Cryptanalysis of Full RIPEMD-128

Franck Landelle and **Thomas Peyrin**

DGA MI - France   NTU - Singapore

Eurocrypt 2013

Athens, Greece - May 28 , 2013

DGA

NANYANG TECHNOLOGICAL UNIVERSITY

Motivations to study RIPEMD−128

- MDx-like hash function is a very frequent design :

1990’	MD-X (MD4,MD5,SHA-1,HAVAL,RIPEMD)
2002	SHA-2 (SHA-224, . . . , SHA-512)

- Some old hash functions are still unbroken :

Broken	MD4,MD5,RIPEMD-0
Broken	HAVAL
Broken	SHA-1
Unbroken	RIPEMD-128, RIPEMD-160
Unbroken	SHA-2

- RIPEMD-128

Design	15 years old.
unbroken	9 years after Wang’s attacks [WLF$^+$05].

General design and Security notions

- A hash function $\mathcal{H}$ is often defined by repeated applications of a compression function h.
- A collision on the hash function $\mathcal{H}$ always comes from a collision on the compression function h:

$$\mathcal{H}(M) = \mathcal{H}(M^*) \Longrightarrow h(cv, m) = h(cv^*, m^*)$$

The conditions on cv and m give different kind of attacks :

Collision $cv = cv^*$ fixed and $m \neq m^*$ free.

Semi-free-start Collision $cv = cv^*$ and $m \neq m^*$ are free.

Free-start Collision $(cv, m) \neq (cv^*, m^*)$ are free.

General design and Security notions

- A hash function $\mathcal{H}$ is often defined by repeated applications of a compression function h.
- A collision on the hash function $\mathcal{H}$ always comes from a collision on the compression function h:

$$\mathcal{H}(M) = \mathcal{H}(M^*) \Longrightarrow h(cv, m) = h(cv^*, m^*)$$

The conditions on cv and m give different kind of attacks :

Collision $cv = cv^*$ fixed and $m \neq m^*$ free.

Semi-free-start Collision $cv = cv^*$ and $m \neq m^*$ are free.

Free-start Collision $(cv, m) \neq (cv^*, m^*)$ are free.

Results on RIPEMD-128 compression function

RIPEMD-128 parameters :

- **Digest** 128 bits
- **Steps** 64 steps.

Known and new results on RIPEMD-128 compression function:

Target	#Steps	Complexity	Ref.
collision	48	2^{40}	[MNS12]
collision	**60**	$2^{57.57}$	**new**
collision	**63**	$2^{59.91}$	**new**
collision	**Full**	$2^{61.57}$	**new**
non-randomness	52	2^{107}	[SW12]
non-randomness	**Full**	$2^{59.57}$	**new**

Results on RIPEMD-128 compression function

RIPEMD-128 parameters :

- Digest: 128 bits
- Steps: 64 steps.

Known and new results on RIPEMD-128 compression function:

Target	#Steps	Complexity	Ref.
collision	48	2^{40}	[MNS12]
collision	60	$2^{57.57}$	new
collision	63	$2^{59.91}$	new
collision	**Full**	$2^{61.57}$	**new**
non-randomness	52	2^{107}	[SW12]
non-randomness	Full	$2^{59.57}$	new

Results on RIPEMD-128 compression function

RIPEMD-128 parameters :

- **Digest** 128 bits
- **Steps** 64 steps.

Known and new results on RIPEMD-128 compression function:

Target	#Steps	Complexity	Ref.
collision	48	2^{40}	[MNS12]
collision	**60**	$2^{57.57}$	**new**
collision	**63**	$2^{59.91}$	**new**
collision	**Full**	$2^{61.57}$	**new**
non-randomness	52	2^{107}	[SW12]
non-randomness	**Full**	$2^{59.57}$	**new**

In the talk

Function RIPEMD-128 compression function

Attack a semi-free-start collision

$$\text{Find } cv, m \neq m^* \,/\, h(cv, m) = h(cv, m^*).$$

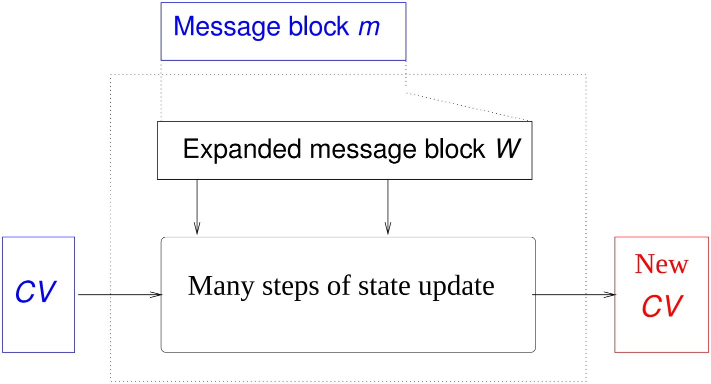
Strategy

- Choose a message difference $\delta_m = m \oplus m^*$
- Find a differential path on all intermediate state variables
- Find conforming cv and m

Outline

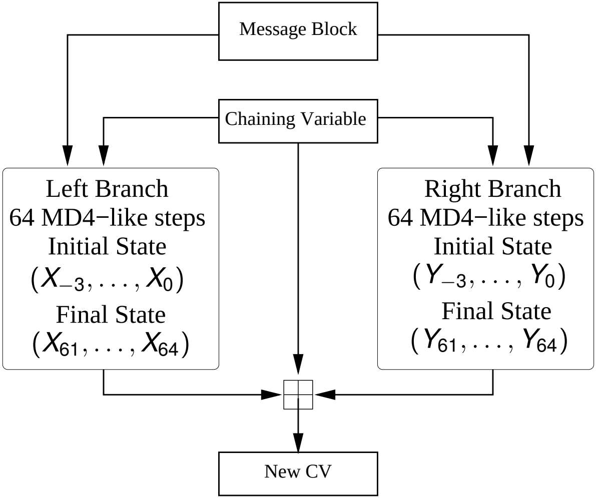
1. Description of RIPEMD-128
2. Finding a differential path
   - Finding a message difference
   - Finding the non linear part
3. Finding a conforming pair
   - Generating a starting point
   - Merging the 2 branches

A compression function



Compression Function

Overview of RIPEMD-128 compression function



The step functions

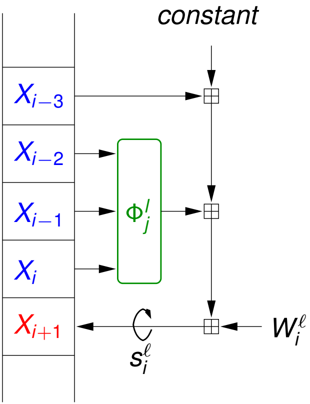


Figure: Left Branch

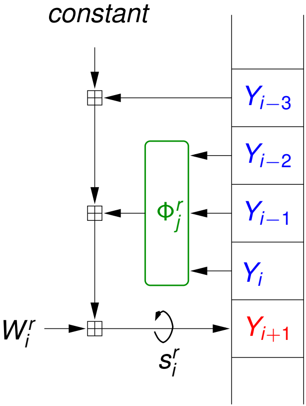


Figure: Right Branch

The boolean functions

Boolean functions in RIPEMD−128:

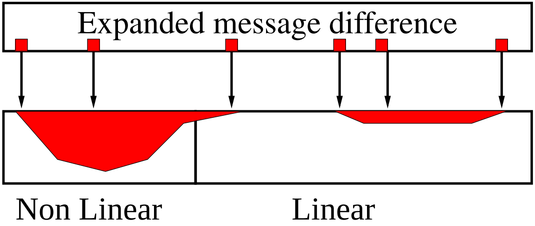
- $\mathrm{XOR}(x, y, z) := x \oplus y \oplus z$,
- $\mathrm{IF}(x, y, z) := x \wedge y \oplus \bar{x} \wedge z$
- $\mathrm{ONX}(x, y, z) := (x \vee \bar{y}) \oplus z$

Steps i	Round j	$\Phi_j^\ell(x, y, z)$	$\Phi_j^r(x, y, z)$
0 to 15	0	$\mathtt{XOR}(x, y, z)$	$\mathtt{IF}(z, x, y)$
16 to 31	1	$\mathtt{IF}(x, y, z)$	$\mathtt{ONX}(x, y, z)$
32 to 47	2	$\mathtt{ONX}(x, y, z)$	$\mathtt{IF}(x, y, z)$
48 to 63	3	$\mathtt{IF}(z, x, y)$	$\mathtt{XOR}(x, y, z)$

Outline

The classical strategy

1. Find a message difference δ_m and a differential path with high probability on the middle and last steps (ideally after the first round).
2. Find a “realistic” non linear differential path on the first steps (ideally on the first round).
3. Find a chaining variable *cv* and a message *m* such that the state differential path is followed.



What is the shape of the differential path ?

Input of a function can help to control the differential propagation.

Properties of the boolean functions:

- XOR : no control of differential propagation
- ONX: some control of differential propagation.
- IF : a good control of differential propagation and permits low diffusion.

Steps i	Round j	$\Phi_j^l(x,y,z)$	$\Phi_j^r(x,y,z)$
0 to 15	0	XOR(x,y,z)	IF(z,x,y)
16 to 31	1	IF(x,y,z)	ONX(x,y,z)
32 to 47	2	ONX(x,y,z)	IF(x,y,z)
48 to 63	3	IF(z,x,y)	XOR(x,y,z)

What is the shape of the differential path ?

Input of a function can help to control the differential propagation.

Properties of the boolean functions:

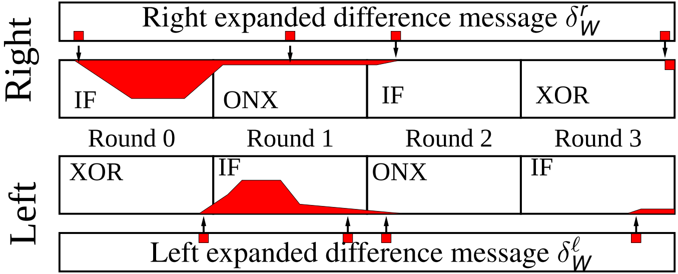
- XOR : no control of differential propagation
- ONX: some control of differential propagation.
- IF : a good control of differential propagation and permits low diffusion.

Steps i	Round j	$\Phi^l_j(x, y, z)$	$\Phi^r_j(x, y, z)$
0 to 15	0	$\mathtt{XOR}(x, y, z)$	$\mathtt{IF}(z, x, y)$
16 to 31	1	$\mathtt{IF}(x, y, z)$	$\mathtt{ONX}(x, y, z)$
32 to 47	2	$\mathtt{ONX}(x, y, z)$	$\mathtt{IF}(x, y, z)$
48 to 63	3	$\mathtt{IF}(z, x, y)$	$\mathtt{XOR}(x, y, z)$

Choose the message block difference:

Goals keep low hamming weight on the expanded message block

Choice Put a difference on a single word of message



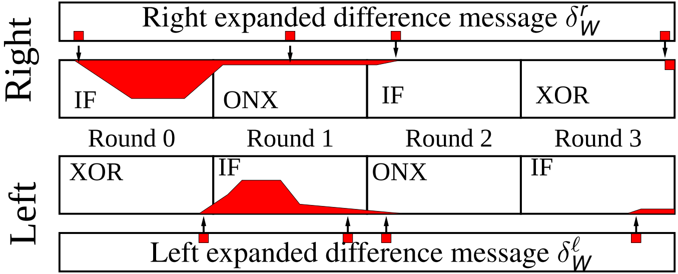
With the message block difference on m_{14}:

- “no difference” on rounds with XOR function.
- Non linear differential paths are in the round with IF

Choose the message block difference:

Goals keep low hamming weight on the expanded message block

Choice Put a difference on a single word of message



With the message block difference on m_{14}:

- "no difference" on rounds with XOR function.
- Non linear differential paths are in the round with IF

Outline

Automatic tool on generalised conditions

We implemented a tool similar to [CR06] for SHA-1 that used generalised conditions.

Hexa	(b, b^*) Notation	$(0,0)$	$(1,0)$	$(0,1)$	$(1,1)$
0xF	?	✓	✓	✓	✓
0x9	-	✓			✓
0x6	x		✓	✓	
0x1	0	✓			
0x2	u		✓		
0x4	n			✓	
0x8	1				✓

Where

- b: a bit during the treatment the message m
- b^*: the same bit for the second message m^*.

Left branch

```
Step               Xi                                  Wi                Πi
13: -------------------------------- | -------------------------------- 13
14: -------------------------------- | x------------------------------- 14
15: ???????????????????????????????? | -------------------------------- 15
16: ???????????????????????????????? | --------------------------------  7
17: ???????????????????????????????? | --------------------------------  4
18: ???????????????????????????????? | -------------------------------- 13
19: ???????????????????????????????? | --------------------------------  1
20: ???????????????????????????????? | -------------------------------- 10
21: ???????????????????????????????? | --------------------------------  6
22: ???????????????????????????????? | -------------------------------- 15
23: ???????????????????????????????? | --------------------------------  3
24: ???????????????????????????????? | -------------------------------- 12
25: ???????????????????????????????? | --------------------------------  0
26: -------u------------------------ | --------------------------------  9
27: 1------0-----u------------------ | --------------------------------  5
28: 0------1-----0------------------ | --------------------------------  2
29: n------------1------------------ | x------------------------------- 14
30: u------------------------------- | -------------------------------- 11
31: u------------------------------- | --------------------------------  8
32: 1------------------------------- | --------------------------------  3
33: -------------------------------- | -------------------------------- 10
34: -------------------------------- | x------------------------------- 14
35: -------------------------------- | --------------------------------  4
```

Left branch

```
Step              Xi                                    Wi                Πi
13: -------------------------------- | -------------------------------- 13
14: -------------------------------- | x------------------------------- 14
15: -----------------------n-------- | -------------------------------- 15
16: -----------unnnn-------0-------- | --------------------------------  7
17: -------n---00000-------1-------- | --1-----------------------------  4
18: -------0---01111---------------- | -------------------------------- 13
19: ---u---1-------n------------1--- | --------------------------------  1
20: ---0-----------0------------0--- | -------------------------------- 10
21: ---1-----------1------------n--- | --------------------------------  6
22: ---------------unnnn--------0--- | -------------------------------- 15
23: ---------------00000--------u--- | --------------------------------  3
24: -------------n-11101--------1--- | -------------------------------- 12
25: -----------n-0--------------1--- | --------------------------------  0
26: -------u---0-1------------------ | --------------------------------  9
27: 1------0---1-u------------------ | --------------------------------  5
28: 0------1-----0------------------ | --------------------------------  2
29: n------------1------------------ | x------------------------------- 14
30: u------------------------------- | -------------------------------- 11
31: u------------------------------- | --------------------------------  8
32: 1------------------------------- | --------------------------------  3
33: -------------------------------- | -------------------------------- 10
34: -------------------------------- | x------------------------------- 14
35: -------------------------------- | --1-----------------------------  4
```

Right branch

```
Step            Yi                                                  Wi                        πi
   : --------------------------------
   : --------------------------------
   : --------------------------------
   : --------------------------------  |  --------------------------------   5
01: --------------------------------  |  x-------------------------------  14
02: ????????????????????????????????  |  --------------------------------   7
03: ????????????????????????????????  |  --------------------------------   0
04: ????????????????????????????????  |  --------------------------------   9
05: ????????????????????????????????  |  --------------------------------   2
06: ????????????????????????????????  |  --------------------------------  11
07: ????????????????????????????????  |  --------------------------------   4
08: ????????????????????????????????  |  --------------------------------  13
09: ????????????????????????????????  |  --------------------------------   6
10: ????????????????????????????????  |  --------------------------------  15
11: ????????????????????????????????  |  --------------------------------   8
12: ????????????????????????????????  |  --------------------------------   1
13: ????????????????????????????????  |  --------------------------------  10
14: ????????????????????????????????  |  --------------------------------   3
15: -------u------------------------  |  --------------------------------  12
16: -------u----u-------------------  |  --------------------------------   6
17: -----u-0----u-------------------  |  --------------------------------  11
18: -----u------0-------------------  |  --------------------------------   3
19: 0----0--------------------------  |  --------------------------------   7
20: u-------------------------------  |  --------------------------------   0
```

Right branch

```
Step            Yi                                        Wi                         πi
   : --------------------------------
   : --------------------------------
   : --------------------------------
   : -----------------------0-------- | --------------------------------  5
01:  -----------------------1-------- | x-------------------------------  14
02:  -----------------------n-------- | --------------------------------  7
03:  -------------------------------- | --------------------------------  0
04:  --0000000----------------------- | --------------------------------  9
05:  --1111111----------------------- | --------------------------------  2
06:  --nuuuuuu----------------------- | --------------------------------  11
07:  --01-----------------------0-000 | --1-----------------------------  4
08:  -01------------------------0-011 | --------------------------------  13
09:  -1----------------10-0-----n-nnn | --------------------------------  6
10:  1n010000----------11-1---------- | --------------------------------  15
11:  00111111-----00--0nu-n---------- | --------------------------------  8
12:  nuuuuuuu-----11--11--0---------- | --------------------------------  1
13:  -------1-----nn--un--u---------- | --------------------------------  10
14:  -------1----01----u------------- | --------------------------------  3
15:  -------u----10----0------------- | --------------------------------  12
16:  -----0-u----u------------------- | --------------------------------  6
17:  -----u-0----u------------------- | --------------------------------  11
18:  -----u------0------------------- | --------------------------------  3
19:  0----0-------------------------- | --------------------------------  7
20:  u------------------------------- | --------------------------------  0
```

Outline

1. Description of RIPEMD-128

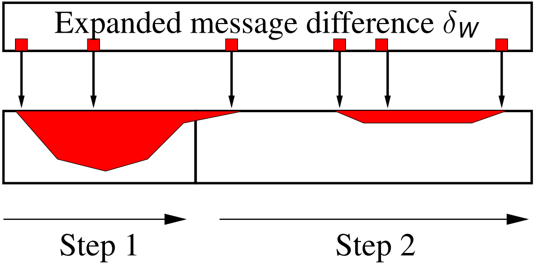
2. Finding a differential path
   - Finding a message difference
   - Finding the non linear part

3. Finding a conforming pair
   - Generating a starting point
   - Merging the 2 branches

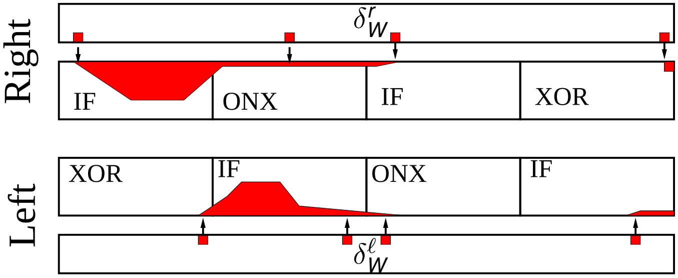
Following a classical differential path

The collision search is composed of two subparts:

- **step 1** handling the low-probability non-linear parts using the message block freedom
- **step 2** the remaining steps in both branches are verified probabilistically



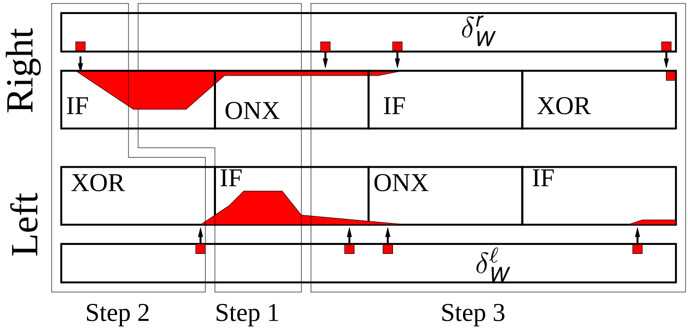
Finding a conforming pair



The collision search is composed of three subparts:

- step 1 Satisfying the Non Linear part of both branches
- step 2 Merging the two branches using some remaining free message words
- step 3 Handling probabilistically the linear differential path in both branches

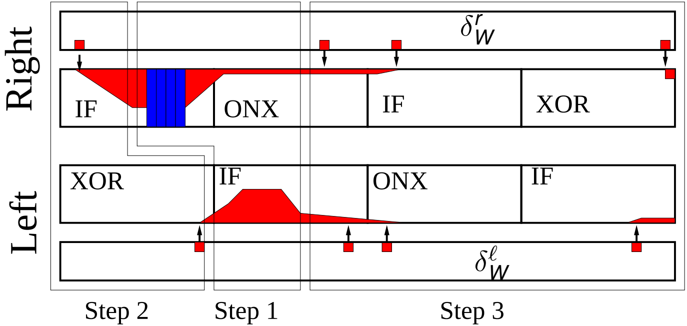
Finding a conforming pair



The collision search is composed of three subparts:

- step 1 Satisfying the Non Linear part of both branches
- step 2 Merging the two branches using some remaining free message words
- step 3 Handling probabilistically the linear differential path in both branches

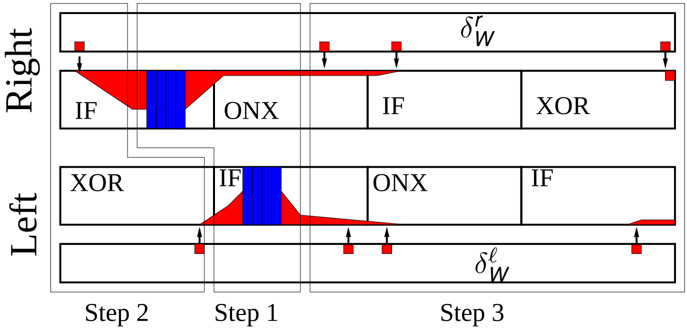
Finding a conforming pair



The collision search is composed of three subparts:

- step 1 Satisfying the Non Linear part of both branches
- step 2 Merging the two branches using some remaining free message words
- step 3 Handling probabilistically the linear differential path in both branches

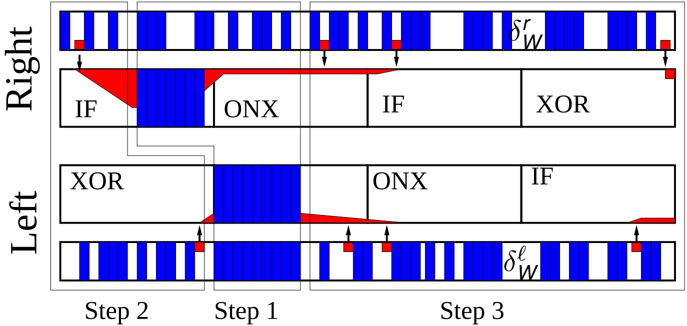
Finding a conforming pair



The collision search is composed of three subparts:

- step 1 Satisfying the Non Linear part of both branches
- step 2 Merging the two branches using some remaining free message words
- step 3 Handling probabilistically the linear differential path in both branches

Finding a conforming pair



The collision search is composed of three subparts:

- **step 1** Satisfying the Non Linear part of both branches
- **step 2** Merging the two branches using some remaining free message words
- **step 3** Handling probabilistically the linear differential path in both branches

Probability of the linear part

Fixed after the first step:

- The probability of the left branch is 2^{-15}.
- The probability of the right branch is $2^{-14.32}$.
- ...
- The overall probability for collision is $2^{-30.32}$.

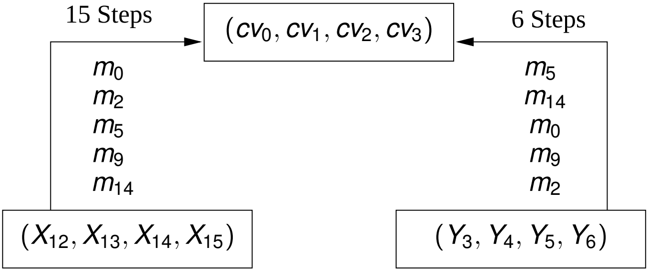
These theoretical probabilities had been verified experimentally.

To get a conforming cv and message pair,
we need to obtain $2^{30.32}$ solutions of the merging system.

Outline

Prepare the merging system

The system is very complex:



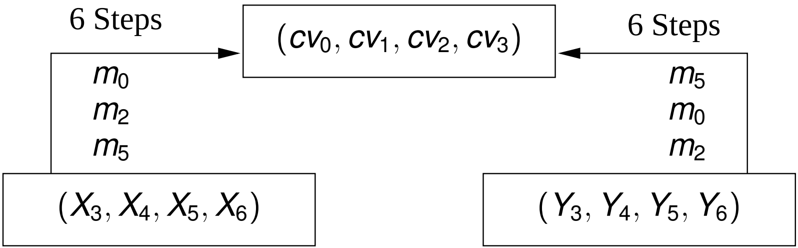
The probability that a random choice of $m_0, m_2, m_5, m_9, m_{14}$ gives a solution is

$$2^{-128}.$$

Reducing the merging system

We fix m_9 and m_{14}

- to get a system that represents less steps of the compression function.
- to get some conditions that help to solve



Impacts of the conditions

The following conditions gives us a simpler merging system.

- $X_5^{\ggg 5} \boxminus m_4 = \texttt{0xffffffff}$ (using m_9)
- $Y_3 = Y_4$ (using m_{11})

For example:

$$
\begin{aligned}
X_0 &= \mathit{Constant} \\
X_1 \oplus X_2 &= \mathit{Constant} \\
Y_1 &= \mathit{Constant} \\
\\
Y_2 &= \mathit{Constant} \boxminus m_2 \\
X_2 &= \mathit{Constant} \boxminus m_5
\end{aligned}
$$

Solving the merging system

To solve the merging system:

1. we find a value of m_2 that verifies $X_{-1} = Y_{-1}$,
2. then we directly deduce m_0 to fulfil $X_0 = Y_0$,
3. we obtain m_5 to satisfy a combination of $X_{-2} = Y_{-2}$ and $X_{-3} = Y_{-3}$
4. finally the 4^{th} equation is verified with probability 2^{-32}.

Complexity of the semi-free-start collision

- Solving the merging system costs 19 RIPEMD-128 step computations
(19/128 of the compression function cost).
- The probability of success of the merging is 2^{-34}.
- We need to find $2^{30.32}$ solutions of the merging system.

The complexity is

$$19/128 \times 2^{34} \times 2^{30.32} \simeq 2^{61.57}$$

calls to the compression function.

Conclusion

This work:

- a new cryptanalysis technique
- a collision attack on the full compression function of RIPEMD-128
- a distinguisher on the hash function of RIPEMD-128

Perspectives:

- improvement of this technique
- an example of collision
- apply to another 2-branches hash function

Thank you for your attention.

Cryptanalysis of Full RIPEMD-128

Franck Landelle and **Thomas Peyrin**

DGA MI - France NTU - Singapore

Eurocrypt 2013

Athens, Greece - May 28 , 2013

C. De Cannière and C. Rechberger.
Finding SHA-1 Characteristics: General Results and Applications.
In X. Lai and K. Chen, editors, *ASIACRYPT*, volume 4284 of *LNCS*, pages 1–20. Springer, 2006.

F. Mendel, T. Nad, and M. Schläffer.
Collision Attacks on the Reduced Dual-Stream Hash Function RIPEMD-128.
In A. Canteaut, editor, *FSE*, volume 7549 of *LNCS*, pages 226–243. Springer, 2012.

Y. Sasaki and L. Wang.
Distinguishers beyond Three Rounds of the RIPEMD-128/-160 Compression Functions.
In F. Bao, P. Samarati, and J. Zhou, editors, *ACNS*, volume 7341 of *LNCS*, pages 275–292. Springer, 2012.

X. Wang, X. Lai, D. Feng, H. Chen, and X. Yu.
Cryptanalysis of the Hash Functions MD4 and RIPEMD.
In *EUROCRYPT*, pages 1–18, 2005.