Linearization Framework for Collision Attacks: Application to CubeHash and MD6

Eric Brier[1], Shahram Khazaei[2],
Willi Meier[3] and Thomas Peyrin[1]

[1]Ingenico, France [2]EPFL, Switzerland [3]FHNW, Switzerland

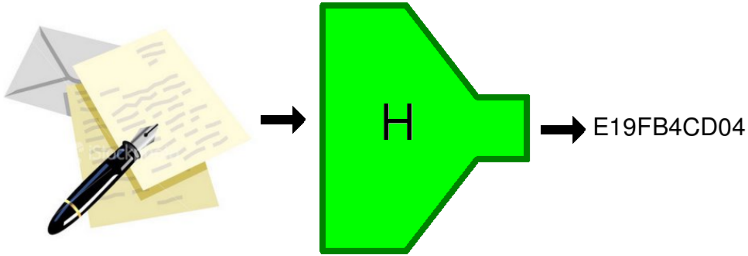
Asiacrypt 2009, Tokyo, Japan - December 10th

Outline

- State of the art of the Hash functions
- Description of CubeHash and MD6
- Attack overview
  - Finding differential paths
    - **linearization**
  - Finding conforming message
    - **condition function**
    - **dependency table**
    - **backtracking**
- Results
- Conclusion

Hash Function

$$H : \{0,1\}^\star \rightarrow \{0,1\}^n$$



Properties

collision resistance: $\mathcal{O}(2^{n/2})$
preimage resistance: $\mathcal{O}(2^n)$
second preimage resistance: $\mathcal{O}(2^n)$

State of the art of hash Function

Real collisions

MD4, MD5, SHA-0

Expected to be cracked next

SHA-1: 2^{60}

Currently used

SHA-2 family

Under review

SHA-3 candidates

Our targets

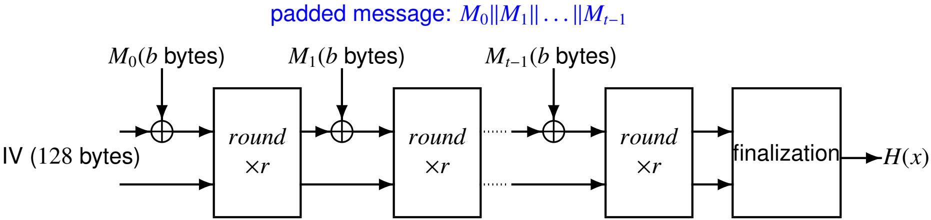
MD6

SHA-3 first round candidate (along with 50 others)
Internal state: 5696 bits (89 64-bit words)
hash output size: up to 512 bits
Operations: $\oplus$, $\lll$, $\ggg$, $\wedge$

CubeHash

SHA-3 second round candidate (along with 13 others)
Internal state size: 1024 bits (32 32-bit words)
hash output: up to 512 bits
Operations: $\oplus$, $\lll$, $\boxplus$

CubeHash-r/b

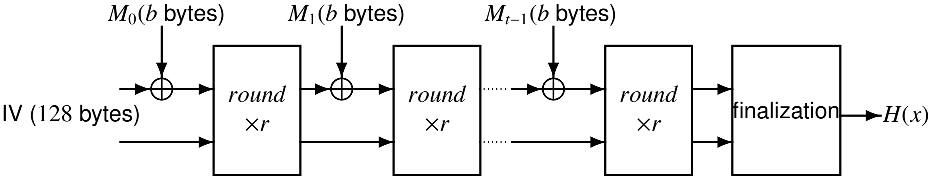

padded message: $M_0||M_1||\ldots||M_{t-1}$
$M_0(b$ bytes)
$M_1(b$ bytes)
$M_{t-1}(b$ bytes)
IV (128 bytes)
round $\times r$
round $\times r$
round $\times r$
finalization
$H(x)$

CubeHash-r/b

Speed

$25r/b$ cycles/byte on a Core 2 Duo in 32-bit mode
$20r/b$ cycles/byte on a Core 2 Duo in 64-bit mode

padded message: $M_0||M_1||\ldots||M_{t-1}$

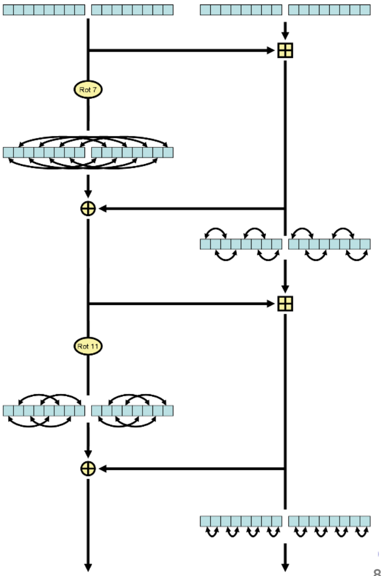


Official instances

Original SHA-3 proposal: CubeHash-8/1
Tweaked proposal: CubeHash-16/32 — ×16 faster

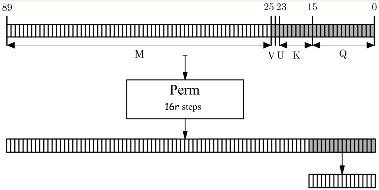
CubeHash round function

- Add S_i into $S_{i \oplus 16}$, for $0 \le i \le 15$.
- Rotate S_i to the left by seven bits, for $0 \le i \le 15$.
- Swap S_i and $S_{i \oplus 8}$, for $0 \le i \le 7$.
- XOR $S_{i \oplus 16}$ into S_i, for $0 \le i \le 15$.
- Swap S_i and $S_{i \oplus 2}$, for $i \in \{16, 17, 20, 21, 24, 25, 28, 29\}$.
- Add S_i into $S_{i \oplus 16}$, for $0 \le i \le 15$.
- Rotate S_i to the left by eleven bits, for $0 \le i \le 15$.
- Swap S_i and $S_{i \oplus 4}$, for $i \in \{0, 1, 2, 3, 8, 9, 10, 11\}$.
- XOR $S_{i \oplus 16}$ into S_i, for $0 \le i \le 15$.
- Swap S_i and $S_{i \oplus 1}$, for $i \in \{16, 18, 20, 22, 24, 26, 28, 30\}$.



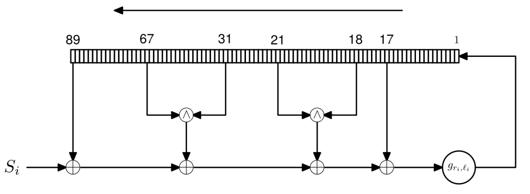
MD6

Compression function



$r = 80$ for 160-bit digests
$r = 168$ for 512-bit digests

Permutation



How to find collisions?

Find M and M' such that $H(M) = H(M')$.

Find Δ s.t. $H(M) = H(M \oplus \Delta)$ with high probability: Linearization.

Given Δ how to find M? Random search?

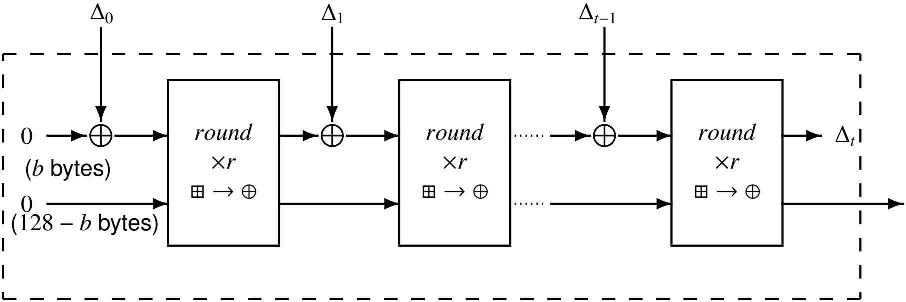
Better solution

- Construct a Condition function such that:

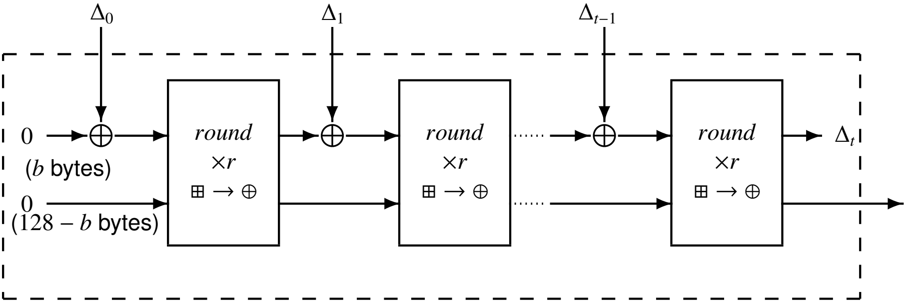
$$\text{Condition}_\Delta(M) = 0 \Longrightarrow H(M) = H(M \oplus \Delta)$$

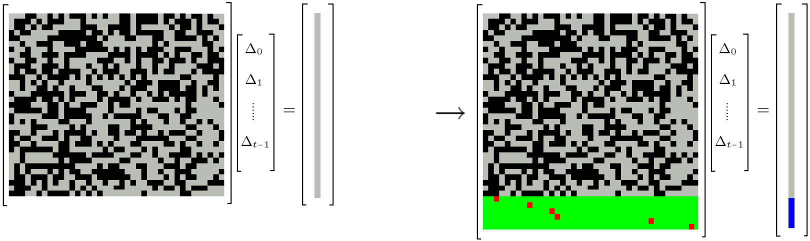
- Find preimages of Condition_Δ function under zero efficently.

Linearization

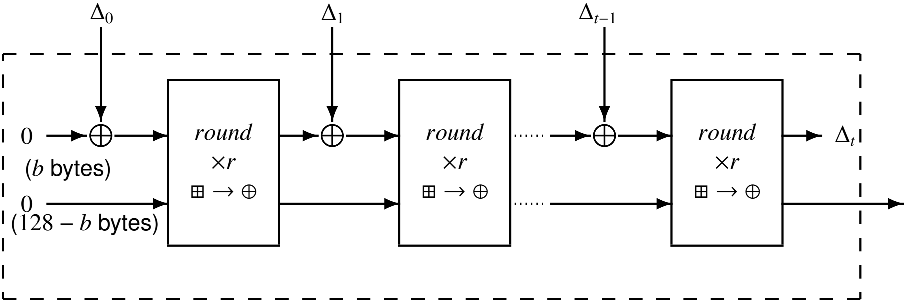


Linearization

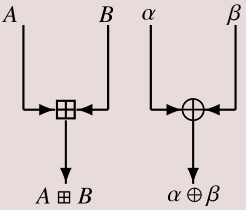




Linearization

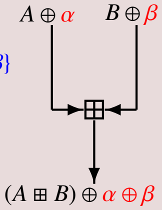


How to compute the probability of a path?



Lemma (Lipmaa-Moriai 2001)

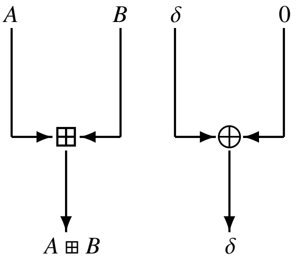
$$\Pr\{((A \oplus \alpha) \boxplus (B \oplus \beta)) \oplus (A \boxplus B) = \alpha \oplus \beta\} = 2^{-wt\left((\alpha \vee \beta) \wedge (2^{n-1}-1)\right)}$$

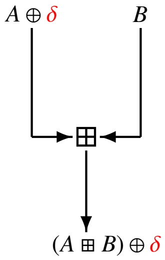


Constructing Condition function

Let n denote the wordsize and $\delta = 2^i$, $0 \leq i \leq n-2$.

Let $C = (A \boxplus B) \oplus A \oplus B$ denote the carry word of A and B.

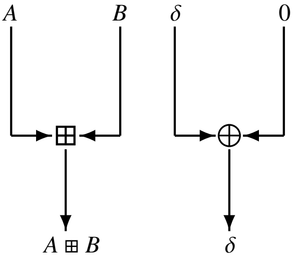




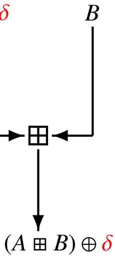
Constructing Condition function

Let n denote the wordsize and $\delta = 2^i$, $0 \le i \le n-2$.

Let $C = (A \boxplus B) \oplus A \oplus B$ denote the carry word of A and B.



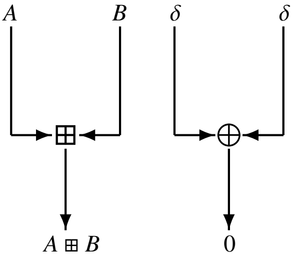
$B_i \oplus C_i = 0$



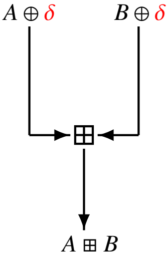
Constructing Condition function

Let n denote the wordsize and $\delta = 2^i$, $0 \leq i \leq n-2$.

Let $C = (A \boxplus B) \oplus A \oplus B$ denote the carry word of A and B.

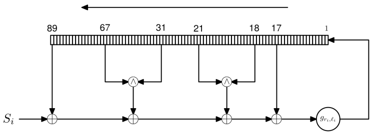


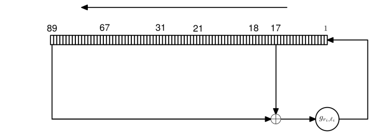
$A_i \oplus B_i \oplus 1 = 0$

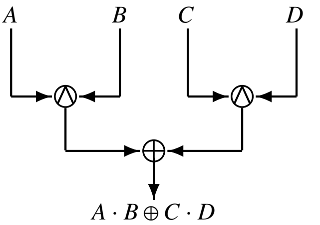


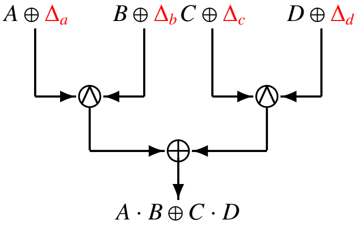
MD6: Linearization and condition function

Linearization



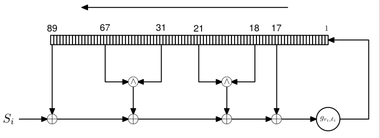


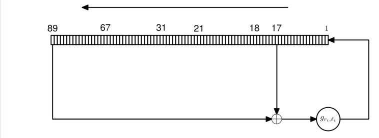


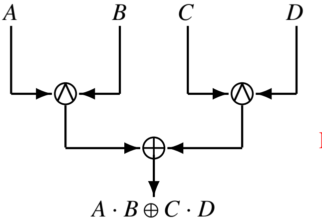


MD6: Linearization and condition function

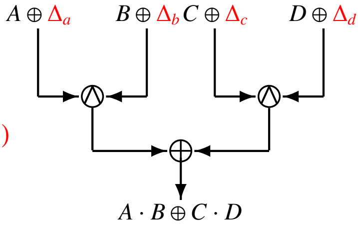
Linearization







$$\Pr = 2^{-wt\left(\Delta_a \vee \Delta_b \vee \Delta_c \vee \Delta_d\right)}$$



$$A \cdot B \oplus C \cdot D \oplus (A \oplus \Delta_a) \cdot (B \oplus \Delta_b) \oplus (C \oplus \Delta_c) \cdot (D \oplus \Delta_d) = 0$$

Finding preimages of Condition function

$$Y = \text{Condition}_{\Delta}(M) = 0$$

If Condition function is complex enough $\Longrightarrow 2^{|Y|}$

Y :

Finding preimages of Condition function

$$Y = \text{Condition}_{\Delta}(M) = 0$$

If Condition function is complex enough $\Longrightarrow 2^{|Y|}$

$Y:$ 

If input and output can be partitioned as

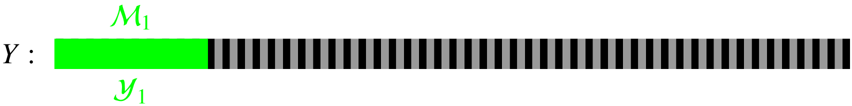
$$\mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_\ell \text{ and } \mathcal{Y}_1, \mathcal{Y}_2, \ldots, \mathcal{Y}_\ell$$

1. $\mathcal{M}_1 \Longrightarrow \mathcal{Y}_1$
2. $\mathcal{M}_1, \mathcal{M}_2 \Longrightarrow \mathcal{Y}_2$
3. $\vdots$
4. $\mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_\ell \Longrightarrow \mathcal{Y}_\ell$

Finding preimages of Condition function

$$Y = \text{Condition}_{\Delta}(M) = 0$$

If Condition function is complex enough $\Longrightarrow 2^{|Y|}$



If input and output can be partitioned as

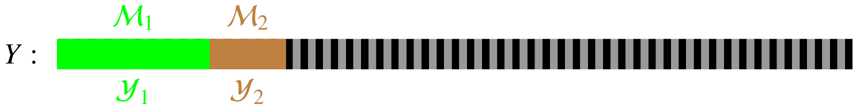
$$\mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_\ell \text{ and } \mathcal{Y}_1, \mathcal{Y}_2, \ldots, \mathcal{Y}_\ell$$

1. $\mathcal{M}_1 \Longrightarrow \mathcal{Y}_1$
2. $\mathcal{M}_1, \mathcal{M}_2 \Longrightarrow \mathcal{Y}_2$
3. $\vdots$
4. $\mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_\ell \Longrightarrow \mathcal{Y}_\ell$

Finding preimages of Condition function

$$Y = \text{Condition}_{\Delta}(M) = 0$$

If Condition function is complex enough $\implies 2^{|Y|}$



If input and output can be partitioned as

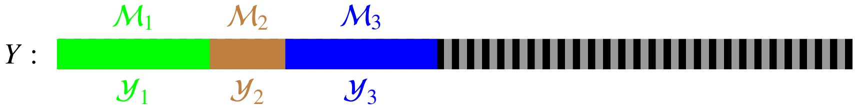
$$\mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_\ell \text{ and } \mathcal{Y}_1, \mathcal{Y}_2, \ldots, \mathcal{Y}_\ell$$

1. $\mathcal{M}_1 \implies \mathcal{Y}_1$
2. $\mathcal{M}_1, \mathcal{M}_2 \implies \mathcal{Y}_2$
3. $\vdots$
4. $\mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_\ell \implies \mathcal{Y}_\ell$

Finding preimages of Condition function

$$Y = \text{Condition}_{\Delta}(M) = 0$$

If Condition function is complex enough $\Longrightarrow 2^{|Y|}$



If input and output can be partitioned as

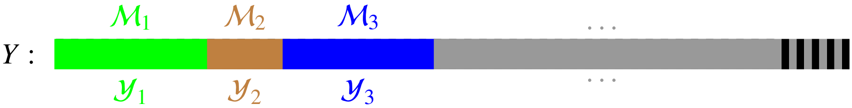
$$\mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_\ell \text{ and } \mathcal{Y}_1, \mathcal{Y}_2, \ldots, \mathcal{Y}_\ell$$

1. $\mathcal{M}_1 \Longrightarrow \mathcal{Y}_1$
2. $\mathcal{M}_1, \mathcal{M}_2 \Longrightarrow \mathcal{Y}_2$
3. $\vdots$
4. $\mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_\ell \Longrightarrow \mathcal{Y}_\ell$

Finding preimages of Condition function

$$Y = \text{Condition}_{\Delta}(M) = 0$$

If Condition function is complex enough $\Longrightarrow 2^{|Y|}$



If input and output can be partitioned as

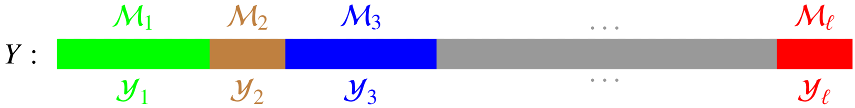
$$\mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_\ell \text{ and } \mathcal{Y}_1, \mathcal{Y}_2, \ldots, \mathcal{Y}_\ell$$

1. $\mathcal{M}_1 \Longrightarrow \mathcal{Y}_1$
2. $\mathcal{M}_1, \mathcal{M}_2 \Longrightarrow \mathcal{Y}_2$
3. $\vdots$
4. $\mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_\ell \Longrightarrow \mathcal{Y}_\ell$

Finding preimages of Condition function

$$Y = \text{Condition}_{\Delta}(M) = 0$$

If Condition function is complex enough $\Longrightarrow 2^{|Y|}$



If input and output can be partitioned as

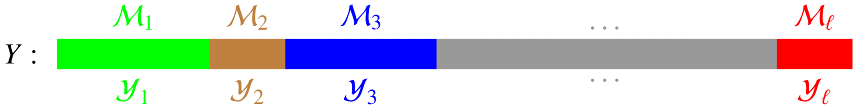
$$\mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_\ell \text{ and } \mathcal{Y}_1, \mathcal{Y}_2, \ldots, \mathcal{Y}_\ell$$

1. $\mathcal{M}_1 \Longrightarrow \mathcal{Y}_1$
2. $\mathcal{M}_1, \mathcal{M}_2 \Longrightarrow \mathcal{Y}_2$
3. $\vdots$
4. $\mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_\ell \Longrightarrow \mathcal{Y}_\ell$

Finding preimages of Condition function

$$Y = \text{Condition}_{\Delta}(M) = 0$$

If Condition function is complex enough $\Longrightarrow 2^{|Y|}$



If input and output can be partitioned as

$$\mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_\ell \text{ and } \mathcal{Y}_1, \mathcal{Y}_2, \ldots, \mathcal{Y}_\ell$$

1. $\mathcal{M}_1 \Longrightarrow \mathcal{Y}_1$
2. $\mathcal{M}_1, \mathcal{M}_2 \Longrightarrow \mathcal{Y}_2$
3. $\vdots$
4. $\mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_\ell \Longrightarrow \mathcal{Y}_\ell$

Complexity = $\sum_{i=0}^{\ell} 2^{q_i}$

$q_{i-1} = |\mathcal{Y}_{i-1}| + \max(0, q_i - |\mathcal{M}_i|)$

for $i = \ell, \ell - 1, \ldots, 1$ with $q_\ell = |\mathcal{Y}_\ell|$

Inpout/Output partitioning of a Condition function

Dependency table ...

Inpout/Output partitioning of a Condition function

0	0	0	0	0	1	0	1	1	1	0	1	1	1	0	0	1	0	1	1	1	1
1	0	0	1	0	0	0	0	1	0	1	0	1	1	0	0	0	0	0	0	0	0
0	0	0	0	1	1	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0
0	1	0	1	0	0	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0
0	1	0	1	1	1	1	0	1	1	0	0	1	1	0	0	0	0	0	0	0	0
0	0	1	0	1	1	1	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0
0	1	1	1	1	1	0	0	0	0	1	0	1	0	0	1	0	1	0	0	1	1
1	1	0	0	0	0	1	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	1	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0
1	0	1	0	0	1	0	0	1	1	1	1	0	0	1	0	1	1	0	1	0	1
0	0	0	1	1	0	1	0	1	1	0	0	0	1	0	0	1	1	0	1	0	1
1	1	1	1	1	1	0	0	1	1	0	1	0	0	1	1	1	0	0	0	0	1
0	0	1	0	1	1	1	1	0	0	0	0	1	1	0	0	0	0	0	0	0	0
1	1	1	1	0	0	1	0	1	0	1	1	1	0	0	0	0	0	1	0	0	1
0	0	0	0	0	1	0	0	1	0	0	0	0	1	1	0	1	1	0	1	0	1
1	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	0	0	0	1	1	1	1	1	1	1	0	1	0	0	0	0	0	0	0	0
1	1	0	0	0	1	0	0	0	1	0	1	1	1	0	0	0	0	0	0	0	0
0	0	1	0	0	0	1	1	0	1	0	1	1	1	0	0	0	0	0	0	0	0
0	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	1	0	1	0	1	1	1	1	0	0	0	1	0	0	0	0	0	0	0	1
0	0	0	1	1	1	0	0	0	1	0	0	0	1	1	1	1	1	0	0	1	1
1	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	0	0	1	1	1	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0
0	1	0	0	1	0	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0
1	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	0	1	1	0	1	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0
0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	0	1	1	0	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0
0	0	0	0	1	0	1	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0

Inpout/Output partitioning of a Condition function

0	0	0	0	0	1	0	1	1	1	0	1	1	1	0	0	1	0	1	1	1	1
1	0	0	1	0	0	0	0	1	0	1	0	1	1	0	0	0	0	0	0	0	0
0	0	0	0	1	1	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0
0	1	0	1	0	0	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0
0	1	0	1	1	1	1	0	1	1	0	0	1	1	0	0	0	0	0	0	0	0
0	0	1	0	1	1	1	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0
0	1	1	1	1	1	0	0	0	0	1	0	1	0	0	1	0	1	0	0	1	1
1	1	0	0	0	0	1	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	1	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0
1	0	1	0	0	1	0	0	1	1	1	1	0	0	1	0	1	1	0	1	0	1
0	0	0	1	1	0	1	0	1	1	0	0	0	1	0	0	1	1	0	1	0	1
1	1	1	1	1	1	0	0	1	1	0	1	0	0	1	1	1	0	0	0	0	1
0	0	1	0	1	1	1	1	0	0	0	0	1	1	0	0	0	0	0	0	0	0
1	1	1	1	0	0	1	0	1	0	1	1	1	0	0	0	0	0	1	0	0	1
0	0	0	0	0	1	0	0	1	0	0	0	0	1	1	0	1	1	0	1	0	1
1	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	0	0	0	1	1	1	1	1	1	1	0	1	0	0	0	0	0	0	0	0
1	1	0	0	0	1	0	0	0	1	0	1	1	1	0	0	0	0	0	0	0	0
0	0	1	0	0	0	1	1	0	1	0	1	1	1	0	0	0	0	0	0	0	0
0	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	1	0	1	0	1	1	1	1	0	0	0	1	0	0	0	0	0	0	0	1
0	0	0	1	1	1	0	0	0	1	0	0	0	1	1	1	1	1	0	0	1	1
1	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	0	0	1	1	1	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0
0	1	0	0	1	0	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0
1	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	0	1	1	0	1	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0
0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	0	1	1	0	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0
0	0	0	0	1	0	1	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0

Inpout/Output partitioning of a Condition function

0	0	0	0	0	1	0	1	1	1	0	1	1	1	0	0	1	0	1	1	1	1
1	0	0	1	0	0	0	0	1	0	1	0	1	1	0	0	0	0	0	0	0	0
0	0	0	0	1	1	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0
0	1	0	1	0	0	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0
0	1	0	1	1	1	1	0	1	1	0	0	1	1	0	0	0	0	0	0	0	0
0	0	1	0	1	1	1	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0
0	1	1	1	1	1	0	0	0	0	1	0	1	0	0	1	0	1	0	0	1	1
1	1	0	0	0	0	1	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	1	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0
1	0	1	0	0	1	0	0	1	1	1	1	0	0	1	0	1	1	0	1	0	1
0	0	0	1	1	0	1	0	1	1	0	0	0	1	0	0	1	1	0	1	0	1
1	1	1	1	1	1	0	0	1	1	0	1	0	0	1	1	1	0	0	0	0	1
0	0	1	0	1	1	1	1	0	0	0	0	1	1	0	0	0	0	0	0	0	0
1	1	1	1	0	0	1	0	1	0	1	1	1	0	0	0	0	0	1	0	0	1
0	0	0	0	0	1	0	0	1	0	0	0	0	1	1	0	1	1	0	1	0	1
1	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	0	0	0	1	1	1	1	1	1	1	0	1	0	0	0	0	0	0	0	0
1	1	0	0	0	1	0	0	0	1	0	1	1	1	0	0	0	0	0	0	0	0
0	0	1	0	0	0	1	1	0	1	0	1	1	1	0	0	0	0	0	0	0	0
0	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	1	0	1	0	1	1	1	1	0	0	0	1	0	0	0	0	0	0	0	1
0	0	0	1	1	1	0	0	0	1	0	0	0	1	1	1	1	1	0	0	1	1
1	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	0	0	1	1	1	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0
0	1	0	0	1	0	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0
1	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	0	1	1	0	1	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0
0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	0	1	1	0	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0
0	0	0	0	1	0	1	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0

Inpout/Output partitioning of a Condition function

0	0	0	0	0	1	0	1	1	1	0	1	1	1	0	0	1	0	1	1	1	1
1	0	0	1	0	0	0	0	1	0	1	0	1	1	0	0	0	0	0	0	0	0
0	0	0	0	1	1	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0
0	1	0	1	0	0	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0
0	1	0	1	1	1	1	0	1	1	0	0	1	1	0	0	0	0	0	0	0	0
0	0	1	0	1	1	1	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0
0	1	1	1	1	1	0	0	0	0	1	0	1	0	0	1	0	1	0	0	1	1
1	1	0	0	0	0	1	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	1	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0
1	0	1	0	0	1	0	0	1	1	1	1	0	0	1	0	1	1	0	1	0	1
0	0	0	1	1	0	1	0	1	1	0	0	0	1	0	0	1	1	0	1	0	1
1	1	1	1	1	1	0	0	1	1	0	1	0	0	1	1	1	0	0	0	0	1
0	0	1	0	1	1	1	1	0	0	0	0	1	1	0	0	0	0	0	0	0	0
1	1	1	1	0	0	1	0	1	0	1	1	1	0	0	0	0	0	1	0	0	1
0	0	0	0	0	1	0	0	1	0	0	0	0	1	1	0	1	1	0	1	0	1
1	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	0	0	0	1	1	1	1	1	1	1	0	1	0	0	0	0	0	0	0	0
1	1	0	0	0	1	0	0	0	1	0	1	1	1	0	0	0	0	0	0	0	0
0	0	1	0	0	0	1	1	0	1	0	1	1	1	0	0	0	0	0	0	0	0
0	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	1	0	1	0	1	1	1	1	0	0	0	1	0	0	0	0	0	0	0	1
0	0	0	1	1	1	0	0	0	1	0	0	0	1	1	1	1	1	0	0	1	1
1	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	0	0	1	1	1	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0
0	1	0	0	1	0	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0
1	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	0	1	1	0	1	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0
0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	0	1	1	0	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0
0	0	0	0	1	0	1	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0

Inpout/Output partitioning of a Condition function

0	0	0	0	0	1	0	1	1	1	0	1	1	1	0	0	1	0	1	1	1	1
1	0	0	1	0	0	0	0	1	0	1	0	1	1	0	0	0	0	0	0	0	0
0	0	0	0	1	1	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0
0	1	0	1	0	0	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0
0	1	0	1	1	1	1	0	1	1	0	0	1	1	0	0	0	0	0	0	0	0
0	0	1	0	1	1	1	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0
0	1	1	1	1	1	0	0	0	0	1	0	1	0	0	1	0	1	0	0	1	1
1	1	0	0	0	0	1	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	1	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0
1	0	1	0	0	1	0	0	1	1	1	1	0	0	1	0	1	1	0	1	0	1
0	0	0	1	1	0	1	0	1	1	0	0	0	1	0	0	1	1	0	1	0	1
1	1	1	1	1	1	0	0	1	1	0	1	0	0	1	1	1	0	0	0	0	1
0	0	1	0	1	1	1	1	0	0	0	0	1	1	0	0	0	0	0	0	0	0
1	1	1	1	0	0	1	0	1	0	1	1	1	0	0	0	0	0	1	0	0	1
0	0	0	0	0	1	0	0	1	0	0	0	0	1	1	0	1	1	0	1	0	1
1	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	0	0	0	1	1	1	1	1	1	1	0	1	0	0	0	0	0	0	0	0
1	1	0	0	0	1	0	0	0	1	0	1	1	1	0	0	0	0	0	0	0	0
0	0	1	0	0	0	1	1	0	1	0	1	1	1	0	0	0	0	0	0	0	0
0	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	1	0	1	0	1	1	1	1	0	0	0	1	0	0	0	0	0	0	0	1
0	0	0	1	1	1	0	0	0	1	0	0	0	1	1	1	1	1	0	0	1	1
1	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	0	0	1	1	1	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0
0	1	0	0	1	0	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0
1	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	0	1	1	0	1	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0
0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	0	1	1	0	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0
0	0	0	0	1	0	1	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0

Inpout/Output partitioning of a Condition function

0	0	0	0	0	1	0	1	1	1	0	1	1	1	0	0	1	0	1	1	1	1
1	0	0	1	0	0	0	0	1	0	1	0	1	1	0	0	0	0	0	0	0	0
0	0	0	0	1	1	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0
0	1	0	1	0	0	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0
0	1	0	1	1	1	1	0	1	1	0	0	1	1	0	0	0	0	0	0	0	0
0	0	1	0	1	1	1	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0
0	1	1	1	1	1	0	0	0	0	1	0	1	0	0	1	0	1	0	0	1	1
1	1	0	0	0	0	1	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	1	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0
1	0	1	0	0	1	0	0	1	1	1	1	0	0	1	0	1	1	0	1	0	1
0	0																				1
1	1																				1
0	0																				0
1	1																				1
0	0																				1
1	1																				0
1	1																				0
1	1																				0
0	0																				0
0	1																				0
1	0																				1
0	0																				1
1	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	0	0	1	1	1	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0
0	1	0	0	1	0	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0
1	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	0	1	1	0	1	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0
0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	0	1	1	0	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0
0	0	0	0	1	0	1	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0

$\mathcal{Y}_1 = \{16, 20, 23, 27, 29, 30\}$ $\quad \mathcal{M}_1 = \{1, 2, 3, 4, 5\}$

$\mathcal{Y}_2 = \{3, 4, 6, 8, 9, 24, 25, 26, 28, 32\}$ $\quad \mathcal{M}_2 = \{6, 7, 8, 9, 10, 11\}$

$\mathcal{Y}_3 = \{2, 5, 12, 13, 15, 17, 18, 19, 31\}$ $\quad \mathcal{M}_3 = \{12, 13, 14\}$

$\mathcal{Y}_4 = \{1, 7, 10, 11, 14, 15, 21, 22\}$ $\quad \mathcal{M}_4 = \{15, 16, 17, 18, 19, 20, 21, 22\}$

CubeHash: differential paths probability

Table: $\log_2$ probability of the best path

$r \setminus b$	1	2	3	4	8	12	16	32	48	64
1	1225	221	46	32	32	–	–	–	–	–
2	1225	221	46	32	32	–	–	–	–	–
3	4238	1881	798	478	478	400	400	400	364	65
4	2614	964	195	189	189	156	156	156	130	130
5	10221	4579	2433	1517	1517	1244	1244	1244	1244	205
6	4238	1881	798	478	478	400	400	400	351	351
7	13365	5820	3028	2124	2124	1748	1748	1748	1748	447
8	2614	2614	1022	1009	1009	830	830	830	637	637

- $\Pr > 2^{-512} \Rightarrow$ second pre-image attack
- $\Pr > 2^{-256} \Rightarrow$ trivial collision attack

CubeHash: Collision attacks complexity

Table: $\log_2$ complexities of the best attack

$r \setminus b$	1	2	3	4	8	12	16	32	48	64
1	1121.0	135.1	24.0	15.0	7.6	–	–	–	–	–
2	1177.0	179.1	27.0	17.0	7.9	–	–	–	–	–
3	4214.0	1793.0	720.0	380.1	292.6	153.5	102.0	55.6	53.3	9.4
4	2598.0	924.0	163.0	138.4	105.3	67.5	60.7	54.7	30.7	28.8
5	10085.0	4460.0	2345.0	1397.0	1286.0	946.0	868.0	588.2	425.0	71.7
6	4230.0	1841.0	760.6	422.1	374.4	260.4	222.6	182.1	147.7	144.0
7	13261.0	5709.0	2940.0	2004.0	1892.0	1423.0	1323.0	978.0	706.0	203.0
8	2606.0	2590.0	982.0	953.0	889.0	699.0	662.0	524.3	313.0	304.4

Our results on MD6

We found a differential path with probability 2^{-90} for 16-round.

The 90 condition bits can be fulfilled in 2^{30} rather than 2^{90}.

r	16	17	18	19
	collision	near-collision (63)	near-collision (144)	near-collision (270)

The output of the compression function is 1024-bit long.

The proposed number of rounds varies from $r = 80$ to $r = 168$ for digests of size 160 to 512 bits.

Conclusions

- Have introduced the concept of Condition function
- Useful when combined with linear differential cryptanalysis
- Applied to reduced-round variants of CubeHash and MD6

Thank you!