

# *A Look at the SHA-3 Competition: Design and Analysis of Hash Functions*

Gaëtan Leurent

École Normale Supérieure  
Paris, France

University of Luxembourg  
January 19, 2010

# Outline

## *Introduction*

- Hash functions
- The MD4 family

## *The SHA-3 competition*

- New designs
- SIMD

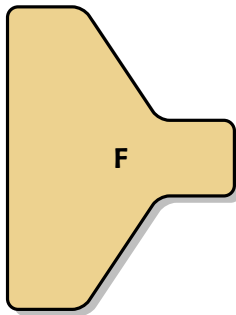
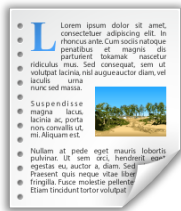
## *New attacks on SHA-3 candidates*

- Self-similarity attacks
- Cancellation cryptanalysis on generalized Feistels



## What is a hash function?

- ▶ A **public** function with **no structural properties**.
  - ▶ Cryptographic strength without keys!
- ▶  $F: \{0, 1\}^* \rightarrow \{0, 1\}^n$



0x1d66ca77ab361c6f

## Security goals

### Preimage attack

Given  $F$  and  $\bar{H}$ , find  $M$  s.t.  $F(M) = \bar{H}$ .

Ideal security:  $2^n$ .

### Second-preimage attack

Given  $F$  and  $M_1$ , find  $M_2 \neq M_1$  s.t.  $F(M_1) = F(M_2)$ .

Ideal security:  $2^n$ .

### Collision attack

Given  $F$ , find  $M_1 \neq M_2$  s.t.  $F(M_1) = F(M_2)$ .

Ideal security:  $2^{n/2}$ .

- ▶ Ideal behaviour: random oracle.

## Security goals

### Preimage attack

Given  $F$  and  $\bar{H}$ , find  $M$  s.t.  $F(M) = \bar{H}$ .

Ideal security:  $2^n$ .

### Second-preimage attack

Given  $F$  and  $M_1$ , find  $M_2 \neq M_1$  s.t.  $F(M_1) = F(M_2)$ .

Ideal security:  $2^n$ .

### Collision attack

Given  $F$ , find  $M_1 \neq M_2$  s.t.  $F(M_1) = F(M_2)$ .

Ideal security:  $2^{n/2}$ .

- ▶ Ideal behaviour: random oracle.

## Security goals

### Preimage attack

Given  $F$  and  $\bar{H}$ , find  $M$  s.t.  $F(M) = \bar{H}$ .

Ideal security:  $2^n$ .

### Second-preimage attack

Given  $F$  and  $M_1$ , find  $M_2 \neq M_1$  s.t.  $F(M_1) = F(M_2)$ .

Ideal security:  $2^n$ .

### Collision attack

Given  $F$ , find  $M_1 \neq M_2$  s.t.  $F(M_1) = F(M_2)$ .

Ideal security:  $2^{n/2}$ .

- ▶ Ideal behaviour: random oracle.

## *Security definitions: difficulties*

- ▶ A single function can not be collision resistant.
  - ▶ Precomputation is allowed in standard security definition
  - ▶ Define a family of function
- ▶ Obvious relations between the security definitions do not hold.
  - ▶ Even more mess with families of functions!

## Use as a one-way function

- ▶ Unix password file
  - ▶ Store  $H(pw)$
  - ▶ Allow verification of the password without storing the password
- ▶ One-time password
  - ▶ User picks  $x$  and server stores  $y = H(H(H(x)))$
  - ▶ To authenticate, user sends a preimage of  $y$
  - ▶ First authentication with  $H(H(x))$ , server now stores  $H(H(x))$
  - ▶ Second authentication with  $H(x)$
  - ▶ ...

## Use as unique identifiers

- ▶ Hash-and-sign
  - ▶ Signature algorithm are costly
  - ▶ Sign  $H(m)$  instead of  $m$
- ▶ Commitment
  - ▶ Alice commits to  $H(m)$  without revealing  $m$ .
  - ▶ Later, she reveals  $m$ .
- ▶ Time-stamping
  - ▶ Authority certifies that  $H(m)$  was known at time  $t_1$
  - ▶  $m$  is revealed at time  $t_2$
  - ▶ Need a stronger notion that second-preimage resistance: herding attack

## Breaking the structure of the input

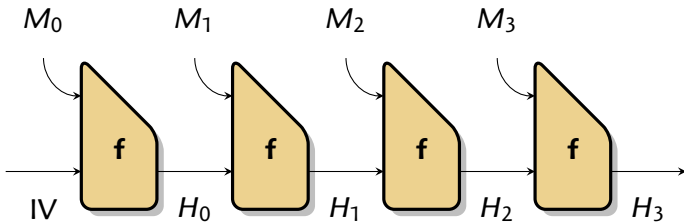
- ▶ Key derivation
- ▶ Full Domain Hash
  - ▶ Avoid the structural properties of RSA
  - ▶ For a RSA key  $(N, e, d)$
  - ▶  $H$  a hash function to  $\mathbb{Z}_N$
  - ▶ Signature:  $s = H(m)^d$
  - ▶ Verification:  $s^e \stackrel{?}{=} H(m)$
- ▶ Rabin signatures
  - ▶ Compute a square root of  $H(m)$  modulo an RSA number
  - ▶ Broken if one can find  $H(m') = -H(m)$

## Use as a MAC

- ▶ Message Authentication Code
  - ▶ Symmetric signature
  
- ▶ Secret-prefix MAC
  - ▶  $MAC_k(m) = H(k\|m)$
  
- ▶ HMAC
  - ▶  $HMAC_k(m) = H(k \oplus \text{opad} \| H(k \oplus \text{ipad} \| m))$
  
- ▶ Challenge-response authentication
  - ▶ Alice sends a random challenge  $r$
  - ▶ Bob replies with  $MAC_k(r)$

## Hash function design

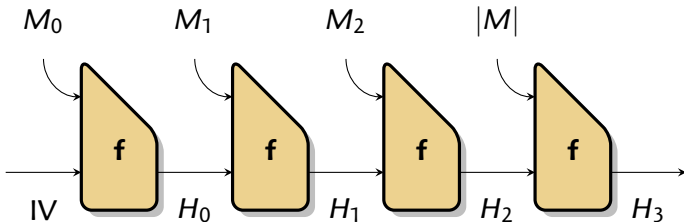
- ▶ Build a smaller compression function, and iterate.
  - ▶ Cut the message in chunks  $M_0, \dots, M_k$
  - ▶  $H_i = f(M_i, H_{i-1})$
  - ▶  $F(M) = H_k$



## Security proof (Merkle, Damgård)

### Theorem

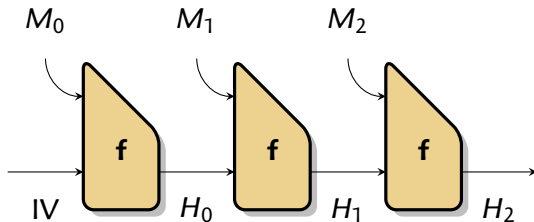
If one finds a collision in the hash function, then one has a collision in the compression function.



- ▶ If  $|M| \neq |M'|$ , collision in last block.
- ▶ Else, look for last block with  $H_i = H'_i$ .

## Length extension attack

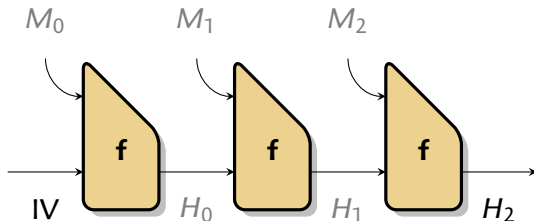
- ▶ Given the hash of an unknown message we can compute the hash of some related messages.



- ▶  $H(M||M') = H_3$  can be computed from  $H(M) = H_2$  and  $M'$ .
  - ▶ Breaks secret-prefix MAC.
- ▶ Solution: use a **finalisation function**.

## Length extension attack

- ▶ Given the hash of an unknown message we can compute the hash of some related messages.

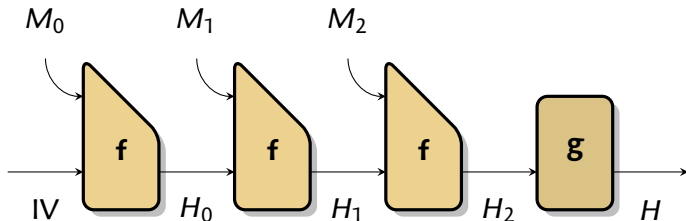


- ▶  $H(M||M') = H_3$  can be computed from  $H(M) = H_2$  and  $M'$ .
  - ▶ Breaks secret-prefix MAC.
- ▶ Solution: use a **finalisation function**.



## Length extension attack

- ▶ Given the hash of an unknown message we can compute the hash of some related messages.



- ▶  $H(M||M') = H_3$  can be computed from  $H(M) = H_2$  and  $M'$ .
  - ▶ Breaks secret-prefix MAC.
- ▶ Solution: use a **finalisation function**.

## Other attacks against Merkle-Damgård

- ▶ Long message second-preimage attack.
  - ▶ Given a message of length  $2^k$ , a preimage costs  $2^{n-k}$ .
- ▶ Multi-collision attack.
  - ▶ Build a set of  $2^k$  colliding messages with time  $k \times 2^n$ .
- ▶ Herding attack.
  - ▶ Commit to a value, and choose the message later.
  - ▶ Cost about  $2^{2n/3}$ .
- ▶ Solution: use a bigger state.

## Other attacks against Merkle-Damgård

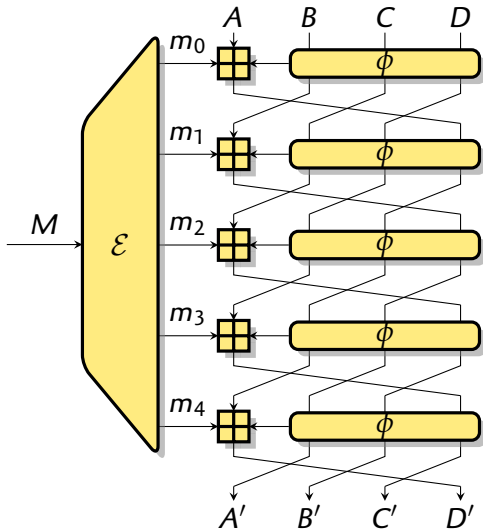
- ▶ Long message second-preimage attack.
  - ▶ Given a message of length  $2^k$ , a preimage costs  $2^{n-k}$ .
- ▶ Multi-collision attack.
  - ▶ Build a set of  $2^k$  colliding messages with time  $k \times 2^n$ .
- ▶ Herding attack.
  - ▶ Commit to a value, and choose the message later.
  - ▶ Cost about  $2^{2n/3}$ .
- ▶ Solution: use a **bigger state**.

## MD family design

- ▶ MD4 designed by Rivest in 1990
- ▶ MD5 designed by Rivest in 1991
- ▶ One of the first dedicated hash function
- ▶ Based on a dedicated block-cipher in Davies-Meyer mode:  

$$H_i = CF(H_{i-1}, M) = E_M(H_{i-1}) \oplus H_{i-1}$$

## MD family design



► Input:

$M \leftarrow$  **Message**

$(A, B, C, D) \leftarrow$  **Chaining value**

► Output:

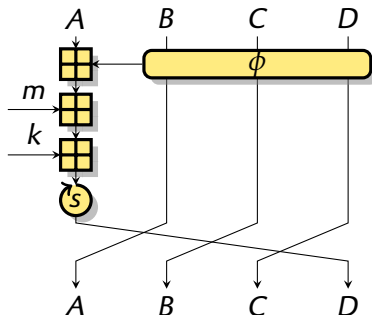
$(A + A', B + B', C + C', D + D')$

► 32-bit registers

► Simple operations

► Message expansion:  
permutation based

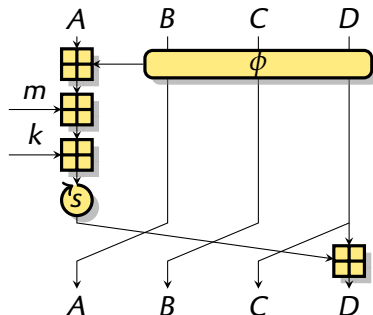
## MD4 design



$$Q_i = (Q_{i-4} \boxplus m_i \boxplus k_i \boxplus \Phi_i(Q_{i-1}, Q_{i-2}, Q_{i-3})) \lll S_i$$

- ▶ 48 steps (16 message words)
- ▶ Boolean functions: IF, MAJ, XOR

## MD5 design



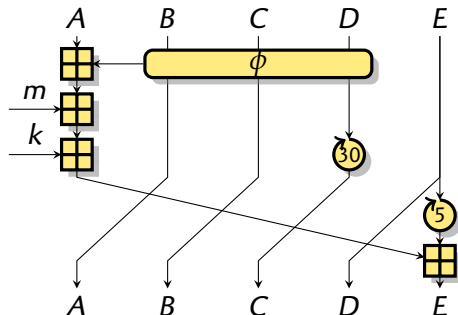
$$Q_i = (Q_{i-4} \boxplus m_i \boxplus k_i \boxplus \Phi_i(Q_{i-1}, Q_{i-2}, Q_{i-3})) \lll s_i \boxplus Q_{i-1}$$

- ▶ 64 steps (16 message words)
- ▶ Boolean functions: IF, MAJ, XOR, ONX

# SHA-1 design

- ▶ Successor to MD4/MD5
- ▶ Designed by NIST in 1993
- ▶ Bigger hash output / bigger state
- ▶ Stronger message expansion
  - ▶ Linear code
  - ▶  $m_i = (m_{i-3} \oplus m_{i-8} \oplus m_{i-14} \oplus m_{i-16}) \lll 1$

# SHA-1 design



$$Q_i = Q_{i-5}^{\lll 30} \boxplus m_i \boxplus k_i \boxplus \Phi_i(Q_{i-2}, Q_{i-3}^{\lll 30}, Q_{i-4}^{\lll 30}) \boxplus Q_{i-1}^{\lll 5}$$

- ▶ 80 steps (16 message words)
- ▶ Boolean functions: IF, MAJ, XOR

## Wang et. al's attacks

- ▶ In 2004, new attacks against MD4, MD5, SHA-1, RIPEMD-0
- ▶ Based on a differential attack:
  - ▶ Consider a pair of message with a small difference
  - ▶ Try to control the propagation of the differences
- ▶ New ideas:
  - ▶ Use a signed difference
  - ▶ Use a set of necessary conditions
  - ▶ Some conditions are easy to satisfy:  
message modification
- ▶ A lot of work by hand to find differential characteristic.

## Main mistakes

*MD4* Not enough rounds

*MD5* A difference in the MSB can stay in the MSB

(Den Boer and Bosselaers, 1993)

$$Q'_i = Q_i \oplus 2^{31}$$

$$Q_i = (Q_{i-4} \boxplus m_i \boxplus k_i \boxplus \Phi_i(Q_{i-1}, Q_{i-2}, Q_{i-3})) \lll_{s_i} \boxplus Q_{i-1}$$

*SHA-1* Message expansion is a cyclic code

It is possible to shift a difference pattern

Used to build local collisions

# Outline

## *Introduction*

- Hash functions
- The MD4 family

## *The SHA-3 competition*

- New designs
- SIMD

## *New attacks on SHA-3 candidates*

- Self-similarity attacks
- Cancellation cryptanalysis on generalized Feistels

## The SHA-3 competition

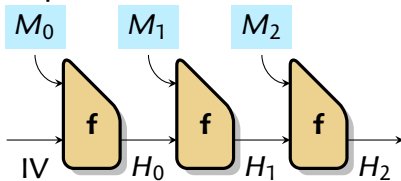
- ▶ Similar to the AES competition
- ▶ Organized by NIST
  
- ▶ Submission dead-line was October 2008: 64 candidates
- ▶ 51 valid submissions
  
- ▶ 14 in the second round (July 2009)
- ▶ 5 finalists in September 2010?
- ▶ Winner in 2012?

## *New designs*

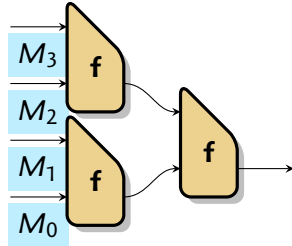
- ▶ Take into consideration recent advances in cryptanalysis
- ▶ Somewhat higher expectation than SHA-2
- ▶ Second round candidates seem quite solid...
- ▶ Wide diversity of designs

# Mode of operation

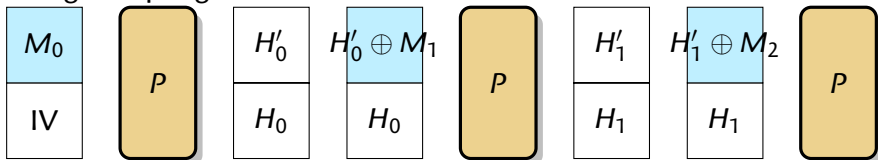
► Sequential



► Tree-based



► Using the sponge construction

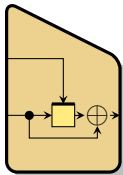


## *Construction of the compression function*

- ▶ From a (supposedly) perfect primitive
  - ▶ Most block cipher based designs, Keccak
  - ▶ Security proofs
    - ▶ By reduction
    - ▶ Indifferentiability proof
- ▶ From a weak primitive with a large state and a small message block
  - ▶ CubeHash, RadioGatún, Grindhal
  - ▶ Security proof only rules out generic attack
- ▶ By reduction to a class of hard problem
  - ▶ Usually slow
  - ▶ Security proof will be asymptotic

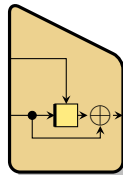
## Construction of the compression function

- ▶ From a block cipher



$$H_i = E_M(H_{i-1}) \oplus H_{i-1}$$

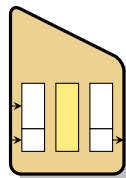
Davies-Meyer



$$H_i = E_{H_{i-1}}(M) \oplus M$$

Matyas-Meyer-Oseas

- ▶ From a permutation



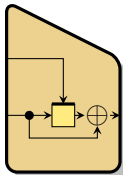
$$H_i = Tr(P(H_{i-1} || M))$$

- ▶ Something else...
  - ▶ Shabal, Grøstl, Luffa, ...

- ▶ Something broken...

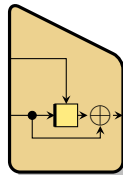
## Construction of the compression function

- ▶ From a block cipher



$$H_i = E_M(H_{i-1}) \oplus H_{i-1}$$

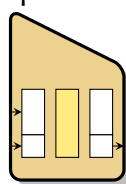
Davies-Meyer



$$H_i = E_{H_{i-1}}(M) \oplus M$$

Matyas-Meyer-Oseas

- ▶ From a permutation



$$H_i = \text{Tr}(P(H_{i-1} || M))$$

- ▶ Something else...
  - ▶ Shabal, Grøstl, Luffa, ...
- ▶ Something broken...

## *Inside the compression function*

- ▶ Feistel or SPN
- ▶ ARX
  - ▶ Additions, Rotation, XOR
  - ▶ Sometimes Shifts, Boolean function
- ▶ AES-based or AES-inspired
  - ▶ Can take advantage of Intel AES instructions
- ▶ Bitsliced

# The design of SIMD

- ▶ SHA-3 candidate selected in the second round
- ▶ Built on the MD/SHA legacy
- ▶ Secure against differential attacks



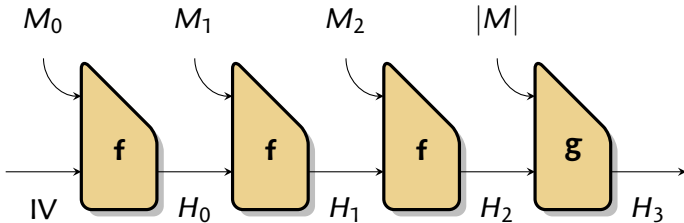
Gaëtan Leurent, Pierre-Alain Fouque, Charles Bouillaguet  
SIMD Is a Message Digest  
Submission to the NIST SHA-3 competition

## Main Features of SIMD

- ▶ Security
  - ▶ Strong message expansion
  - ▶ **Proof of security** against differential cryptanalysis
  
- ▶ Parallelism
  - ▶ Small scale parallelism (inside the compression function):  
good for hardware / software with SIMD instructions
  - ▶ Can use two cores: message expansion / compression
  
- ▶ Performance
  - ▶ Very good on high-end desktops: **11 cycles/byte** on Core2
  - ▶ Good if SIMD instructions are available:  
*SSE* on x86, *Altivec* on PowerPC, *lwMMXt* on ARM, *VIS* on SPARC...
  - ▶ Drawback: no portable efficient implementation.

## What mode of operation?

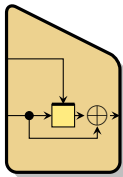
- ▶ Iterate a compression function
  - ▶ Easier to analyse
- ▶ Double the size of the state
  - ▶ Avoid generic attacks
- ▶ Finalisation function takes the message size as input





## How to build the compression function?

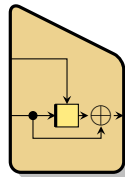
### ▶ Davies-Meyer:



$$H_i = E_M(H_{i-1}) \oplus H_{i-1}$$

- ▶ differential attack on  $C$   
 $\rightsquigarrow$  related key attack on  $E$

### ▶ Matyas-Meyer-Oseas



$$H_i = E_{H_{i-1}}(M) \oplus M$$

- ▶ differential attack on  $C$   
 $\rightsquigarrow$  differential attacks  $E$

▶ Two inputs:  $H_{i-1}$  hard to control /  $M$  easy to control.

▶ With DM, **message expansion** can reduce control over  $M$

## The Message Expansion

	Message block	Expanded message	Minimal distance
SIMD-256	512 bits	4096 bits	520 bits
SIMD-512	1024 bits	8192 bits	1032 bits

- ▶ Provides resistance to differential attack
- ▶ Based on (error correcting) codes with a good minimal distance
- ▶ Concatenated code:
  - ▶ outer code gives a high word distance
  - ▶ inner code gives a high bit distance

# Outer Code

## Reed-Solomon code

- ▶ Interpret the input ( $k$  words) as a polynomial of degree  $k - 1$  over some finite field
- ▶ Evaluate on  $n$  points ( $n > k$ )
- ▶ **MDS code**: minimal distance  $n - k + 1$

	$k$	$n$	$d$
SIMD-256	64	128	65
SIMD-512	128	256	129

- ▶ Efficiency:
  - ▶ Compute with an FFT algorithm
  - ▶ Use the field  $\mathbb{F}_{257}$
- ▶ Add a constant part: affine code

## Inner code

We encode the output words of the FFT twice,  
through two different inner codes.

Very efficient codes, with a single 16-bit multiplication.

$$I_{185} : \mathbb{F}_{257} \mapsto \mathbb{Z}_{2^{16}}$$

$$x \rightarrow 185 \boxtimes \tilde{x} \quad \text{where } -128 \leq \tilde{x} \leq 128 \text{ and } \tilde{x} = x \pmod{257}$$

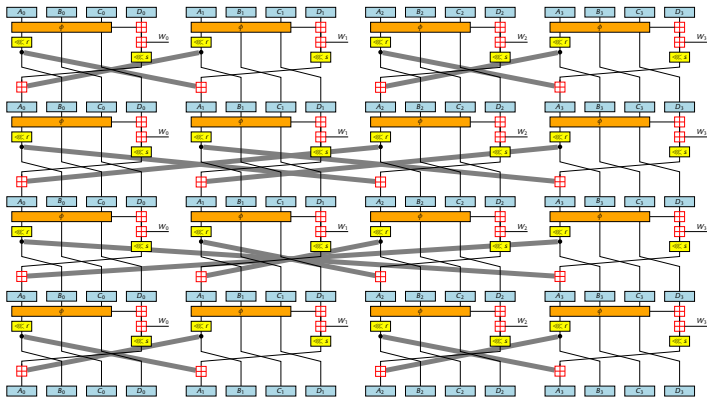
$$I_{233} : \mathbb{F}_{257} \mapsto \mathbb{Z}_{2^{16}}$$

$$x \rightarrow 233 \boxtimes \tilde{x} \quad \text{where } -128 \leq \tilde{x} \leq 128 \text{ and } \tilde{x} = x \pmod{257}$$

The magic constants 185 and 233 give a **minimal distance of 4 bits**.  
(also for signed difference)

## How to build the compressing part?

- ▶ Unbalanced Feistels with simple bit-wise functions
  - ▶ Follow the MD/SHA family
- ▶ Use parallel Feistel to allow a bigger state





# Outline

## *Introduction*

- Hash functions
- The MD4 family

## *The SHA-3 competition*

- New designs
- SIMD

## *New attacks on SHA-3 candidates*

- Self-similarity attacks
- Cancellation cryptanalysis on generalized Feistels

## Self-similarity attacks

- ▶ Generalization of the complementation property of DES
- ▶ Applied to SHA-3 candidate *Lesamnta*

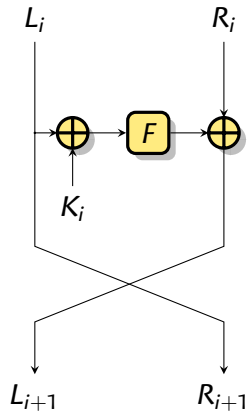


Charles Bouillaguet, Orr Dunkelman, Gaëtan Leurent, and Pierre-Alain Fouque

Another Look at Complementation Properties

## DES's Complementation Property

- ▶ If the key is bitwise complemented, so are all the subkeys.
- ▶ If the input to the round function is also bitwise complemented, the complementation is canceled.
- ▶ In other words, the input to the S-boxes is the same.
- ▶ **DES's complementation property:**



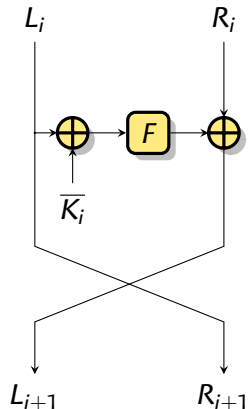
## DES's Complementation Property

- ▶ If the key is bitwise complemented, so are all the subkeys.

$$K \rightarrow K_1, K_2, \dots, K_{16} \text{ and}$$

$$\overline{K} \rightarrow \overline{K_1}, \overline{K_2}, \dots, \overline{K_{16}}$$

- ▶ If the input to the round function is also bitwise complemented, the complementation is canceled.
- ▶ In other words, the input to the S-boxes is the same.
- ▶ **DES's complementation property:**





## DES's Complementation Property

- ▶ If the key is bitwise complemented, so are all the subkeys.

$K \rightarrow \overline{K_1}, \overline{K_2}, \dots, \overline{K_{16}}$  and

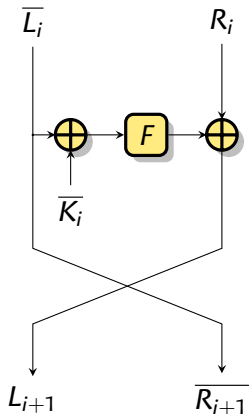
$\overline{K} \rightarrow K_1, K_2, \dots, K_{16}$

- ▶ If the input to the round function is also bitwise complemented, the complementation is canceled.

- ▶ In other words, the input to the S-boxes is the same.

And the output of the S-boxes.

- ▶ DES's complementation property:



## DES's Complementation Property

- ▶ If the key is bitwise complemented, so are all the subkeys.

$$K \rightarrow K_1, K_2, \dots, K_{16} \text{ and}$$

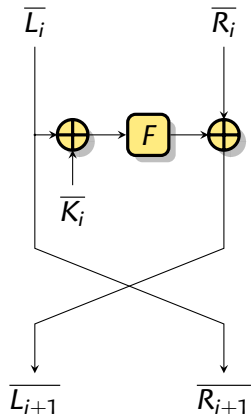
$$\overline{K} \rightarrow \overline{K_1}, \overline{K_2}, \dots, \overline{K_{16}}$$

- ▶ If the input to the round function is also bitwise complemented, the complementation is canceled.
- ▶ In other words, the input to the S-boxes is the same.

And the output of the S-boxes.

- ▶ **DES's complementation property:**

$$DES_K(P) = \overline{DES_{\overline{K}}(\overline{P})}$$



## Examples in hash functions

- ▶ In CHI:
 
$$CF(H, M) = CF(\overline{H}, \overline{M})$$
  - ▶ This property is a collision in the compression function.
  
- ▶ In MD5:
 
$$CF(H, M) = CF(H \oplus 2^{32}, M \oplus 2^{32})$$
 with probability  $2^{-48}$ 
  - ▶ Basic property used in many attacks
  
- ▶ Can we find more?
  - ▶ Look for simple transformations  $\phi$ ,  $\psi$  and  $\theta$  such that:
 
$$\theta(CF(X, M)) = CF(\phi(X), \psi(M))$$

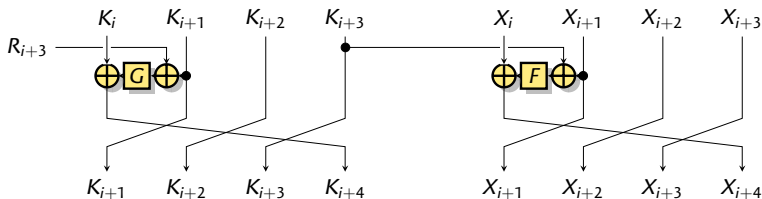
# Lesamnta

- ▶ Davies-Meyer with an MMO compression function
- ▶ Generalized Feistel
- ▶ Round function is AES-based



Shoichi Hirose, Hidenori Kuwakado, Hirotaka Yoshida  
SHA-3 Proposal: Lesamnta  
Submission to the NIST SHA-3 competition

## Lesamnta (cont.)

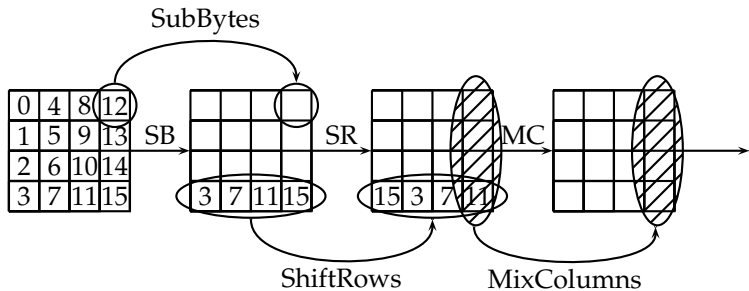


$$X_{i+4} = X_i \oplus F(X_{i+1} \oplus K_{i+3})$$

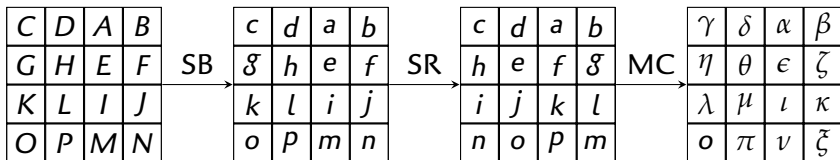
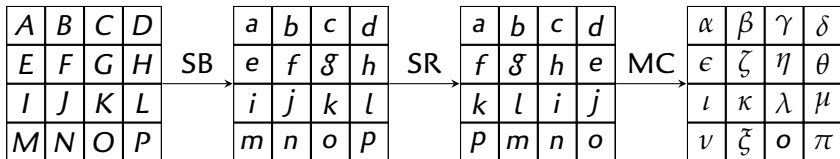
$$K_{i+4} = K_i \oplus G(K_{i+1} \oplus R_{i+3}).$$

- ▶ Message loaded to  $K_{-3}, K_{-2}, K_{-1}, K_0$
- ▶ Chaining value loaded to  $X_{-3}, X_{-2}, X_{-1}, X_0$
- ▶  $F$  and  $G$  AES-based

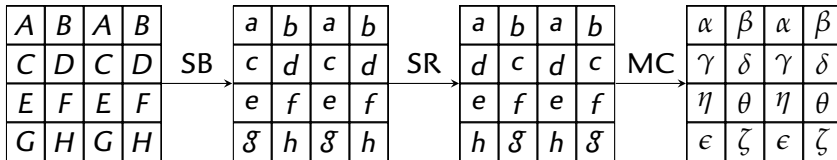
# Some Interesting Properties of AES [LSWD04]



## Some Interesting Properties of AES [LSWD04]



# Some Interesting Properties of AES [LSWD04]



## Some Interesting Properties of Lesamnta's $F$ and $G$

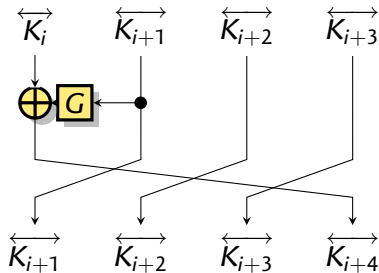
- ▶ *Lesamnta's*  $F$  posses similar properties:  

$$F(X, Y) = (Z, W) \Rightarrow F(Y, X) = (W, Z).$$
- ▶ The same is true for  $G$  as well:  

$$G(X, Y) = (Z, W) \Rightarrow G(Y, X) = (W, Z).$$
- ▶ Let  $\overleftarrow{(a, b)} = (a, b)$ 
  - ▶  $F(\overleftarrow{x}) = \overleftarrow{F(x)}$
  - ▶  $G(\overleftarrow{x}) = \overleftarrow{G(x)}$

## Complementation-like property in Lesamnta

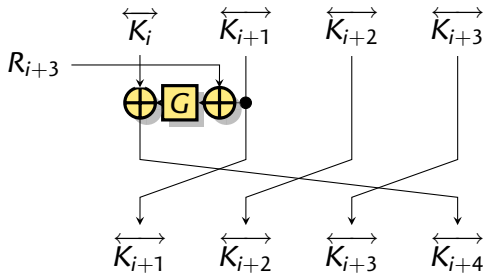
- ▶ Can we use this in the key-schedule?



- ▶ No, because of the constants
- ▶ On the other hand, the constants are almost symmetric...

## Complementation-like property in Lesamnta

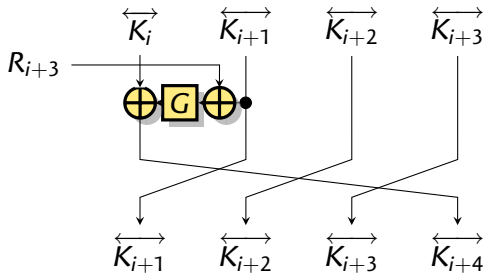
- ▶ Can we use this in the key-schedule?



- ▶ No, because of the **constants**
- ▶ On the other hand, the constants are almost symmetric...

## Complementation-like property in Lesamnta

- ▶ Can we use this in the key-schedule?



- ▶ No, because of the **constants**
- ▶ On the other hand, the constants are almost symmetric...

## Lesamnta's constants

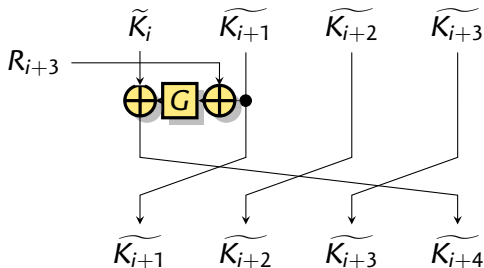
- ▶  $R_i = (2i, 2i + 1)$
- ▶  $R_i \oplus \overleftarrow{R}_i = (1, 1)$
- ▶ Let  $\widetilde{(a, b)} = \overleftarrow{(a, b)} \oplus (1, 1) = (b \oplus 1, a \oplus 1)$
- ▶  $\widetilde{\widetilde{R}}_i = R_i$

## Lesamnta's constants

- ▶  $R_i = (2i, 2i + 1)$
- ▶  $R_i \oplus \overleftarrow{R}_i = (1, 1)$
- ▶ Let  $\widetilde{(a, b)} = \overleftarrow{(a, b)} \oplus (1, 1) = (b \oplus 1, a \oplus 1)$
- ▶  $\widetilde{R}_i = R_i$

## Complementation-like property in Lesamnta, part II

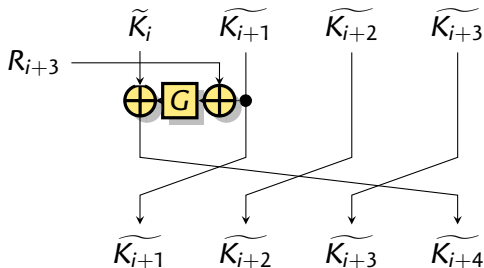
- ▶ Can we use this in the key-schedule?



- ▶  $\widetilde{K}_{i+1} \oplus R_{i+3} = \overleftarrow{K_{i+1} \oplus R_{i+3}}$
- ▶  $G(\widetilde{K}_{i+1} \oplus R_{i+3}) = \overleftarrow{G(K_{i+1} \oplus R_{i+3})}$
- ▶  $\widetilde{K}_i \oplus G(\widetilde{K}_{i+1} \oplus R_{i+3}) = K_i \oplus G(K_{i+1} \oplus R_{i+3}) = \widetilde{K}_{i+4}$

## Complementation-like property in Lesamnta, part II

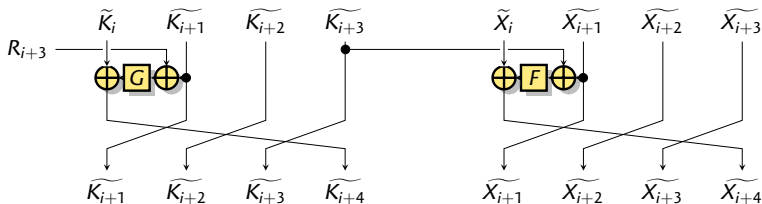
- ▶ Can we use this in the key-schedule?



- ▶  $\widetilde{K}_{i+1} \oplus R_{i+3} = \widetilde{K}_{i+1} \oplus R_{i+3}$
- ▶  $G(\widetilde{K}_{i+1} \oplus R_{i+3}) = G(\widetilde{K}_{i+1} \oplus R_{i+3})$
- ▶  $\widetilde{K}_i \oplus G(\widetilde{K}_{i+1} \oplus R_{i+3}) = K_i \oplus G(\widetilde{K}_{i+1} \oplus R_{i+3}) = \widetilde{K}_{i+4}$

## Complementation-like property in Lesamnta, part II

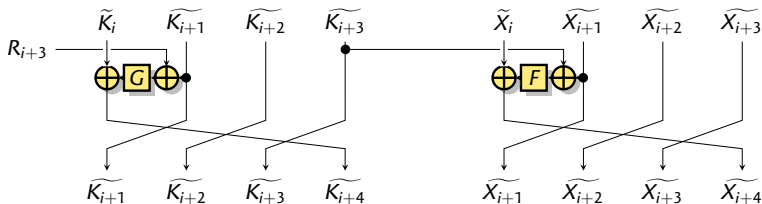
- ▶ Can we use this in the full compression function?



- ▶  $K_i \rightarrow \tilde{K}_i$
- ▶  $\tilde{X}_{i+1} \oplus \tilde{K}_{i+3} = \overleftarrow{X_{i+1} \oplus K_{i+3}}$
- ▶  $F(\tilde{X}_{i+1} \oplus \tilde{K}_{i+3}) = \overleftarrow{F(X_{i+1} \oplus K_{i+3})}$
- ▶  $\tilde{X}_i \oplus F(\tilde{X}_{i+1} \oplus \tilde{K}_{i+3}) = X_i \oplus F(\overleftarrow{X_{i+1} \oplus K_{i+3}}) = \tilde{X}_{i+4}$

## Complementation-like property in Lesamnta, part II

- ▶ Can we use this in the full compression function?



- ▶  $K_i \rightarrow \widetilde{K}_i$
- ▶  $\widetilde{X}_{i+1} \oplus \widetilde{K}_{i+3} = \overleftarrow{X_{i+1} \oplus K_{i+3}}$
- ▶  $F(\widetilde{X}_{i+1} \oplus \widetilde{K}_{i+3}) = \overleftarrow{F(X_{i+1} \oplus K_{i+3})}$
- ▶  $\widetilde{X}_i \oplus F(\widetilde{X}_{i+1} \oplus \widetilde{K}_{i+3}) = X_i \oplus F(\overleftarrow{X_{i+1} \oplus K_{i+3}}) = \widetilde{X}_{i+4}$

## Some Really Interesting Property of Lesamnta

- ▶  $CF(\tilde{X}, \tilde{K}) = \overleftrightarrow{CF(X, K)}$
- ▶ If  $\tilde{X} = X$  and  $\tilde{K} = K$ , then  $\overleftrightarrow{CF(X, K)} = CF(X, K)$ 
  - ▶ The output is in a subspace of size  $2^{n/2}$ .
- ▶ Collision in the compression function in time  $2^{n/4}$
- ▶ Second-preimage on weak messages
- ▶ Improved herding attack

## Some Really Interesting Property of Lesamnta

- ▶  $CF(\tilde{X}, \tilde{K}) = \overleftrightarrow{CF(X, K)}$
- ▶ If  $\tilde{X} = X$  and  $\tilde{K} = K$ , then  $\overleftrightarrow{CF(X, K)} = CF(X, K)$ 
  - ▶ The output is in a subspace of size  $2^{n/2}$ .
- ▶ Collision in the compression function in time  $2^{n/4}$
- ▶ Second-preimage on weak messages
- ▶ Improved herding attack

## Some Really Interesting Property of Lesamnta

- ▶  $CF(\tilde{X}, \tilde{K}) = \overleftrightarrow{CF(X, K)}$
- ▶ If  $\tilde{X} = X$  and  $\tilde{K} = K$ , then  $\overleftrightarrow{CF(X, K)} = CF(X, K)$ 
  - ▶ The output is in a subspace of size  $2^{n/2}$ .
- ▶ Collision in the compression function in time  $2^{n/4}$
- ▶ Second-preimage on weak messages
- ▶ Improved herding attack

## *Self-similarity property*

- ▶ Sometimes, simple relation can go through a function
- ▶ The constants are used to avoid this...
  - ▶ But sometimes the constants are weak

## Cancellation cryptanalysis on generalized Feistels

- ▶ Cancel the effect of the non-linear components  
Using twice the same input pairs
- ▶ Fix some parts of the state to reduce the diffusion



Charles Bouillaguet, 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>



Praveen Gauravaram, Gaëtan Leurent, Florian Mendel, María Naya-Plasencia, Thomas Peyrin, Christian Rechberger, and Martin Schläffer

Cryptanalysis of the 10-Round Hash and Full Compression Function of *SHAvite-3*<sub>512</sub>







## Feistel design

- ▶ Ideal: each  $F_i$  is an independent ideal function/permutation
- ▶ In practice:  $F_i(x) = F(k_i \oplus x)$  with a **fixed**  $F$

*Properties of  $F_i(x) = F(k_i \oplus x)$*

- (i)  $\exists c_{ij} : \forall x, F_i(x \oplus c_{ij}) = F_j(x)$ .
- (ii)  $\forall \alpha, \#\{x : F_i(x) \oplus F_j(x) = \alpha\}$  is even
- (iii)  $\bigoplus_x F_k(F_i(x) \oplus F_j(x)) = 0$

- ▶  $c_{ij} = k_i \oplus k_j$

## Feistel design

- ▶ Ideal: each  $F_i$  is an independent ideal function/permutation
- ▶ In practice:  $F_i(x) = F(k_i \oplus x)$  with a **fixed**  $F$

*Properties of  $F_i(x) = F(k_i \oplus x)$*

- (i)  $\exists c_{ij} : \forall x, F_i(x \oplus c_{ij}) = F_j(x)$ .
- (ii)  $\forall \alpha, \#\{x : F_i(x) \oplus F_j(x) = \alpha\}$  is even
- (iii)  $\bigoplus_x F_k(F_i(x) \oplus F_j(x)) = 0$

- ▶  $c_{ij} = k_i \oplus k_j$

## Feistel design

- ▶ Ideal: each  $F_i$  is an independent ideal function/permutation
- ▶ In practice:  $F_i(x) = F(k_i \oplus x)$  with a **fixed**  $F$

### Properties of $F_i(x) = F(k_i \oplus x)$

- (i)  $\exists c_{i,j} : \forall x, F_i(x \oplus c_{i,j}) = F_j(x)$ .
- (ii)  $\forall \alpha, \#\{x : F_i(x) \oplus F_j(x) = \alpha\}$  is even
- (iii)  $\bigoplus_x F_k(F_i(x) \oplus F_j(x)) = 0$

▶  $c_{ij} = k_i \oplus k_j$

## Feistel design

- ▶ Ideal: each  $F_i$  is an independent ideal function/permutation
- ▶ In practice:  $F_i(x) = F(k_i \oplus x)$  with a **fixed**  $F$

*Properties of  $F_i(x) = F(k_i \oplus x)$*

- (i)  $\exists c_{ij} : \forall x, F_i(x \oplus c_{ij}) = F_j(x)$ .
- (ii)  $\forall \alpha, \#\{x : F_i(x) \oplus F_j(x) = \alpha\}$  is even
- (iii)  $\bigoplus_x F_k(F_i(x) \oplus F_j(x)) = 0$

- ▶  $c_{ij} = k_i \oplus k_j$

## The cancellation property

$i$	$S_i$	$T_i$	$U_i$	$V_i$
0	$a$	$b$	$c$	$d$
1	$F_0(c) \oplus d$	$a$	$b$	$c$
2	$F_1(b) \oplus c$	$F_0(c) \oplus d$	$a$	$b$
3	$F_2(a) \oplus b$	$F_1(b) \oplus c$	$F_0(c) \oplus d$	$a$
4	$F_3(F_0(c) \oplus d) \oplus a$	$F_2(a) \oplus b$	$F_1(b) \oplus c$	$F_0(c) \oplus d$
5	$F_4(F_1(b) \oplus c) \oplus F_0(c) \oplus d$	$F_3(F_0(c) \oplus d) \oplus a$	$F_2(a) \oplus b$	$F_1(b) \oplus c$

round 5  $F_4(F_1(b) \oplus c) \oplus F_0(c)$

Cancel if  $F_1(b) = K_0 \oplus K_4$

$$\Rightarrow b \stackrel{\Delta}{=} F_1^{-1}(K_0 \oplus K_4)$$

- ▶ If  $b$  is fixed to the right value, simple expressions.
- ▶ Easy in hash function.

## The cancellation property

$i$	$S_i$	$T_i$	$U_i$	$V_i$
0	$a$	$b$	$c$	$d$
1	$F_0(c) \oplus d$	$a$	$b$	$c$
2	$F_1(b) \oplus c$	$F_0(c) \oplus d$	$a$	$b$
3	$F_2(a) \oplus b$	$F_1(b) \oplus c$	$F_0(c) \oplus d$	$a$
4	$F_3(F_0(c) \oplus d) \oplus a$	$F_2(a) \oplus b$	$F_1(b) \oplus c$	$F_0(c) \oplus d$
5	<del><math>F_4(F_1(b) \oplus c) \oplus F_0(c)</math></del>	$F_3(F_0(c) \oplus d) \oplus a$	$F_2(a) \oplus b$	$F_1(b) \oplus c$

round 5  $F_4(F_1(b) \oplus c) \oplus F_0(c)$

Cancel if  $F_1(b) = K_0 \oplus K_4$

$$\Rightarrow b \stackrel{\Delta}{=} F_1^{-1}(K_0 \oplus K_4)$$

- ▶ If  $b$  is fixed to the right value, simple expressions.
- ▶ Easy in hash function.

## The cancellation property

$i$	$S_i$	$T_i$	$U_i$	$V_i$
0	$a$	$b$	$c$	$d$
1	$F_0(c) \oplus d$	$a$	$b$	$c$
2	$F_1(b) \oplus c$	$F_0(c) \oplus d$	$a$	$b$
3	$F_2(a) \oplus b$	$F_1(b) \oplus c$	$F_0(c) \oplus d$	$a$
4	$F_3(F_0(c) \oplus d) \oplus a$	$F_2(a) \oplus b$	$F_1(b) \oplus c$	$F_0(c) \oplus d$
5	<del><math>F_4(F_1(b) \oplus c) \oplus F_0(c)</math></del> $\oplus d$	$F_3(F_0(c) \oplus d) \oplus a$	$F_2(a) \oplus b$	$F_1(b) \oplus c$

round 5  $F_4(F_1(b) \oplus c) \oplus F_0(c)$

Cancel if  $F_1(b) = K_0 \oplus K_4$

$$\Rightarrow b \stackrel{\Delta}{=} F_1^{-1}(K_0 \oplus K_4)$$

- ▶ If  $b$  is fixed to the right value, simple expressions.
- ▶ Easy in hash function.

## The cancellation property

$i$	$S_i$	$T_i$	$U_i$	$V_i$
0	$a$	$b$	$c$	$d$
1	$F_0(c) \oplus d$	$a$	$b$	$c$
2	$F_1(b) \oplus c$	$F_0(c) \oplus d$	$a$	$b$
3	$F_2(a) \oplus b$	$F_1(b) \oplus c$	$F_0(c) \oplus d$	$a$
4	$F_3(F_0(c) \oplus d) \oplus a$	$F_2(a) \oplus b$	$F_1(b) \oplus c$	$F_0(c) \oplus d$
5	<del><math>F_4(F_1(b) \oplus c) \oplus F_0(c)</math></del> $\oplus d$	$F_3(F_0(c) \oplus d) \oplus a$	$F_2(a) \oplus b$	$F_1(b) \oplus c$

round 5  $F_4(F_1(b) \oplus c) \oplus F_0(c)$

Cancel if  $F_1(b) = K_0 \oplus K_4$

$$\Rightarrow b \stackrel{\Delta}{=} F_1^{-1}(K_0 \oplus K_4)$$

- ▶ If  $b$  is fixed to the right value, simple expressions.
- ▶ Easy in hash function.

## Attack Overview

- ▶ Choose one part of the output
  - ▶ Preimage and collision attacks.
- ▶ Mostly generic in the round function.

### Basic algorithm

- ▶ Start from a state in the middle
- ▶ Fix some parts of the state to satisfy the cancellation conditions.
- ▶ One output word will have a relatively simple expression.
- ▶ Invert the expression to choose one word of the output.

## Result overview

- ▶ Attacks on reduced *Lesamnta*
  - ▶ 24 rounds out of 32: collision and preimage
  - ▶ previous attacks: 16 rounds
- ▶ Attack on reduced *SHAvite-3*<sub>512</sub>
  - ▶ 10 rounds out of 14: preimage
  - ▶ previous attacks: 8 rounds
- ▶ Pseudo-attack on full *SHAvite-3*<sub>512</sub> compression function
  - ▶ chosen-salt chosen-counter preimage

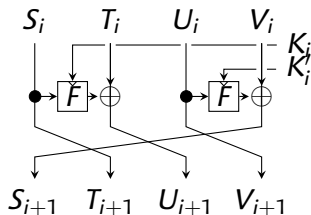
## Result overview

- ▶ Attacks on reduced *Lesamnta*
  - ▶ 24 rounds out of 32: collision and preimage
  - ▶ previous attacks: 16 rounds
- ▶ Attack on reduced *SHAvite-3*<sub>512</sub>
  - ▶ 10 rounds out of 14: preimage
  - ▶ previous attacks: 8 rounds
- ▶ Pseudo-attack on full *SHAvite-3*<sub>512</sub> compression function
  - ▶ chosen-salt chosen-counter preimage

## Result overview

- ▶ Attacks on reduced *Lesamnta*
  - ▶ 24 rounds out of 32: collision and preimage
  - ▶ previous attacks: 16 rounds
- ▶ Attack on reduced *SHAvite-3*<sub>512</sub>
  - ▶ 10 rounds out of 14: preimage
  - ▶ previous attacks: 8 rounds
- ▶ Pseudo-attack on full *SHAvite-3*<sub>512</sub> compression function
  - ▶ chosen-salt chosen-counter preimage

## SHAvite-3<sub>512</sub>



- ▶ 14 rounds
- ▶ Davies-Meyer (message is the key)
- ▶  $F_i(x) = AES(AES(AES(AES(x \oplus k_i^0) \oplus k_i^1) \oplus k_i^2) \oplus k_i^3)$



Eli Biham and Orr Dunkelman

The SHAvite-3 Hash Function

Submission to the NIST SHA-3 competition

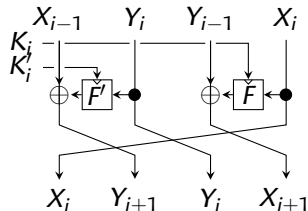
# Cancellation differential path: SHAvite-3<sub>512</sub>

$i$	$S_i$	$T_i$	$U_i$	$V_i$
-4	?	$x$	?	$x_2$
-3	$x_2$	$x_1$	$x$	-
-2	-	-	$x_1$	$x$
-1	$x$	-	-	-
0	-	$x$	-	-
1	-	$y$	$x$	-
2	-	$z$	$y$	$x$
3	$x$	$w$	$z$	-
4	-	?	$w$	$z$
5	$z$	?	?	-
FF	?	?	?	$x_2$

$x \rightarrow y$

$x \rightarrow y, z \rightarrow w$

$z \rightarrow w$



▶ Same attack as previously

▶ But...

▶  $F$  has many keys...

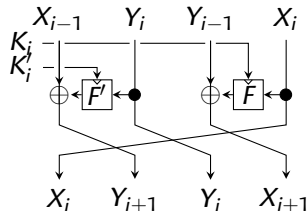
# Cancellation differential path: SHAvite-3<sub>512</sub>

$i$	$S_i$	$T_i$	$U_i$	$V_i$
-4	?	$x$	?	$x_2$
-3	$x_2$	$x_1$	$x$	-
-2	-	-	$x_1$	$x$
-1	$x$	-	-	-
0	-	$x$	-	-
1	-	$y$	$x$	-
2	-	$z$	$y$	$x$
3	$x$	$w$	$z$	-
4	-	?	$w$	$z$
5	$z$	?	?	-
FF	?	?	?	$x_2$

$x \rightarrow y$

$x \rightarrow y, z \rightarrow w$

$z \rightarrow w$



▶ Same attack as previously

▶ But...

▶  $F$  has many keys...

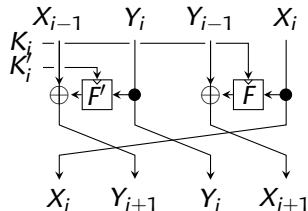
# Cancellation differential path: SHAvite-3<sub>512</sub>

$i$	$S_i$	$T_i$	$U_i$	$V_i$
-4	?	$x$	?	$x_2$
-3	$x_2$	$x_1$	$x$	-
-2	-	-	$x_1$	$x$
-1	$x$	-	-	-
0	-	$x$	-	-
1	-	$y$	$x$	-
2	-	$z$	$y$	$x$
3	$x$	$w$	$z$	-
4	-	?	$w$	$z$
5	$z$	?	?	-
FF	?	?	?	$x_2$

$x \rightarrow y$

$x \rightarrow y, z \rightarrow w$

$z \rightarrow w$



- ▶ Same attack as previously
- ▶ But...
- ▶  $F$  has many keys...







## Attacking the key schedule

- ▶ We can build a chaining value satisfying the 6 conditions with cost  $2^{224}$ .
- ▶ Each chaining value can be used  $2^{128}$  times to fix 128 bits of the output.
- ▶ Attacks on 9-round *SHAvite-3*<sub>512</sub>:
  - ▶ Free-start preimage with complexity  $2^{480}$
  - ▶ Preimage with complexity  $2^{497}$ .

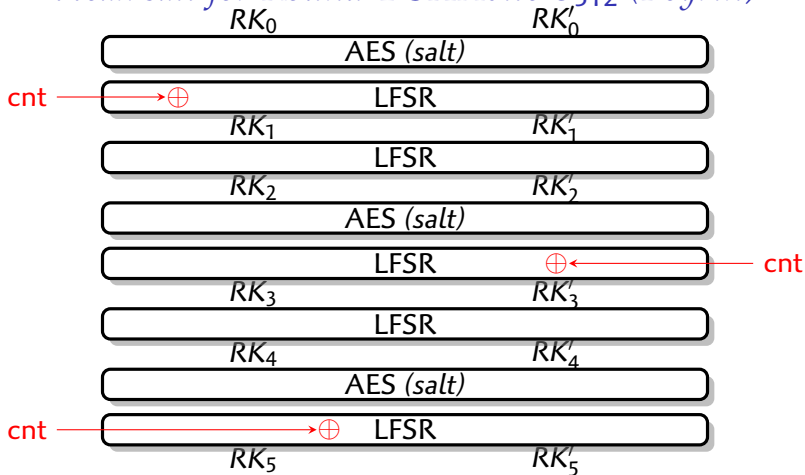
## Adding more rounds

$i$	$A_i$	$B_i$	$C_i$	$D_i$	conditions
3	?	$B_3$	?	?	
4	?	?	$B_3$	$D_4$	
5	$D_4$	$B_5$	?	$B_3 + F'_4(D_4)$	$F_5(B_5) = 0$
6	$B_5 + F'_4(D_4)$	$D_4$	$B_3$	$D_6$	$RK_6 = RK'_4$
7	$D_6$	$B_3$	$D_4$	$B_5 + F'_6(D_6)$	$F_7(B_3) = 0$
8	$B_3 + F'_6(D_6)$	$D_6$	$B_3$	$D_8$	$RK_8 = RK'_6$
9	$D_8$	$B_5$	$D_6$	$B_3 + F'_8(D_8)$	$RK_9 = RK_5$
10	$B_5 + F'_8(D_8)$	$D_8$	$B_5$	$D_{10}$	$RK_{10} = RK'_8$
11	$D_{10}$	$B_3$	$D_8$	$B_5 + F'_{10}(D_{10})$	$RK_{11} = RK_7$

- ▶ Only two conditions on the state
- ▶ Many conditions on the key

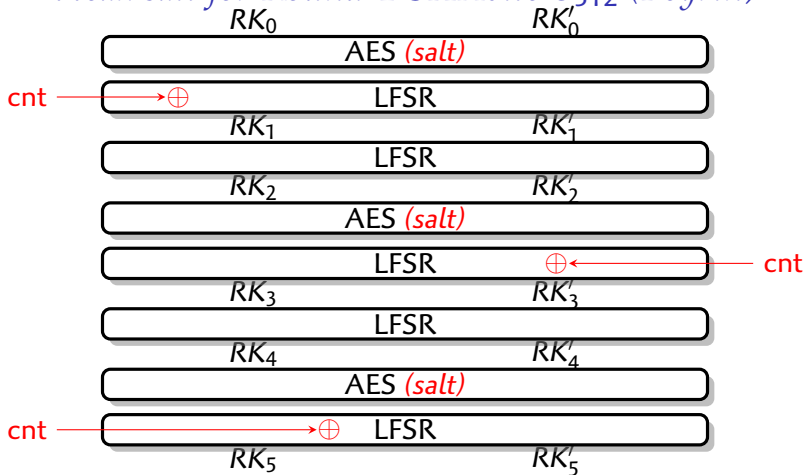


## Weak salt for Round-1 SHAvite-3<sub>512</sub> (Peyrin)



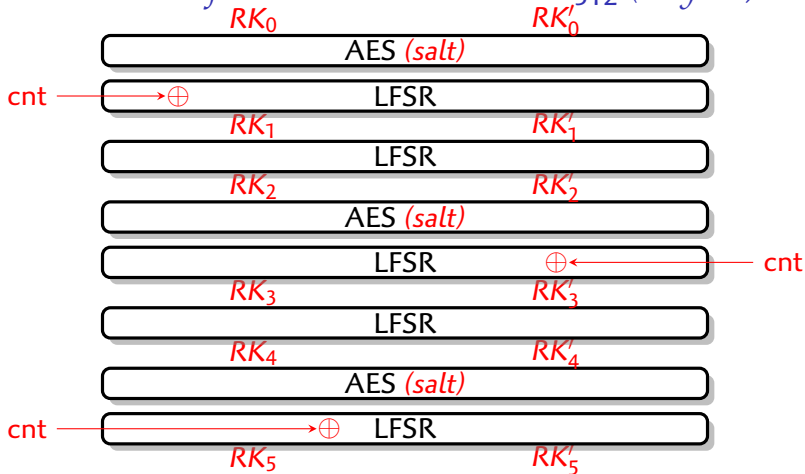
- ▶ Take the zero counter;
- ▶ Take the salt that sends zero to zero;
- ▶ Use the zero message: all the subkeys are zero.

## Weak salt for Round-1 SHAvite-3<sub>512</sub> (Peyrin)



- ▶ Take the zero counter;
- ▶ Take the salt that sends zero to zero;
- ▶ Use the zero message: all the subkeys are zero.

## Weak salt for Round-1 SHAvite-3<sub>512</sub> (Peyrin)



- ▶ Take the zero counter;
- ▶ Take the salt that sends zero to zero;
- ▶ Use the zero message: all the subkeys are zero.









## Weak salt for Round-2 SHAvite-3<sub>512</sub>

$i$	$RK_i$				$RK'_i$				$r$
	$k_{0,i}^0$	$k_{0,i}^1$	$k_{0,i}^2$	$k_{0,i}^3$	$k_{1,i}^0$	$k_{1,i}^1$	$k_{1,i}^2$	$k_{1,i}^3$	
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	?★	?	

## 14-round attack

**Input:** Target value H

**Output:** message, chaining value, salt, counter

1: **repeat**

2:     Take a random weak salt, and the corresponding message

3:     Compute  $2^{128}$  states with 128 chosen output bits

4: **until** a full preimage is found ( $2^{256}$  iterations)

- ▶ Pseudo-preimage attack: complexity  $2^{384}$  and  $2^{128}$  memory
- ▶ Pseudo-preimage attack: complexity  $2^{448}$  without memory
- ▶ Pseudo-collision attack: complexity  $2^{192}$  and  $2^{128}$  memory.

## 14-round attack

**Input:** Target value H

**Output:** message, chaining value, salt, counter

1: **repeat**

2:     Take a random weak salt, and the corresponding message

3:     Compute  $2^{128}$  states with 128 chosen output bits

4: **until** a full preimage is found ( $2^{256}$  iterations)

- ▶ Pseudo-preimage attack: complexity  $2^{384}$  and  $2^{128}$  memory
- ▶ Pseudo-preimage attack: complexity  $2^{448}$  without memory
- ▶ Pseudo-collision attack: complexity  $2^{192}$  and  $2^{128}$  memory.

# Questions?

Thank you for your attention!