Cryptanalysis of the LAKE Hash Family

A. Biryukov[1], P. Gauravaram[2], Jian Guo[3], D. Khovratovich[1], S. Ling[3], K. Matusiewicz[2], Ivica Nikolic[1], J. Pieprzyk[4] and H. Wang[3]

[1]University of Luxembourg, Luxembourg
[2]Technical University of Denmark, Denmark
[3]Nanyang Technological University, Singapore
[4]Macquaire University, Australia

FSE2009, 23 Feb 2009

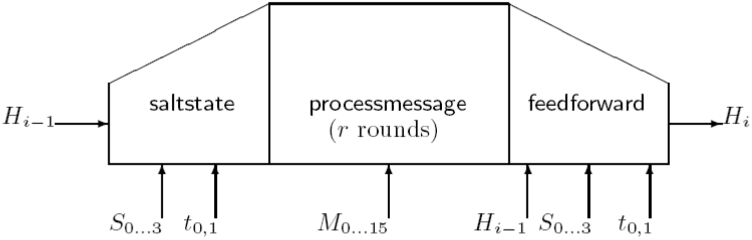
Overview

Outline

Introduction to LAKE

- Designed by Jean-Philippe Aumasson, Willi Meier, and Raphael C.-W. Pahan (FSE2008)
- Two main instances: LAKE-256 and LAKE-512
- Follows HAIFA structure, i.e. there are four inputs for compression function: chaining value H, message block M, salt S and block index t - number of bits/byte/blocks processed so far.
- Internal wide-pipe
- Two main internal functions: f, g
- Related work: Mendel and Schläffer find collisions for 4 out of 8 rounds with complexity 2^{109} (in ACISP2008).

LAKE compression function



- **SaltState**: Expand the chaining value from n bits to $2n$ bits
- **ProcessMessage**: 8 rounds for LAKE-256 and 10 rounds for LAKE-512
- **FeedForward**: Shrink back to n bits.

SaltState

$$
\begin{array}{ll}
F_0 = H_0 & F_8 = g(H_0, S_0 \oplus t_0, C_8, 0) \\
F_1 = H_1 & F_9 = g(H_1, S_1 \oplus t_1, C_9, 0) \\
F_2 = H_2 & F_{10} = g(H_2, S_2, C_{10}, 0) \\
F_3 = H_3 & F_{11} = g(H_3, S_3, C_{11}, 0) \\
F_4 = H_4 & F_{12} = g(H_4, S_0, C_{12}, 0) \\
F_5 = H_5 & F_{13} = g(H_5, S_1, C_{13}, 0) \\
F_6 = H_6 & F_{14} = g(H_6, S_2, C_{14}, 0) \\
F_7 = H_7 & F_{15} = g(H_7, S_3, C_{15}, 0)
\end{array}
$$

$$
g(a, b, c, d) = ((a + b) \ggg 1) \oplus (c + d)
$$

ProcessMessage

r-th round:

$$
\begin{aligned}
L_0 &= f(F_{15}, F_0, M_{\sigma_r(0)}, C_0) \\
L_1 &= f(L_0, F_1, M_{\sigma_r(1)}, C_1) \\
L_2 &= f(L_1, F_2, M_{\sigma_r(2)}, C_2) \\
&\dots \\
L_{15} &= f(L_{14}, F_{15}, M_{\sigma_r(15)}, C_{15})
\end{aligned}
$$

$$
\begin{aligned}
F_0 &= g(L_{15}, L_0, F_0, L_1) \\
F_1 &= g(F_0, L_1, F_1, L_2) \\
F_2 &= g(F_1, L_2, F_2, L_3) \\
&\dots \\
F_{15} &= g(F_{14}, L_{15}, F_{15}, L_0)
\end{aligned}
$$

$$f(a, b, c, d) = a + b \vee C_0 + (c + (a \wedge C_1)) \ggg 7 + (b + (c \oplus d)) \ggg 13$$

FeedForward

$$
\begin{aligned}
H_0 &= f(F_0, F_8, S_0 \oplus t_0, H_0) \\
H_1 &= f(F_1, F_9, S_1 \oplus t_1, H_1) \\
H_2 &= f(F_2, F_{10}, S_2, H_2) \\
H_3 &= f(F_3, F_{11}, S_3, H_3) \\
H_4 &= f(F_4, F_{12}, S_0, H_4) \\
H_5 &= f(F_5, F_{13}, S_1, H_5) \\
H_6 &= f(F_6, F_{14}, S_2, H_6) \\
H_7 &= f(F_7, F_{15}, S_3, H_7)
\end{aligned}
$$

Our work

We show:

- (H, S)-type collision
- (H, t)-type collision
- H-type near-collision

Key Observations

- f is not injective with respect to a, b, c $\Rightarrow$ internal collisions.

  $$f(a,b,c,d) = a + b \vee C_0 + ((c + (a \wedge C_1)) \ggg 7) + ((b + (c \oplus d)) \ggg 13)$$

- (F, F') is collision for first round **ProcessMessage** $\Rightarrow$ collision for all rounds.

Outline

(H, S)-type collision: procedure and technique

1. Find proper (F, F') - collision for **ProcessMessage**: Modeling and high-level differential finding
2. Solve **SaltState** and **FeedForward**.
3. Combine and reduce complexity.

(H,S)-type collision: the differential

SaltState

$\Delta F_0 \leftarrow \Delta H_0$
$\Delta F_1 \leftarrow \Delta H_1$
$\Delta F_2 \leftarrow \Delta H_2$
$F_3 \leftarrow H_3$
$\Delta F_4 \leftarrow \Delta H_4$
$\Delta F_5 \leftarrow \Delta H_5$
$\Delta F_6 \leftarrow \Delta H_6$
$F_7 \leftarrow H_7$
$F_8 \leftarrow g(\Delta H_0, \Delta S_0 \oplus t_0, C_8, 0)$
$F_9 \leftarrow g(\Delta H_1, \Delta S_1 \oplus t_1, C_9, 0)$
$F_{10} \leftarrow g(\Delta H_2, \Delta S_2, C_{10}, 0)$
$F_{11} \leftarrow g(H_3, S_3, C_{11}, 0)$
$F_{12} \leftarrow g(\Delta H_4, \Delta S_0, C_{12}, 0)$
$F_{13} \leftarrow g(\Delta H_5, \Delta S_1, C_{13}, 0)$
$F_{14} \leftarrow g(\Delta H_6, \Delta S_2, C_{14}, 0)$
$F_{15} \leftarrow g(H_7, S_3, C_{15}, 0)$

FeedForward

$H_0 \leftarrow f(R_0, R_8, \Delta S_0 \oplus t_0, \Delta H_0)$
$H_1 \leftarrow f(R_1, R_9, \Delta S_1 \oplus t_1, \Delta H_1)$
$H_2 \leftarrow f(R_2, R_{10}, \Delta S_2, \Delta H_2)$
$H_3 \leftarrow f(R_3, R_{11}, S_3, H_3)$
$H_4 \leftarrow f(R_4, R_{12}, \Delta S_0, \Delta H_4)$
$H_5 \leftarrow f(R_5, R_{13}, \Delta S_1, \Delta H_5)$
$H_6 \leftarrow f(R_6, R_{14}, \Delta S_2, \Delta H_6)$
$H_7 \leftarrow f(R_7, R_{15}, S_3, H_7)$

ProcessMessage

$L_0 \leftarrow f(F_{15}, \Delta F_0, M_{\sigma(0)}, C_0)$
$\Delta L_1 \leftarrow f(L_0, \Delta F_1, M_{\sigma(1)}, C_1)$
$\Delta L_2 \leftarrow f(\Delta L_1, \Delta F_2, M_{\sigma(2)}, C_2)$
$L_3 \leftarrow f(\Delta L_2, F_3, M_{\sigma(3)}, C_3)$
$L_4 \leftarrow f(L_3, \Delta F_4, M_{\sigma(4)}, C_4)$
$\Delta L_5 \leftarrow f(L_4, \Delta F_5, M_{\sigma(5)}, C_5)$
$\Delta L_6 \leftarrow f(\Delta L_5, \Delta F_6, M_{\sigma(6)}, C_6)$
$L_7 \leftarrow f(\Delta L_6, F_7, M_{\sigma(7)}, C_7)$
$L_8 \leftarrow f(L_7, F_8, M_{\sigma(8)}, C_8)$
$\ldots$
$L_{15} \leftarrow f(L_{14}, F_{15}, M_{\sigma(15)}, C_{15})$

$W_0 \leftarrow g(L_{15}, L_0, \Delta F_0, \Delta L_1)$
$W_1 \leftarrow g(W_0, \Delta L_1, \Delta F_1, \Delta L_2)$
$W_2 \leftarrow g(W_1, \Delta L_2, \Delta F_2, L_3)$
$W_3 \leftarrow g(W_2, L_3, F_3, L_4)$
$W_4 \leftarrow g(W_3, L_4, \Delta F_4, \Delta L_5)$
$W_5 \leftarrow g(W_4, \Delta L_5, \Delta F_5, \Delta L_6)$
$W_6 \leftarrow g(W_5, \Delta L_6, \Delta F_6, L_7)$
$W_7 \leftarrow g(W_6, L_7, F_7, L_8)$
$\ldots$
$W_{15} \leftarrow g(W_{14}, L_{15}, F_{15}, W_0)$

(H, S)-type collision: Algorithm and Complexity

Algorithm 1 Find solutions for ProcessMessage

```
Randomly pick (L_2, L'_2) ∈ S_fa
repeat
    Randomly pick F_1, compute F'_1 = −1 − ΔL_2 − F_1
until f_b(F_1) − f_b(F'_1) ∈ S^A_{fb_odd}
repeat
    Randomly pick L_1, F_2
    Compute L'_1 = f_b(F'_1) − f_b(F_1) + L_1
    Compute F'_2 so that f_b(F'_2) = ΔL_2 + f_a(L_1) − f_a(L'_1) + f_b(F_2)
until p11 is fulfilled
Pick (F_0, F'_0) ∈ S_fb so that ΔF_0 + ΔL_1 = 0
```

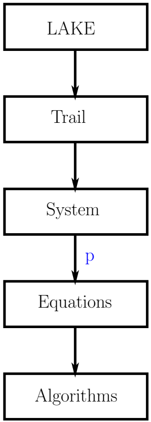
1. Solving the ProcessMessage: 2^{30}
2. Solving SaltState and FeedForward: 2^{24}

Overall complexity: 2^{54}. Choosing (L_2, L'_2) carefully reduces to 2^{42}

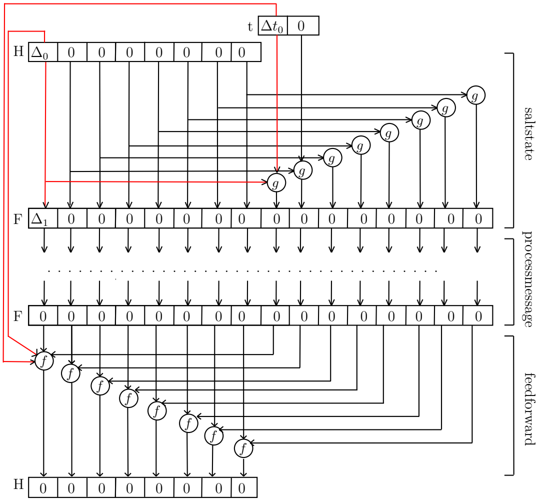
Outline

(H, t) collision - General strategy

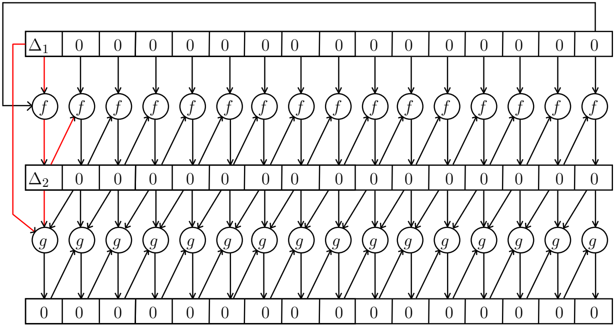
1. Build a differential trail for the whole CF
2. Rewrite the trail as a system of equations
3. Reduce the system to simple equations
4. Call algorithms that solve the equations



(H, t) collision - Differential trail



(H, t) collision - Differential trail

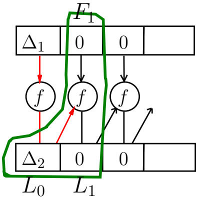


The trail is valid !!!

(H, t) collision - Getting a system

Write only the equations with non zero input to f and g

Example:



$$\Delta L_1 = f(L_0^{'}, F_1, M_1, C_1) - f(L_0, F_1, M_1, C_1) = 0$$

Equation:

$$L_0^{'} - L_0 + [M_1 + (L_0^{'} \wedge C_1)] \ggg 7 - [M_1 + (L_0 \wedge C_1)] \ggg 7 = 0$$

The system has 5 equations

(H, t) collision - Reducing the system

- Expand $(x + y) \lll c = x \lll c + y \lll c$ (with some probability)
- Use some other tricks

Example:

$$L_0^{'} - L_0 + [M_1 + (L_0^{'} \wedge C_1)] \ggg 7 - [M_1 + (L_0 \wedge C_1)] \ggg 7 = 0$$
$$L_0^{'} - L_0 + (L_0^{'} \wedge C_1) \ggg 7 - (L_0 \wedge C_1) \ggg 7 = 0$$

Let $L_0^{'} - L_0 = R$

$$(X + A) \wedge C = X \wedge C + B$$

where $X = L_0, A = R, B = (-R) \ll 7, C = C_1$.

(H, t) collision - Algorithms

The whole system is reduced to equations of type:

$$(X + A) \wedge C = X \wedge C + B$$
$$(X + A) \vee C = X \vee C + B$$
$$(X + A) \oplus C_1 = X \oplus C_2 + B$$
$$((X + A) \oplus X) \ggg 1 = (Y + B) \oplus Y$$

Lemma

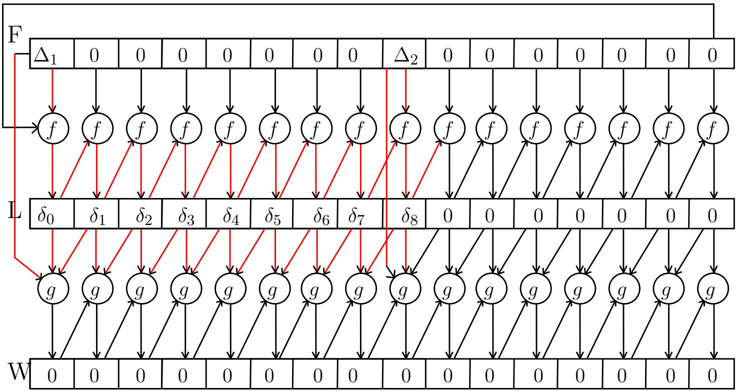
Exist efficient algorithms that solve these equations.

Outline

(H) near-collision - Strategy

- Build a trail only for the first two procedures of the compression function (near collision)
- Get a system from the trail
- Find an equation for the system by a “smart” bruteforce

(H) near-collision - Differential trail



The trail for ProcessMessage is valid !!!

(H) near-collision - System and solutions

- Rewrite the trail as a system of equations. The system has around 20 equations
- Simplify some of the equations
- Step-by-step bruteforce. Once a solution for an equation is found it (almost) does not effect the other equations

Complexities

- (H, t) collision - 2^{33} CF calls
  Practical, collision example in the Appendix
- (H) near-collision - 2^{99} CF calls
  Built a partially solved system

Outline

Conclusions

- Non-injective functions should be used carefully
- Caution on the two additional inputs: salt and block index
- The attacks are NOT applicable to BLAKE