;++i)
= x[i];
< 16;++i)
x[i];
< 16;++i)
TE(y[i],7);
< 16;++i)
 + 16];
< 16;++i)
x[i + 16];
< 16;++i)
 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 c

Traditional confi
see how many ro
third-party crypt

Culmination of a
Aumasson, Brier
Meier, Naya-Plas
collision in Cube
2^{231} for CubeHa

Do you have an i

http://cubeha
/prizes.html

```
)
)
)
7);
)
)
];
)
```

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 cryptanalys

Traditional confidence-buil
see how many rounds survi
third-party cryptanalysis.

Culmination of attacks by
Aumasson, Brier, Dai, Kha
Meier, Naya-Plasencia, Pey
collision in CubeHash3/64;
2^{231} for CubeHash5/64.

Do you have an improved a

http://cubehash.cr.yp
/prizes.html

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

half of a round:

```
i = 0;i < 16;++i)
 + 16] += x[i];
i = 0;i < 16;++i)
 ^ 4] = x[i];
i = 0;i < 16;++i)
] = ROTATE(y[i],11);
i = 0;i < 16;++i)
] ^= x[i + 16];
i = 0;i < 16;++i)
 ^ 1] = x[i + 16];
i = 0;i < 16;++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:

CubeH

S

round:

```
< 16;++i)
= x[i];
< 16;++i)
x[i];
< 16;++i)
TE(y[i],11);
< 16;++i)
 + 16];
< 16;++i)
x[i + 16];
< 16;++i)
 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/by

CubeHash8/32	
4.42	6
6.03	6
6.29	6
6.44	3
7.32	3
9.00	6
9.47	3

SHA-512	
12.41	6
9.92	6
13.09	6
116.61	3
20.31	3
17.58	6
59.92	3

```
)
)
)
11);
)
)
];
)
```

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 el

CubeHash8/32	CPU
4.42	64 Core i7
6.03	64 Phenom
6.29	64 Core 2
6.44	32 Core 2
7.32	32 Phenom
9.00	64 Atom 3
9.47	32 Atom 3

SHA-512	CPU
12.41	64 Core i7
9.92	64 Phenom
13.09	64 Core 2
116.61	32 Core 2
20.31	32 Phenom
17.58	64 Atom 3
59.92	32 Atom 3

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

CubeHash8/32	CPU
4.42	64 Core i7 920
6.03	64 Phenom 9550
6.29	64 Core 2 Duo 6f6
6.44	32 Core 2 Duo 6f6
7.32	32 Phenom 9550
9.00	64 Atom 330
9.47	32 Atom 330

SHA-512	CPU
12.41	64 Core i7 920
9.92	64 Phenom 9550
13.09	64 Core 2 Duo 6f6
116.61	32 Core 2 Duo 6f6
20.31	32 Phenom 9550
17.58	64 Atom 330
59.92	32 Atom 330