

# Cryptanalysis of the 10-Round Hash and Full Compression Function of SHAvite-3-512

Praveen Gauravaram<sup>1</sup>, Gaëten Leurent<sup>2</sup>, Florian Mendel<sup>3</sup>, María Naya-Plasencia<sup>4</sup>, Thomas Peyrin<sup>5</sup>, Christian Rechberger<sup>6</sup>, Martin Schläffer<sup>3</sup>,

<sup>1</sup>Department of Mathematics, DTU, Denmark,

<sup>2</sup>ENS, France,

<sup>3</sup>IAIK, TU Graz, Austria,

<sup>4</sup>FHNW Windisch, Switzerland,

<sup>5</sup>Ingenico, France,

<sup>6</sup>ESAT/COSIC, K.U.Leuven and IBBT, Belgium

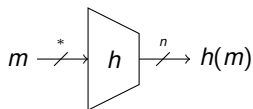
**Africacrypt 2010**

(initially discussed at ECRYPT2 Hash<sup>3</sup> workshop)

- 1 Motivation
- 2 SHAvite-3
- 3 Basic Attack Strategy
- 4 Attack on Compression Function
- 5 Attack on Hash Function
- 6 Conclusion

- 1 Motivation
- 2 SHAvite-3
- 3 Basic Attack Strategy
- 4 Attack on Compression Function
- 5 Attack on Hash Function
- 6 Conclusion

# Cryptographic Hash Function



Hash function  $h$  maps arbitrary length input  $m$  to  $n$ -bit output  $h(m)$

- Collision Resistance
  - find  $m, m'$  with  $m \neq m'$  and  $h(m) = h(m')$
  - birthday attack applies (freedom to choose  $h(m)$ )
  - generic complexity:  $2^{n/2}$
- Second-Preimage Resistance
  - given  $m, h(m)$  find  $m'$  with  $m \neq m'$  and  $h(m) = h(m')$
  - generic complexity:  $2^n$
- Preimage Resistance
  - given  $h(m)$  find  $m$
  - generic complexity:  $2^n$

# Hash Function Cryptanalysis

- Recent improvements in hash functions cryptanalysis
  - last decade: major weaknesses in many hash functions
  - especially in MD-family of hash functions
  - NIST standard SHA-1 broken
- NIST SHA-3 competition [Nat07] (2008-2012)
  - find a successor of SHA-1 and SHA-2
  - similar as AES competition (2000)

# SHA-3 Candidates

- 64 submissions to NIST call (October 2008)
- 51 round 1 candidates (December 2008)
  - many broken, too slow, not chosen, ...
- 14 round 2 candidates (August 2009)
  - chosen by NIST, tweaks allowed
- 5 finalists (fall 2010)
  - to focus analysis
- choose winner in 2011
  - standardize SHA-3 in 2012

# How to Compare Attacks on SHA-3 Candidates?

## Attacks on Building Blocks

- very different requirements for different designs
  - building blocks often not ideal
  - sponge: trivial “compression function” collisions/preimages
  - distinguishers on building blocks?
- when is an attack interesting?
  - NIST: not anticipated by the designers
  - if it extends to the hash function

# How to Compare Attacks on SHA-3 Candidates?

## Attacks on Building Blocks

- very different requirements for different designs
  - building blocks often not ideal
  - sponge: trivial “compression function” collisions/preimages
  - distinguishers on building blocks?
- when is an attack interesting?
  - NIST: not anticipated by the designers
  - if it extends to the hash function

## Attacks on Hash Function

- same requirements for all candidates
- a lot easier to compare
  - attacks on reduced hash function?
  - still hard to compare different security parameter(s)

# How to Compare Attacks on SHA-3 Candidates?

## Attacks on Building Blocks

- very different requirements for different designs
  - building blocks often not ideal
  - sponge: trivial “compression function” collisions/preimages
  - distinguishers on building blocks?
- when is an attack interesting?
  - NIST: not anticipated by the designers
  - if it extends to the hash function

## Attacks on Hash Function

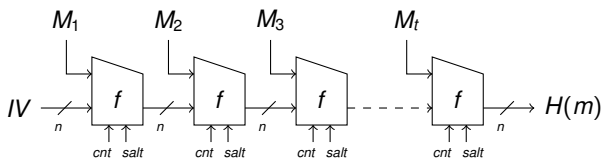
- same requirements for all candidates
- a lot easier to compare
  - attacks on reduced hash function?
  - still hard to compare different security parameter(s)

Collection of SHA-3 Attacks:

[http://ehash.iaik.tugraz.at/wiki/The\\_SHA-3\\_Zoo](http://ehash.iaik.tugraz.at/wiki/The_SHA-3_Zoo)

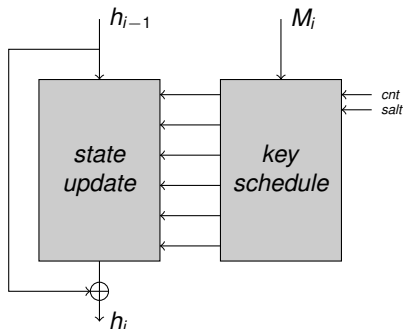
- 1 Motivation
- 2 SHAvite-3**
- 3 Basic Attack Strategy
- 4 Attack on Compression Function
- 5 Attack on Hash Function
- 6 Conclusion

# Description of SHAvite-3-512



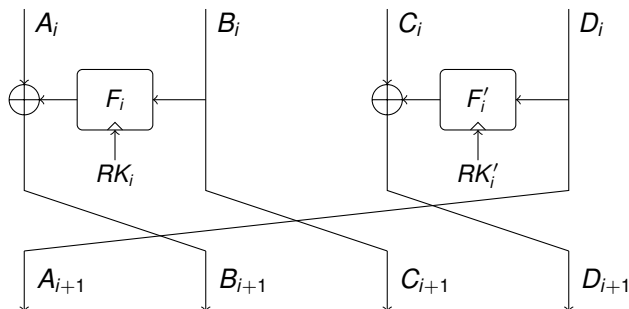
- Designed by Orr Dunkelman and Eli Biham [BD08]
  - Round 2 candidate
  - tweaked
- Iterated hash function
  - single-pipe construction
  - Haifa design principle

# SHAvite-3-512 Compression Function



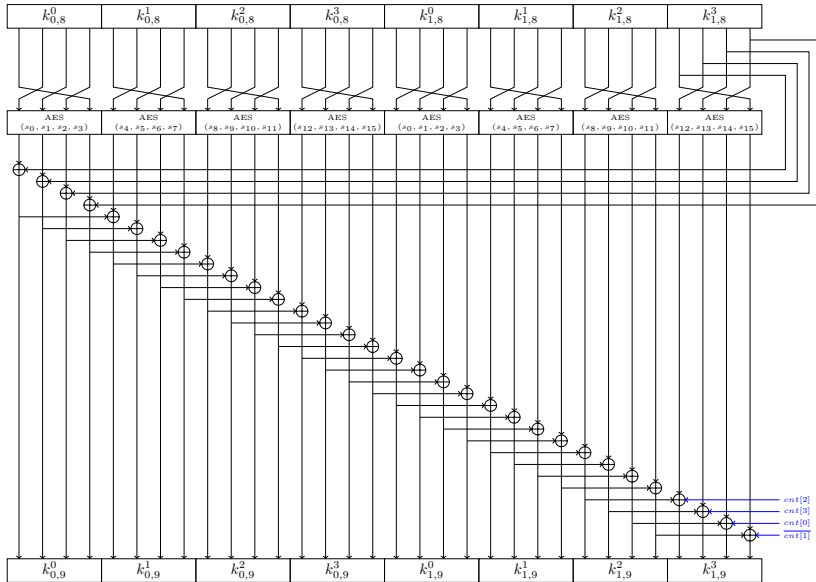
- block cipher in Davies-Meyer mode
- state update:
  - 14-round Feistel network (F-function: 4 AES rounds)
- key schedule:
  - parallel AES rounds with linear mixing layers

# State Update

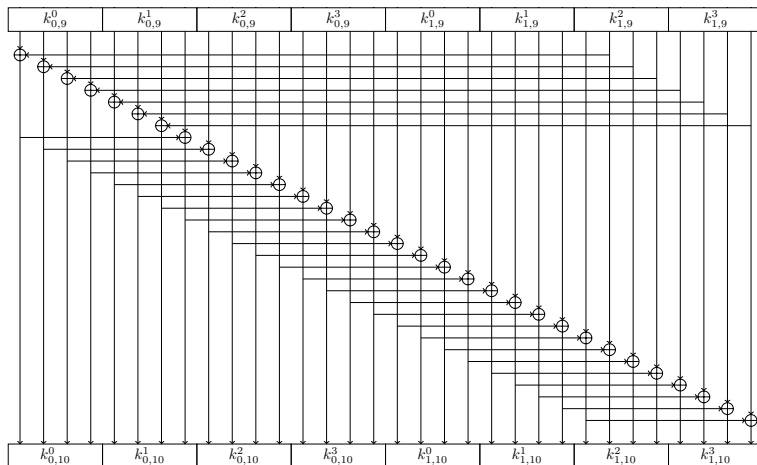


- $F_i(x) = AES(AES(AES(AES(x \oplus k_{0,i}^0) \oplus k_{0,i}^1) \oplus k_{0,i}^2) \oplus k_{0,i}^3))$
- $AES(x) = MixColumns(ShiftRows(SubBytes(x)))$
- $RK_i = (k_{0,i}^0, k_{0,i}^1, k_{0,i}^2, k_{0,i}^3)$

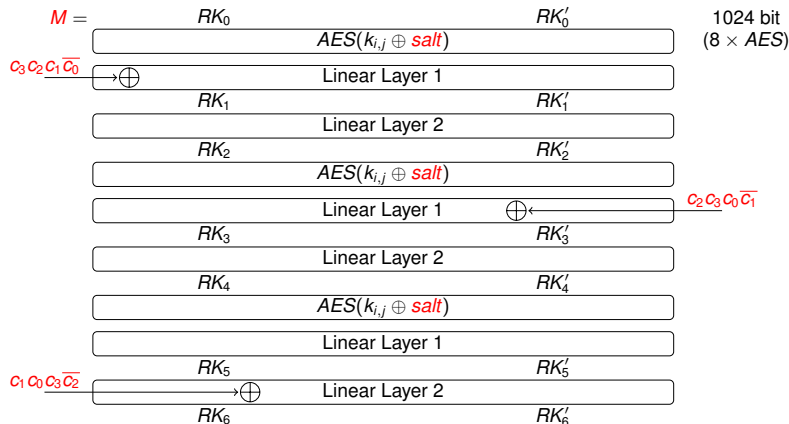
# Key Schedule



# Key Schedule



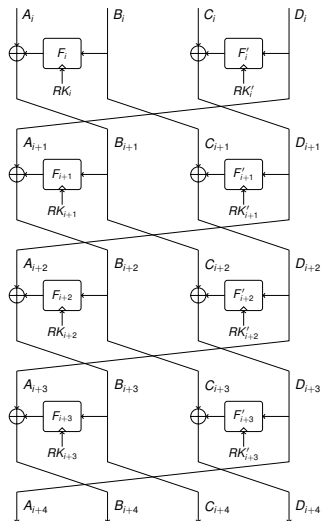
# Key Schedule (schematic)



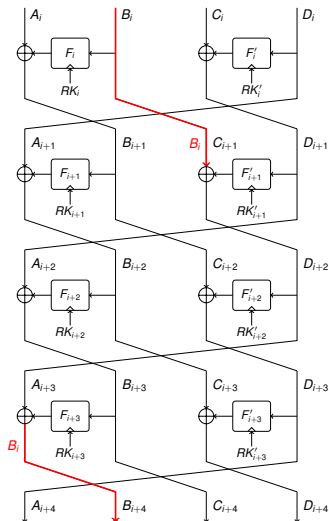
- Round 1: plain counter words added:  $cnt = c_0 c_1 c_2 c_3$
- Round 2: inverted and shuffled counter words added

- 1 Motivation
- 2 SHAvite-3
- 3 Basic Attack Strategy**
- 4 Attack on Compression Function
- 5 Attack on Hash Function
- 6 Conclusion

# Cancellation Property [BDLF09]



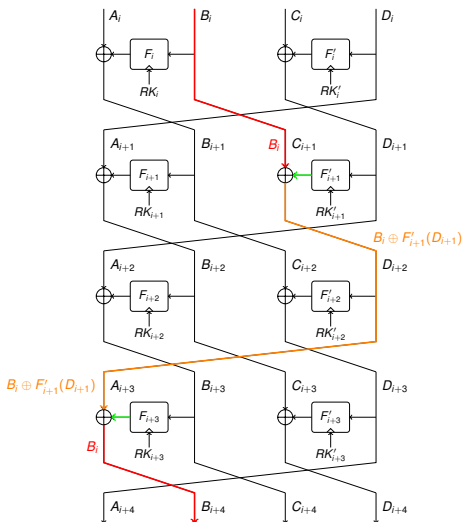
# Cancellation Property [BDLF09]



- idea: keep  $B_i$  unchanged

$$B_{i+4} = B_i$$

# Cancellation Property [BDLF09]



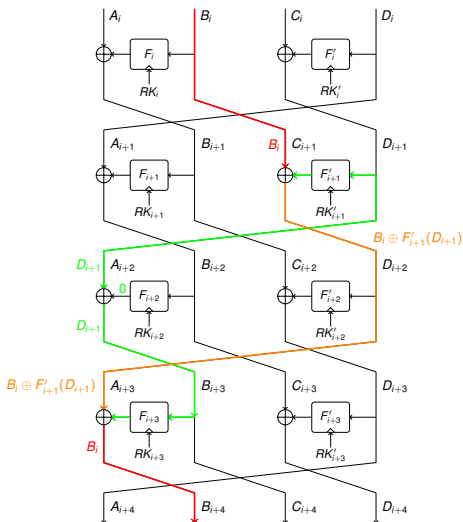
- idea: keep  $B_i$  unchanged

$$B_{i+4} = B_i$$

- when does this happen?

$$F_{i+3}(B_{i+3}) = F'_{i+1}(D_{i+1})$$

# Cancellation Property [BDLF09]



- idea: keep  $B_i$  unchanged

$$B_{i+4} = B_i$$

- when does this happen?

$$F_{i+3}(B_{i+3}) = F'_{i+1}(D_{i+1})$$

- or more specific:

$$F_{i+2}(B_{i+2}) = 0$$

$$RK_{i+3} = RK'_{i+1}$$

- second case:

- two 128-bit conditions
- but easier to fulfill
- conditions can be “interleaved”

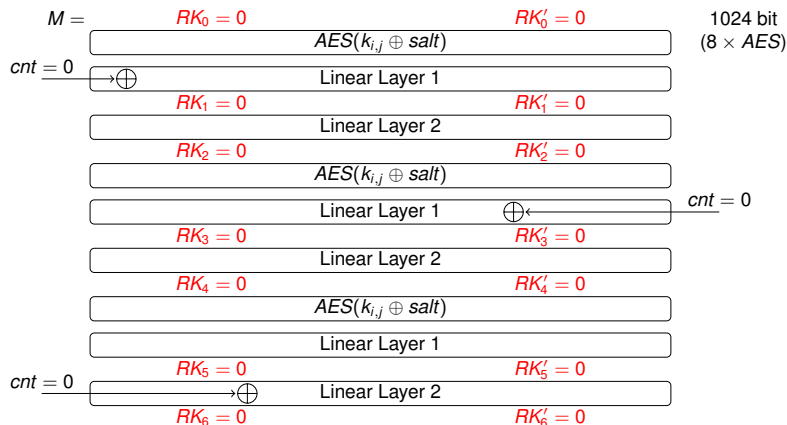
- interleave cancellation property with same value
  - $Z = B_i = B_{i+4}$
  - $Z = B_{i+2} = B_{i+4}$

# Interleaving

$i$	$A_i$	$B_i$	$C_i$	$D_i$	conditions
3	?	$Z$	?	?	
4	?	?	$Z$	$D_4$	
5	$D_4$	$Z$	?	$Z + F'_4(D_4)$	$F_5(Z) = 0$
6	$Z + F'_4(D_4)$	$D_4$	$Z$	$D_6$	$RK_6 = RK'_4$
7	$D_6$	$Z$	$D_4$	$Z + F'_6(D_6)$	$RK_7 = RK_5$
8	$Z + F'_6(D_6)$	$D_6$	$Z$	$D_8$	$RK_8 = RK'_6$
9	$D_8$	$Z$	$D_6$	$Z + F'_8(D_8)$	$RK_9 = RK_5$
10	$Z + F'_8(D_8)$	$D_8$	$Z$	$D_{10}$	$RK_{10} = RK'_8$
11	$D_{10}$	$Z$	$D_8$	$Z + F'_{10}(D_{10})$	$RK_{11} = RK_7$

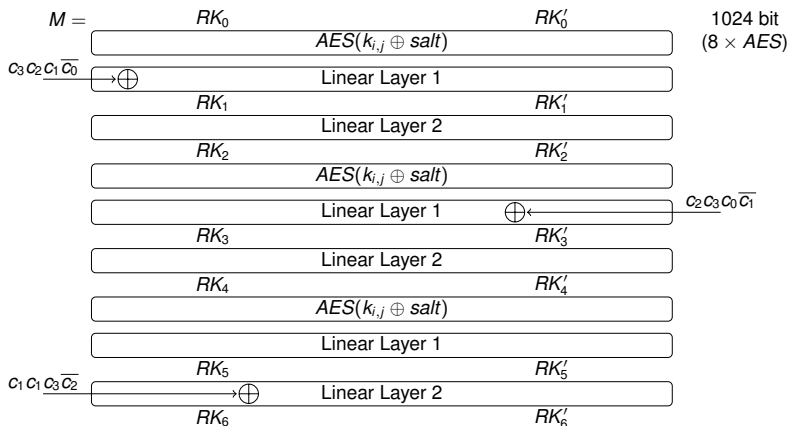
- interleave cancellation property with same value
    - $Z = B_i = B_{i+4}$
    - $Z = B_{i+2} = B_{i+4}$
  - conditions on state fulfill each other
    - we can choose  $Z = F_5^{-1}(0)$
- ⇒ we get conditions only on keys (message expansion)

# Weak Subkeys for SHAvite-3-512 (Round 1)



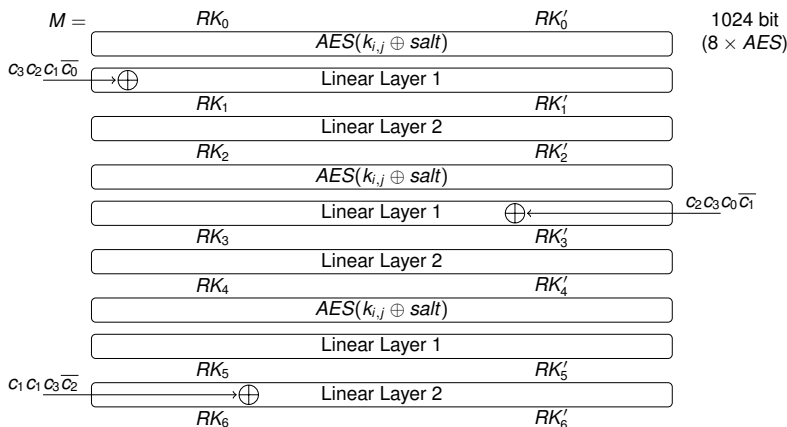
- construct all-zero subkeys [Pey09]
  - take the zero counter  $cnt = 0$
  - choose salt such that  $0 = AES(0 \oplus salt)$
  - $salt = 0x525252...52$

# Weak Subkeys for SHAvite-3-512 (Round 2)



- tweak for SHAvite-3-512 (Round 2):
  - some counter words are inverted
  - all-zero subkey not possible anymore

# Weak Subkeys for SHAvite-3-512 (Round 2)



- tweak for SHAvite-3-512 (Round 2):
  - some counter words are inverted
  - all-zero subkey not possible anymore
- choose  $c_2 c_3 c_0 \overline{c_1} = 0$  (valid counter!)
  - many round keys get zero

# Weak Subkeys for SHAvite-3-512 (Round 2)

$i$	$RK_i$				$RK'_i$				$r$
	$k_{0,i}^0$	$k_{0,i}^1$	$k_{0,i}^2$	$k_{0,i}^3$	$k_{1,i}^0$	$k_{1,i}^1$	$k_{1,i}^2$	$k_{1,i}^3$	
0	?	?	?	?	?	?	?	?	$M$
1	?*	?	?	?	?	?	?	0	1
2	0	?	?	?	?	0	0	0	2
3	0	?	?	?	0	0	0	0	3
4	0	?	0	0	0	0	0	0	4
5	0	0*	0	0	0	0	0	0	5
6	0	0	0	0	0	0	0	0	6
7	0	0	0	0	0	0	0	0	7
8	0	0	0	0	0	0	0	0	
9	0	0	0	0*	0	0	0	0	
10	0	0	0	0	0	0	0	0	
11	0	0	0	0	0	0	0	0	
12	0	0	0	0	0	0	0	0	
13	0	0	0	0	0	0	?*	?	

- key conditions are fulfilled for  $Z = B_3 = B_5 = \dots = B_{13}$

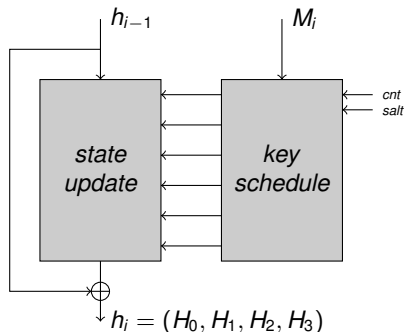
# Weak Subkeys for SHAvite-3-512 (Round 2)

$i$	$RK_i$				$RK'_i$				$r$
	$k_{0,i}^0$	$k_{0,i}^1$	$k_{0,i}^2$	$k_{0,i}^3$	$k_{1,i}^0$	$k_{1,i}^1$	$k_{1,i}^2$	$k_{1,i}^3$	
0	?	?	?	?	?	?	?	?	$M$
1	?*	?	?	?	?	?	?	0	1
2	0	?	?	?	?	0	0	0	2
3	0	?	?	?	0	0	0	0	3
4	0	?	0	0	0	0	0	0	4
5	0	0*	0	0	0	0	0	0	5
6	0	0	0	0	0	0	0	0	6
7	0	0	0	0	0	0	0	0	7
8	0	0	0	0	0	0	0	0	
9	0	0	0	0*	0	0	0	0	
10	0	0	0	0	0	0	0	0	
11	0	0	0	0	0	0	0	0	
12	0	0	0	0	0	0	0	0	
13	0	0	0	0	0	0	?*	?	

- key conditions are fulfilled for  $Z = B_3 = B_5 = \dots = B_{13}$
- in fact we can find  $2^{224}$  weak salts

- 1 Motivation
- 2 SHAvite-3
- 3 Basic Attack Strategy
- 4 Attack on Compression Function**
- 5 Attack on Hash Function
- 6 Conclusion

# Compression Function



- 1 partial preimage attack (of 128 bits):
  - given  $H_2$ , compute  $M_i, cnt, salt, h_i$
- 2 collision or preimage only on  $H_0, H_1, H_3$ 
  - complexity  $2^{192}$  and  $2^{384}$  ( $|h_i| = 512$ )

# The 14-Round Characteristic

$i$	$A_i$	$B_i$	$C_i$	$D_i$	conditions
0	?	?	?	?	
1	?	?	?	?	
2	?	$X$	?	?	
3	?	$Z$	$X$	?	
4	?	$Y$	$Z$	$D_4$	
5	$D_4$	$Z$	$Y$	$Z + F'_4(D_4)$	$F_5(Z) = 0$
6	$Z + F'_4(D_4)$	$D_4$	$Z$	$D_6$	$RK_6 = RK'_4$
7	$D_6$	$Z$	$D_4$	$Z + F'_6(D_6)$	$RK_7 = RK_5$
8	$Z + F'_6(D_6)$	$D_6$	$Z$	$D_8$	$RK_8 = RK'_6$
9	$D_8$	$Z$	$D_6$	$Z + F'_8(D_8)$	$RK_9 = RK_7$
10	$Z + F'_8(D_8)$	$D_8$	$Z$	$D_{10}$	$RK_{10} = RK'_8$
11	$D_{10}$	$Z$	$D_8$	$Z + F'_{10}(D_{10})$	$RK_{11} = RK_9$
12	$Z + F'_{10}(D_{10})$	$D_{10}$	$Z$	?	$RK_{12} = RK'_{10}$
13	?	$Z$	$D_{10}$	?	$RK_{13} = RK_{11}$
14	?	?	$Z$	?	

- choose  $(M, cnt, salt)$  according to key conditions
- compute  $Z$  in round 5
- we know that we get  $C_{14} = Z$
- missing: compute  $X, Y$  for given  $H_2 = C_0 \oplus C_{14}$

# Partial Preimage for 14-Rounds

$i$	$A_i$	$B_i$	$C_i$	$D_i$	<i>conditions</i>
0	?	?	?	?	
1	?	?	?	?	
2	?	$X$	?	?	
3	?	$Z$	$X$	?	
4	?	$Y$	$Z$	$D_4$	
5	$D_4$	$Z$	$Y$	$Z + F'_4(D_4)$	$F_5(Z) = 0$
6	$Z + F'_4(D_4)$	$D_4$	$Z$	$D_6$	$RK_6 = RK'_4$
7	$D_6$	$Z$	$D_4$	$Z + F'_6(D_6)$	$RK_7 = RK_5$
8	$Z + F'_6(D_6)$	$D_6$	$Z$	$D_8$	$RK_8 = RK'_6$
9	$D_8$	$Z$	$D_6$	$Z + F'_8(D_8)$	$RK_9 = RK_7$
10	$Z + F'_8(D_8)$	$D_8$	$Z$	$D_{10}$	$RK_{10} = RK'_8$
11	$D_{10}$	$Z$	$D_8$	$Z + F'_{10}(D_{10})$	$RK_{11} = RK_9$
12	$Z + F'_{10}(D_{10})$	$D_{10}$	$Z$	?	$RK_{12} = RK'_{10}$
13	?	$Z$	$D_{10}$	?	$RK_{13} = RK_{11}$
14	?	?	$Z$	?	

- write  $H_2 = C_0 \oplus C_{14}$  as a function of  $X, Y, Z$ :

$$H_2 = F_2(X) + F'_0(X + F_1(Z + F_4(Y) + F'_2(Y + F_3(Z))))$$

# Partial Preimage for 14-Rounds

$i$	$A_i$	$B_i$	$C_i$	$D_i$	conditions
0	?	?	?	?	
1	?	?	?	?	
2	?	$X$	?	?	
3	?	$Z$	$X$	?	
4	?	$Y$	$Z$	$D_4$	
5	$D_4$	$Z$	$Y$	$Z + F'_4(D_4)$	$F_5(Z) = 0$
6	$Z + F'_4(D_4)$	$D_4$	$Z$	$D_6$	$RK_6 = RK'_4$
7	$D_6$	$Z$	$D_4$	$Z + F'_6(D_6)$	$RK_7 = RK_5$
8	$Z + F'_6(D_6)$	$D_6$	$Z$	$D_8$	$RK_8 = RK'_6$
9	$D_8$	$Z$	$D_6$	$Z + F'_8(D_8)$	$RK_9 = RK_7$
10	$Z + F'_8(D_8)$	$D_8$	$Z$	$D_{10}$	$RK_{10} = RK'_8$
11	$D_{10}$	$Z$	$D_8$	$Z + F'_{10}(D_{10})$	$RK_{11} = RK_9$
12	$Z + F'_{10}(D_{10})$	$D_{10}$	$Z$	?	$RK_{12} = RK'_{10}$
13	?	$Z$	$D_{10}$	?	$RK_{13} = RK_{11}$
14	?	?	$Z$	?	

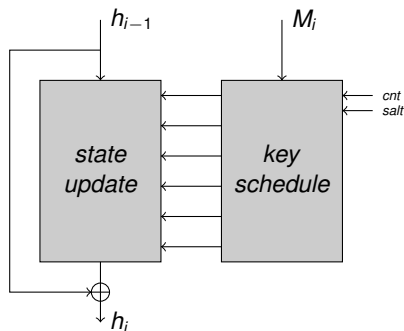
- write  $H_2 = C_0 \oplus C_{14}$  as a function of  $X, Y, Z$ :

$$H_2 = F_2(X) + F'_0(X + F_1(Z + F_4(Y) + F'_2(Y + F_3(Z))))$$

- solve for  $X, Y$  using birthday effect ( $2^{64}$ ):

$$F'_0{}^{-1}(H_2 + F_2(X)) + X = F_1(Z + F_4(Y) + F'_2(Y + F_3(Z)))$$

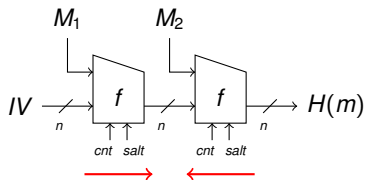
# Results for the Full Compression Function



- collision attack:
    - complexity  $2^{192}$  and  $2^{128}$  memory
  - preimage attacks:
    - complexity  $2^{384}$  and  $2^{128}$  memory
    - complexity  $2^{448}$  without memory
- with chosen salt and chosen counter

- 1 Motivation
- 2 SHAvite-3
- 3 Basic Attack Strategy
- 4 Attack on Compression Function
- 5 Attack on Hash Function**
- 6 Conclusion

# Hash Function



Can we extend the attack on the compression function to an attack on the hash function?

- in general: yes
  - if the design is single-pipe,
  - and we fix one output word:
  - do a meet-in-the-middle attack on 512 bit chaining value (two blocks needed)
- in this case: no
  - because salt is different for each 2nd block

⇒ extend the attack of [BDLF10] by one round

# Characteristic for 10 Rounds

$i$	$A_i$	$B_i$	$C_i$	$D_i$	<i>condition</i>
0	?	?	?	?	
1	?	?	?	?	
2	?	$X$	?	?	
3	?	$Z_7$	$X$	?	
4	?	$Y$	$Z_7$	$D_4$	
5	$D_4$	$Z_5$	$Y$	$Z_7 + F'_4(D_4)$	
6	$Z_7 + F'_4(D_4)$	$D_4 + F_5(Z_5)$	$Z_5$	$D_6$	$F_6(D_4 + F_5(Z_5)) = F'_4(D_4)$
7	$D_6$	$Z_7$	?	$Z_5 + F'_6(D_6)$	
8	$Z_5 + F'_6(D_6)$	$D_6 + F_7(Z_7)$	$Z_7$	?	$F_8(D_6 + F_7(Z_7)) = F'_6(D_6)$
9	?	$Z_5$	?	?	
10	?	?	$Z_5$	?	

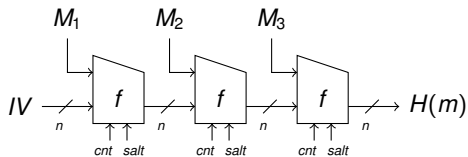
- fulfill conditions by carefully choosing subkey values [BDLF10]:

$$(k_{0,4}^1, k_{0,4}^2, k_{0,4}^3) = (k_{1,6}^1, k_{1,6}^2, k_{1,6}^3) \text{ and } k_{0,4}^0 + k_{1,6}^0 = F_5(Z_5)$$

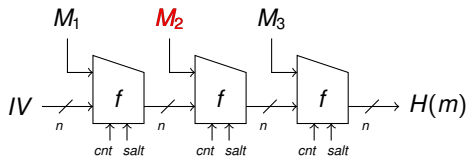
$$(k_{0,6}^1, k_{0,6}^2, k_{0,6}^3) = (k_{1,8}^1, k_{1,8}^2, k_{1,8}^3) \text{ and } k_{0,6}^0 + k_{1,8}^0 = F_7(Z_7)$$

- compute  $H_2 = C_0 \oplus C_{14}$  as a function of  $X, Y, Z_5, Z_7$ 
  - using birthday effect again ( $2^{64}$ )

# Hash Function Attack on 10 Rounds

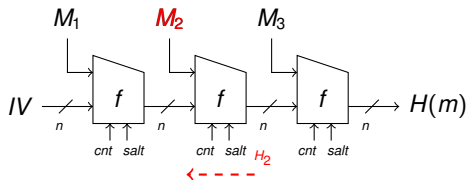


# Hash Function Attack on 10 Rounds



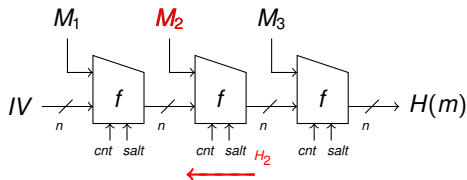
- 1 find a message ( $M_2$ ) according to key conditions ( $2^{224}$ )

# Hash Function Attack on 10 Rounds



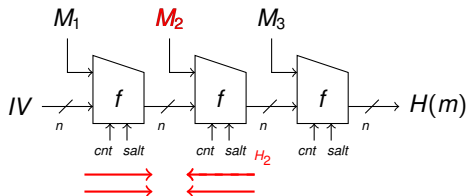
- 1 find a message ( $M_2$ ) according to key conditions ( $2^{224}$ )
- 2 find all  $2^{128}$  partial preimages (solutions for  $X, Y$ )
  - using cycle finding algorithm
  - total complexity:  $2^{128+64} = 2^{192}$

# Hash Function Attack on 10 Rounds



- 1 find a message ( $M_2$ ) according to key conditions ( $2^{224}$ )
- 2 find all  $2^{128}$  partial preimages (solutions for  $X, Y$ )
  - using cycle finding algorithm
  - total complexity:  $2^{128+64} = 2^{192}$
- 3 to find a preimage for the compression function
  - repeat previous steps  $2^{256}$  times
  - total complexity:  $2^{224+256} = 2^{480}$

# Hash Function Attack on 10 Rounds



- 1 find a message ( $M_2$ ) according to key conditions ( $2^{224}$ )
- 2 find all  $2^{128}$  partial preimages (solutions for  $X, Y$ )
  - using cycle finding algorithm
  - total complexity:  $2^{128+64} = 2^{192}$
- 3 to find a preimage for the compression function
  - repeat previous steps  $2^{256}$  times
  - total complexity:  $2^{224+256} = 2^{480}$
- 4 construct a second-preimage for the hash function
  - using unbalanced meet-in-the-middle attack
  - complexity:  $2^{497}$  and  $2^{16}$  memory

- 1 Motivation
- 2 SHAvite-3
- 3 Basic Attack Strategy
- 4 Attack on Compression Function
- 5 Attack on Hash Function
- 6 Conclusion**

- Attacks on SHAvite-3-512:
  - Full Compression Function
  - 10/14 Rounds for the Hash Function
- Why does it work?
  - *salt*, *cnt* inputs: weaker compression functions (harder to extend attacks to hash function)
  - regular key schedule
  - Feistel: we can keep properties for many rounds
  - single-pipe design
- Security margin already rather small
- Attack did not use properties of AES yet  
(even works for ideal permutation instead of AES rounds)

# References I



Eli Biham and Orr Dunkelman.

The SHAvite-3 Hash Function.

Submission to NIST, 2008.

Available online at <http://ehash.iaik.tugraz.at/uploads/f/f5/Shavite.pdf>.



Charles Boullaguet, Orr Dunkelman, Gaëtan Leurent, and Pierre-Alain Fouque.

Attacks on Hash Functions based on Generalized Feistel - Application to Reduced-Round Lesamnta and SHAvite-3<sub>512</sub>.

Cryptology ePrint Archive, Report 2009/634, 2009.

<http://eprint.iacr.org/>.



Charles Boullaguet, Orr Dunkelman, Gaëtan Leurent, and Pierre-Alain Fouque.

Another Look at Complementation Properties.

In Seokhie Hong and Tetsu Iwata, editors, *FSE*, LNCS. Springer, 2010.

to appear.



National Institute of Standards and Technology.

Announcing Request for Candidate Algorithm Nominations for a New Cryptographic Hash Algorithm (SHA-3) Family.

*Federal Register*, 27(212):62212–62220, November 2007.

Available: [http://csrc.nist.gov/groups/ST/hash/documents/FR\\_Notice\\_Nov07.pdf](http://csrc.nist.gov/groups/ST/hash/documents/FR_Notice_Nov07.pdf).



Thomas Peyrin.

Chosen-salt, chosen-counter, pseudo-collision on SHAvite-3 compression function, 2009.

Available online: <http://ehash.iaik.tugraz.at/uploads/e/ea/Peyrin-SHAvite-3.txt>.