Rebound Attack

Florian Mendel

Institute for Applied Information Processing and Communications (IAIK)
Graz University of Technology
Inffeldgasse 16a, A-8010 Graz, Austria

http://www.iaik.tugraz.at/

Outline

SHA-3 competition

Abacus	ECHO	Lesamnta	SHAMATA
ARIRANG	ECOH	Luffa	SHAvite-3
AURORA	Edon-R	LUX	SIMD
BLAKE	EnRUPT	Maraca	Skein
Blender	ESSENCE	MCSSHA-3	Spectral Hash
Blue Midnight Wish	FSB	MD6	StreamHash
Boole	Fugue	MeshHash	SWIFFTX
Cheetah	Grøstl	NaSHA	Tangle
CHI	Hamsi	NKS2D	TIB3
CRUNCH	HASH 2X	Ponic	Twister
CubeHash	JH	SANDstorm	Vortex
DCH	Keccak	Sarmal	WaMM
Dynamic SHA	Khichidi-1	Sgàil	Waterfall
Dynamic SHA2	LANE	Shabal	ZK-Crypt

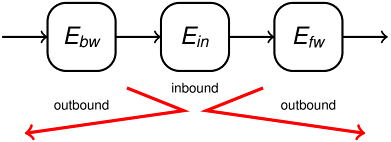
SHA-3 competition

Abacus	ECHO	Lesamnta	SHAMATA
ARIRANG	ECOH	Luffa	SHAvite-3
AURORA	Edon-R	LUX	SIMD
BLAKE	EnRUPT	Maraca	Skein
Blender	ESSENCE	MCSSHA-3	Spectral Hash
Blue Midnight Wish	FSB	MD6	StreamHash
Boole	Fugue	MeshHash	SWIFFTX
Cheetah	Grøstl	NaSHA	Tangle
CHI	Hamsi	NKS2D	TIB3
CRUNCH	HASH 2X	Ponic	Twister
CubeHash	JH	SANDstorm	Vortex
DCH	Keccak	Sarmal	WaMM
Dynamic SHA	Khichidi-1	Sgàil	Waterfall
Dynamic SHA2	LANE	Shabal	ZK-Crypt

The Rebound Attack [MRST09]

- Tool in the differential cryptanalysis of hash functions
- Invented during the design of Grøstl
  - AES-based designs allow a simple application of the idea
- Has been applied to a wide range of hash functions
  - Echo, Grøstl, JH, Lane, Luffa, Maelstrom, Skein, Twister, Whirlpool, ...

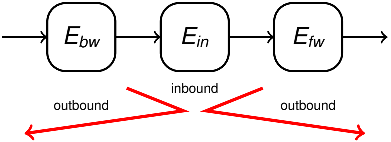
The Rebound Attack



- Applies to block cipher and permutation based designs:

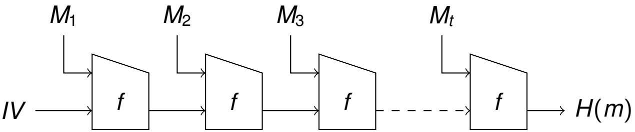
$$E = E_{fw} \circ E_{in} \circ E_{bw} \qquad\qquad P = P_{fw} \circ P_{in} \circ P_{bw}$$

The Rebound Attack



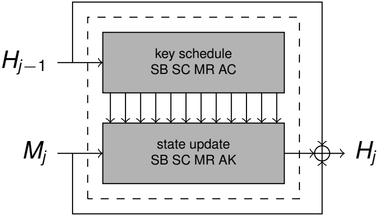
- Inbound phase
  - efficient meet-in-the-middle phase in E_{in}
  - using available degrees of freedom
- Outbound phase
  - probabilistic part in E_{bw} and E_{fw}
  - repeat inbound phase if needed

The Whirlpool Hash Function



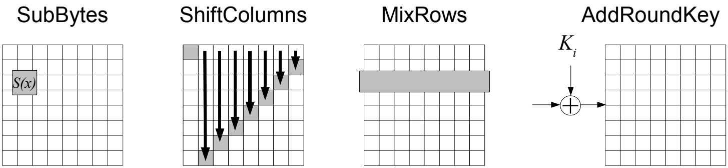
- designed by Barretto and Rijmen [BR00]
  - evaluated by NESSIE
  - standardized by ISO/IEC 10118-3:2003
- iterative, based on the Merkle-Damgård design principle
- message block, chaining values, hash size: 512 bit

The Whirlpool Compression Function



- 512-bit hash value and using 512-bit message blocks
- Block-cipher based design (similar to AES)
  - Miyaguchi-Preneel mode with conservative key schedule

The Whirlpool Round Transformations

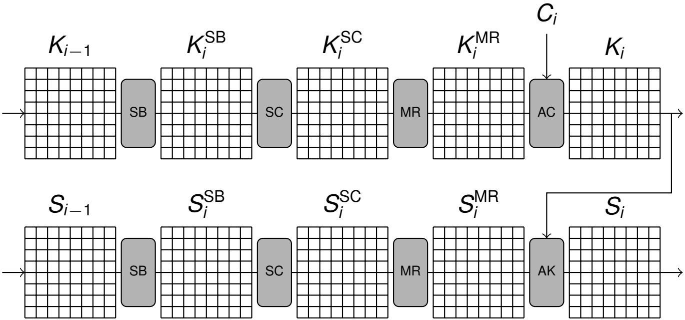


- The state update and the key schedule update an 8×8 state S and K of 64 bytes
- 10 rounds each
- AES like round transformation

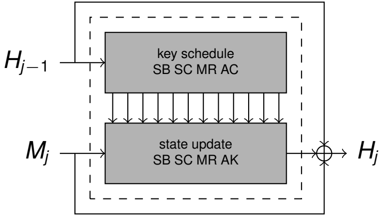
$$r_i = AK \circ MR \circ SC \circ SB$$

Notations

- Round i



Collision Attack on Whirlpool

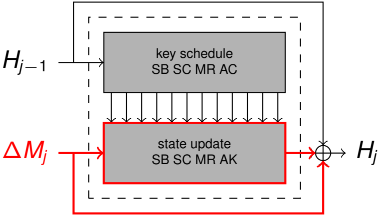


1-block collision:

- fixed H_{j-1} (to IV)
- $f(M_j, H_{j-1}) = f(M_j^*, H_{j-1}),\ M_j \neq M_j^*$

generic complexity 2^{256} ($n = 512$)

Collision Attack on Whirlpool

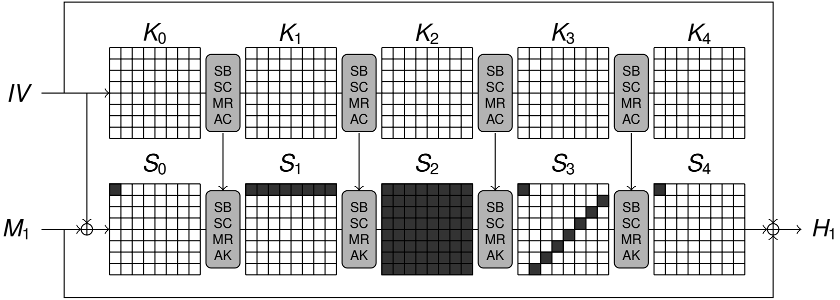


1-block collision:

- fixed H_{j-1} (to IV)
- $f(M_j, H_{j-1}) = f(M_j^*, H_{j-1}),\ M_j \neq M_j^*$

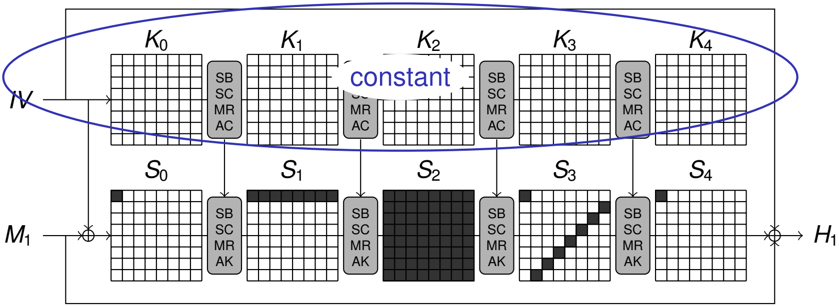
generic complexity 2^{256} ($n = 512$)

Collision Attack on 4 Rounds



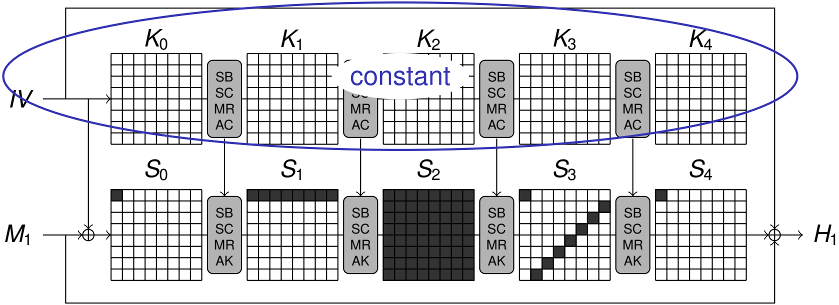
- Differential trail with minimum number of active S-boxes
  - 81 for any 4-round trail ($1 \rightarrow 8 \rightarrow 64 \rightarrow 8$)
  - maximum differential probability: $(2^{-5})^{81} = 2^{-405}$

Collision Attack on 4 Rounds



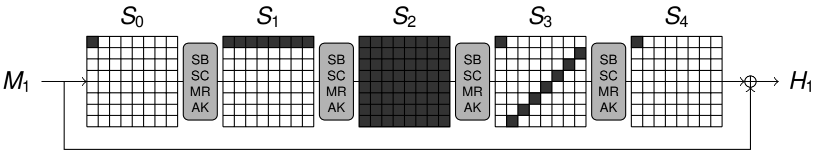
- Differential trail with minimum number of active S-boxes
  - 81 for any 4-round trail ($1 \to 8 \to 64 \to 8$)
  - maximum differential probability: $(2^{-5})^{81} = 2^{-405}$

Collision Attack on 4 Rounds



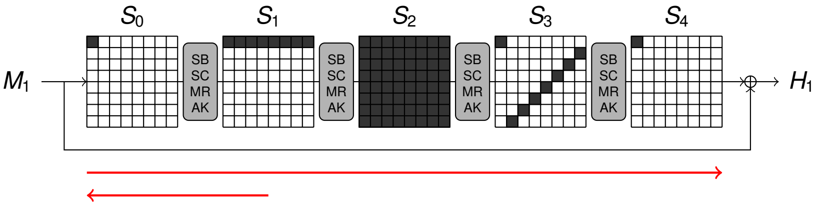
- Differential trail with minimum number of active S-boxes
  - 81 for any 4-round trail ($1 \to 8 \to 64 \to 8$)
  - maximum differential probability: $(2^{-5})^{81} = 2^{-405}$
- How to find a message pair following the differential trail?

First: Use Truncated Differences



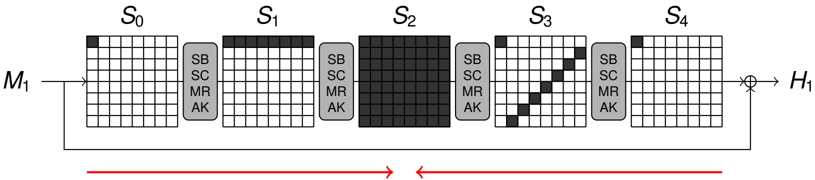
- byte-wise truncated differences: active / not active
  - we do not mind about actual differences
  - single active byte at input and output is enough
  - probabilistic in MixRows: 2^{-56} for $8 \to 1$
- we can remove many restrictions (more freedom)
  - hopefully less complexity of message search

How to Find a Message Pair?



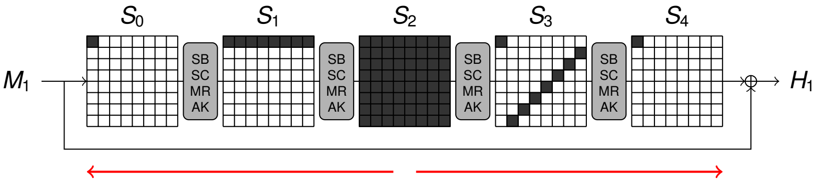
- message modification?

How to Find a Message Pair?



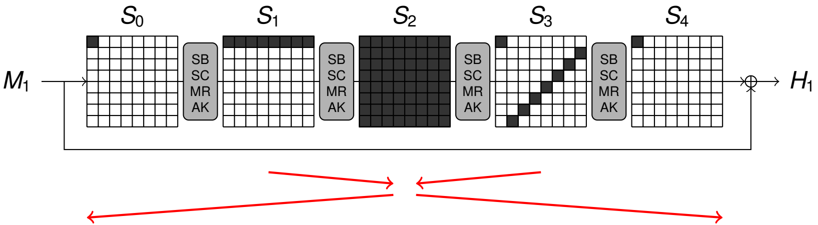
- message modification?
- meet in the middle?

How to Find a Message Pair?



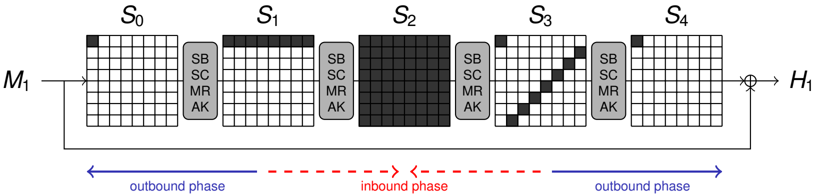
- message modification?
- meet in the middle?
- inside out?

How to Find a Message Pair?



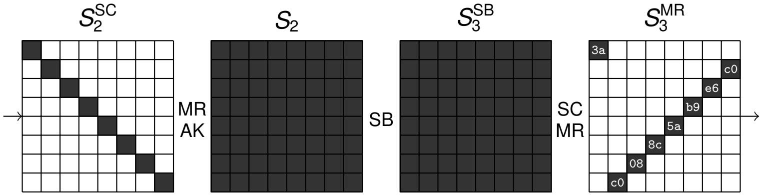
- message modification?
- meet in the middle?
- inside out?
- rebound!

Rebound Attack on 4 Rounds [MRST09]



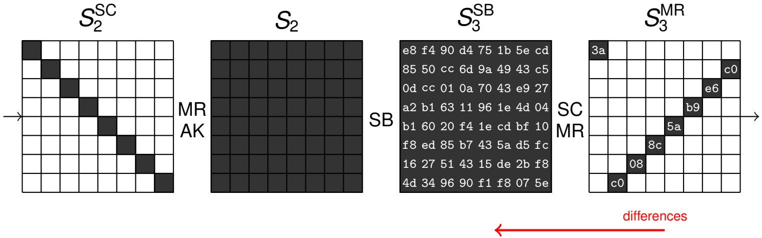
- **Inbound phase**
  1. (1) start with differences in round 2 and 3
  2. (2) match-in-the-middle at S-box using values of the state
- **Outbound phase**
  3. (3) probabilistic propagation in MixRows in round 1 and 4
  4. (4) match one-byte difference of feed-forward

Inbound Phase



(1) Start with arbitrary differences in state S_3^{MR}

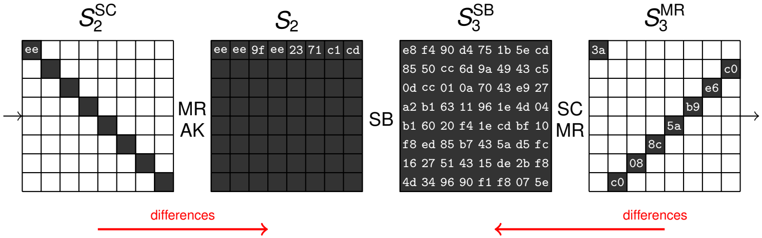
Inbound Phase



(1) Start with arbitrary differences in state S_3^{MR}

- linearly propagate all differences backward to S_3^{SB}

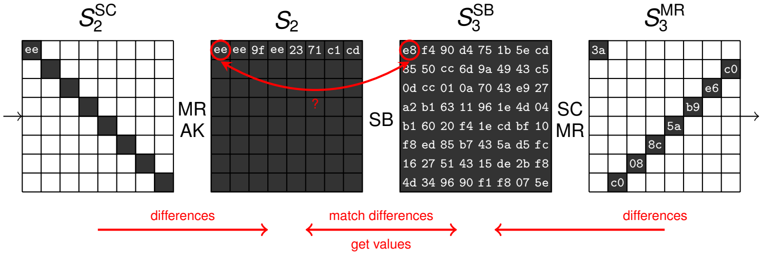
Inbound Phase



(1) Start with arbitrary differences in state S_3^{MR}

- linearly propagate all differences backward to S_3^{SB}
- linearly propagate row-wise forward from S_2^{SC} to S_2

Inbound Phase



(1) Start with arbitrary differences in state S_3^{MR}

- linearly propagate all differences backward to S_3^{SB}
- linearly propagate row-wise forward from S_2^{SC} to S_2

(2) Match-in-the-middle at SubBytes layer

- check if differences can be connected (for each S-box)

Match-in-the-Middle for Single S-box

$$\boxed{\Delta a} \longrightarrow Sbox \longrightarrow \boxed{\Delta b}$$

- Check for matching input/output differences

$$Sbox(x) \oplus Sbox(x \oplus \Delta a) = \Delta b$$

- Use Difference Distribution Table (DDT)

Difference Distribution Table (Whirlpool)

in \ out		10	11	12	13	14	15	16	17	18	19	1a	1b	1c	1d	1e	1f	
00		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
01		0	6	2	0	0	6	2	0	0	0	4	0	0	0	0	0	
02		0	0	0	0	0	0	0	2	0	0	0	0	4	0	0	0	
03		0	2	2	0	2	2	0	0	2	0	0	0	0	0	0	2	
04		0	0	2	2	0	4	0	0	0	2	2	2	2	0	2	0	
05		0	0	0	0	0	2	0	2	0	0	0	0	0	0	4	2	
06	.	4	0	2	0	0	2	0	0	2	6	2	4	0	2	2	0	.
07	.	0	2	2	0	0	2	0	0	4	0	2	0	2	0	2	0	.
08	.	0	0	0	0	2	2	2	0	0	0	0	2	2	4	4	0	.
09		8	0	0	0	2	4	2	2	0	0	0	0	0	2	0	2	
0a		0	0	0	0	2	0	2	0	2	0	2	0	0	0	0	0	
0b		8	2	2	2	2	0	0	0	0	2	2	2	2	2	0	4	
0c		0	2	2	0	0	0	0	4	0	2	2	0	0	2	4	2	
0d		0	2	2	0	0	2	4	4	0	0	2	2	0	0	0	2	
0e		4	0	4	2	0	0	0	0	2	0	2	0	4	2	0	0	
0f		0	2	0	0	0	2	0	0	0	0	0	0	2	0	2	2	
								.	.	.								

Differences can be connected if there is a non-zero entry in the table

Match-in-the-Middle for Single S-box

- Check for matching input/output differences

$$Sbox(x) \oplus Sbox(x \oplus \Delta a) = \Delta b$$

- Using Difference Distribution Table (DDT)
- Solve equation for all x and count the number of solutions

Difference Distribution Table (Whirlpool)

- The number of differentials and possible pairs for the Sbox

solutions	frequency
0	39655
2	20018
4	5043
6	740
8	79
256	1

- 25880/65025 entries (with $\Delta a, \Delta b \neq 0$) in DDT are nonzero
- we get either 2, 4, 6 or 8 values for each match
- $\frac{25880}{65025} \cdot \frac{65280}{25880} = 1.004$ values (right pairs) on average

Match-in-the-Middle for Single S-box

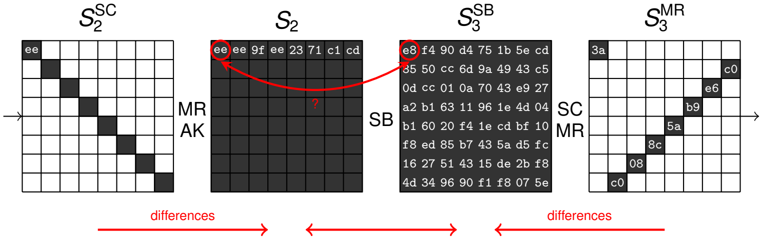
$$\Delta a \longrightarrow \ Sbox \ \longrightarrow \Delta b$$

- Check for matching input/output differences
- Using Difference Distribution Table (DDT)

$$Sbox(x) \oplus Sbox(x \oplus \Delta a) = \Delta b$$

- Solve equation for all x and count the number of solutions.
  - $\sim$ 1 value (right pair) on average

Inbound Phase



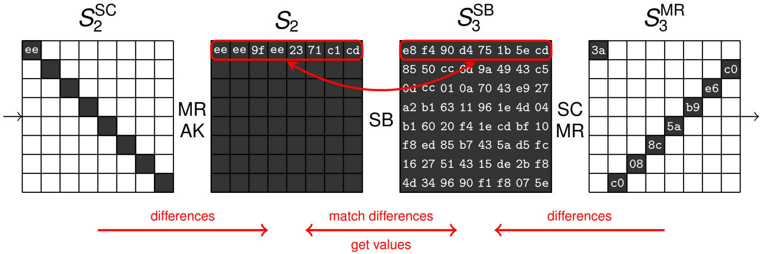
(1) Start with arbitrary differences in state S_3^{MR}

- linearly propagate all differences backward to S_3^{SB}
- linearly propagate row-wise forward from S_2^{SC} to S_2

(2) Match-in-the-middle at SubBytes layer

- check if differences can be connected (for each S-box)

Inbound Phase



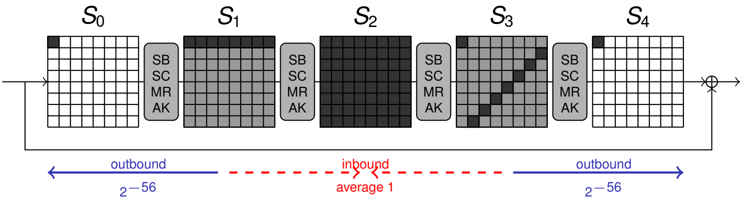
(1) Start with arbitrary differences in state S_3^{MR}

- linearly propagate all differences backward to S_3^{SB}
- linearly propagate row-wise forward from S_2^{SC} to S_2

(2) Match-in-the-middle at SubBytes layer

- check if differences can be connected (for each S-box)
- we need to solve each row at once: complexity $\sim 2^{10.6}$ (average 1)

Outbound Phase



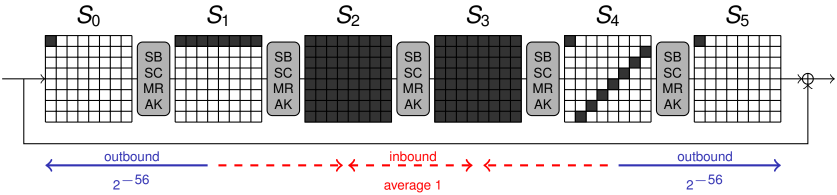
(3) Propagate through MixRows of round 1 and round 4

- using truncated differences (active bytes: $8 \rightarrow 1$)
- probability: 2^{-56} in each direction

(4) Match difference in one active byte of feed-forward (2^{-8})

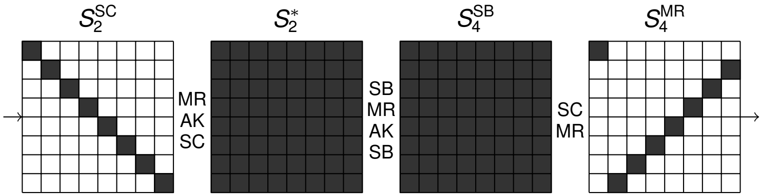
$\Rightarrow$ collision for 4 rounds of Whirlpool with complexity 2^{120}

Extending the Attack to 5 Rounds [LMR+09]



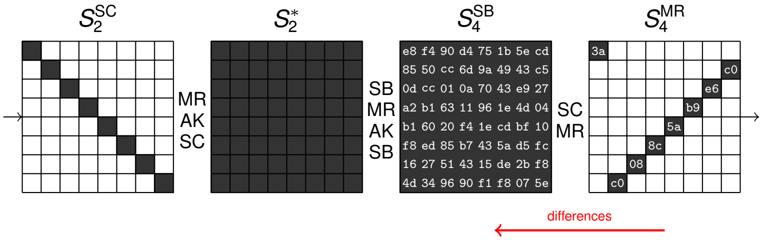
- By adding one round in the inbound phase of the attack we can extend the attack to 5 rounds
- The outbound phase is identical to the attack on 4 rounds
  - probability: 2^{-120}

Inbound Phase



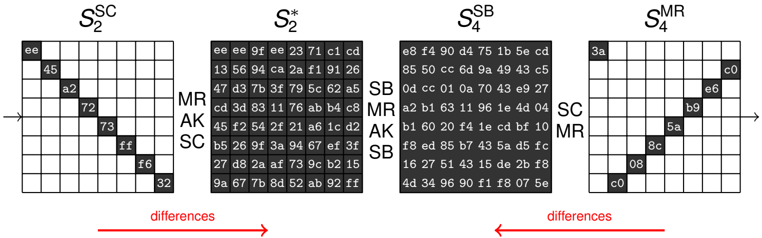
(1) Start with arbitrary differences in state S_2^{SC} and S_4^{MR}

Inbound Phase



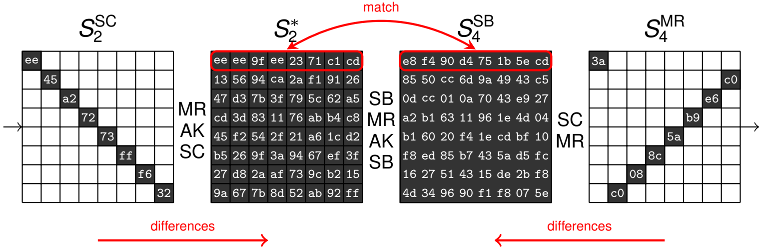
(1) Start with arbitrary differences in state S_2^{SC} and S_4^{MR}

Inbound Phase



(1) Start with arbitrary differences in state S_2^{SC} and S_4^{MR}

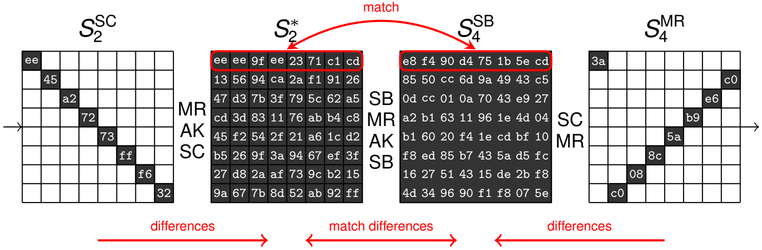
Inbound Phase



(1) Start with arbitrary differences in state S_2^{SC} and S_4^{MR}

(2) Match-in-the-middle at SuperBox (SB – MR – AK – SB)

Inbound Phase

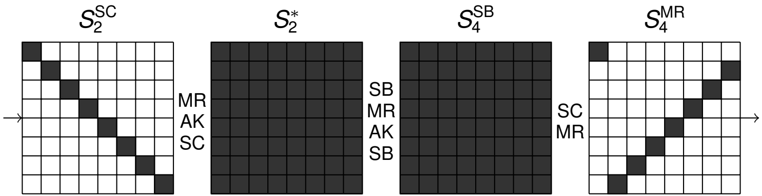


(1) Start with arbitrary differences in state S_2^{SC} and S_4^{MR}

(2) Match-in-the-middle at SuperBox (SB – MR – AK – SB)

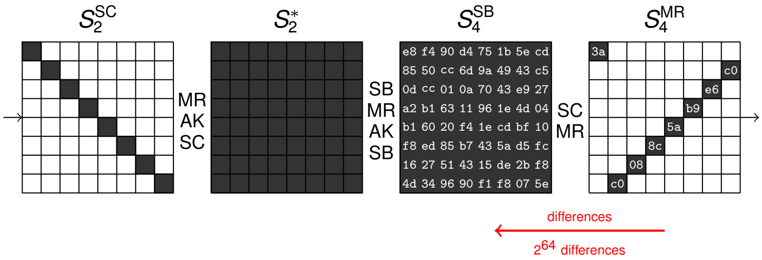
- similar to 64-bit S-box (DDT has size 2^{128})

Inbound Phase



(1) Start with arbitrary differences in state S_2^{SC} and S_4^{MR}

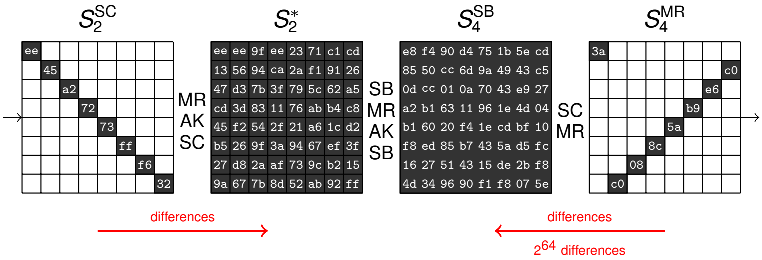
Inbound Phase



(1) Start with arbitrary differences in state S_2^{SC} and S_4^{MR}

- we propagate all 2^{64} differences backward at once

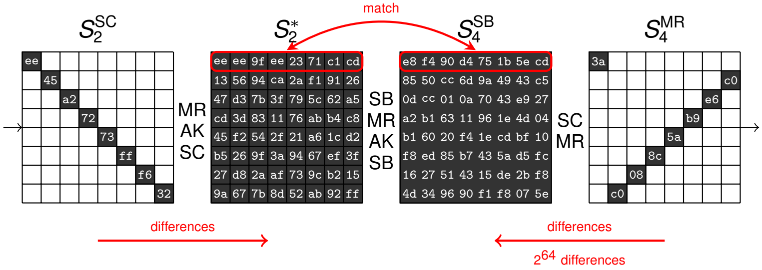
Inbound Phase



(1) Start with arbitrary differences in state S_2^{SC} and S_4^{MR}

- we propagate all 2^{64} differences backward at once

Inbound Phase

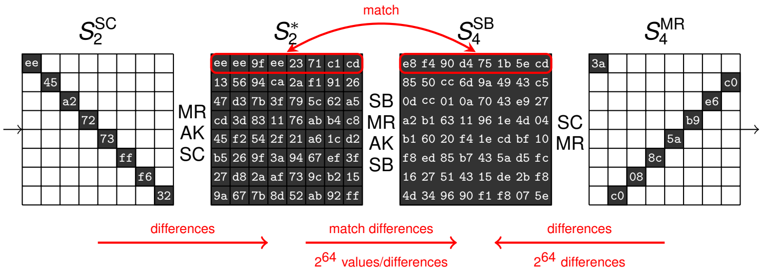


(1) Start with arbitrary differences in state S_2^{SC} and S_4^{MR}

- we propagate all 2^{64} differences backward at once

(2) Match-in-the-middle at SuperBox (SB – MR – AK – SB)

Inbound Phase

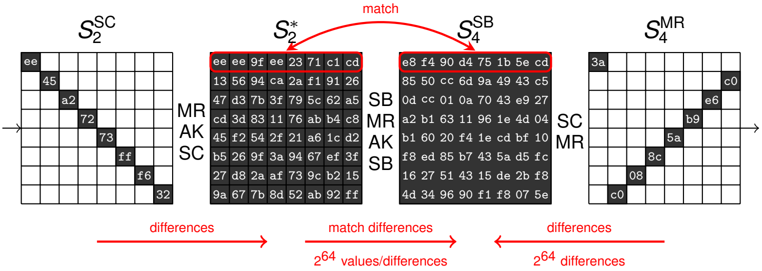


(1) Start with arbitrary differences in state S_2^{SC} and S_4^{MR}

- we propagate all 2^{64} differences backward at once

(2) Match-in-the-middle at SuperBox (SB – MR – AK – SB)

Inbound Phase



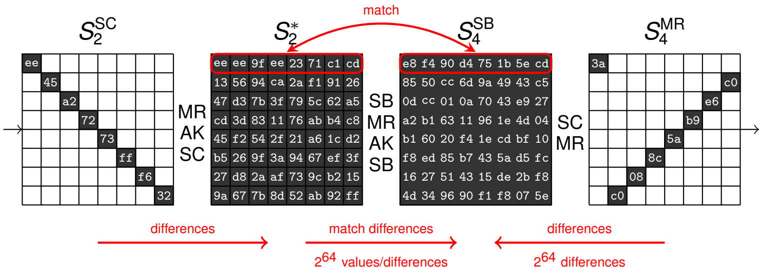
(1) Start with arbitrary differences in state S_2^{SC} and S_4^{MR}

- we propagate all 2^{64} differences backward at once

(2) Match-in-the-middle at SuperBox (SB – MR – AK – SB)

- with complexity 2^{64} we get $\sim 2^{64}$ right pairs

Inbound Phase



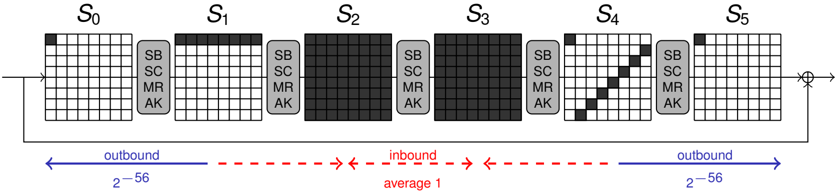
(1) Start with arbitrary differences in state S_2^{SC} and S_4^{MR}

- we propagate all 2^{64} differences backward at once

(2) Match-in-the-middle at SuperBox (SB – MR – AK – SB)

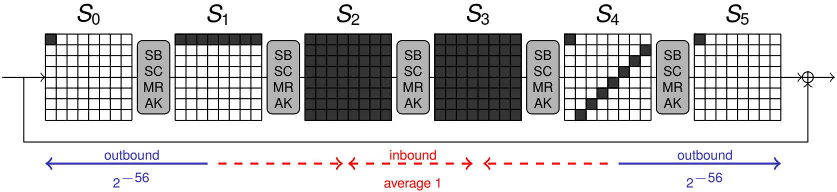
- with complexity 2^{64} we get $\sim 2^{64}$ right pairs
- time-memory trade-off with $T \cdot M = 2^{128}$ with $T \geq 2^{64}$

Extending the Attack to 5 Rounds [LMR+09]



- By adding one round in the inbound phase of the attack we can extend the attack to 5 rounds
- The outbound phase is identical to the attack on 4 rounds
  - probability: 2^{-120}

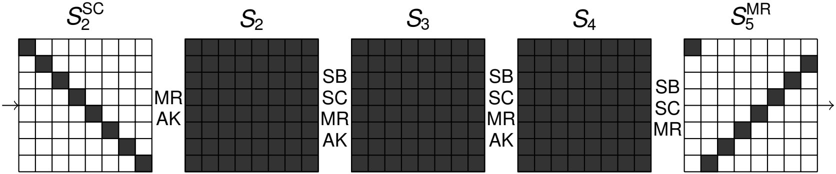
Extending the Attack to 5 Rounds [LMR+09]



- By adding one round in the inbound phase of the attack we can extend the attack to 5 rounds
- The outbound phase is identical to the attack on 4 rounds
  - probability: 2^{-120}

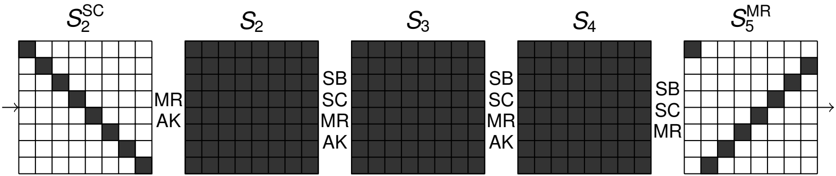
$\Rightarrow$ Construct 2^{120} starting points in the inbound phase with average complexity 1 (but increased memory of 2^{64})

Extending the Attack to 6 Rounds?



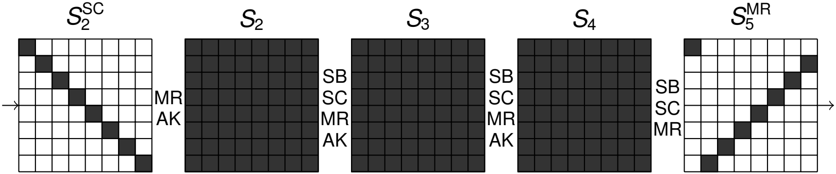
- Add one more round in the inbound phase [JNPP12]

Extending the Attack to 6 Rounds?



- Add one more round in the inbound phase [JNPP12]
    - Complexity $\Rightarrow 2^{256}$ for 1 solution

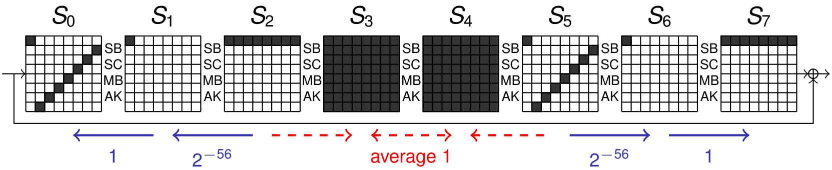
Extending the Attack to 6 Rounds?



- Add one more round in the inbound phase [JNPP12]
  - Complexity $\Rightarrow 2^{256}$ for 1 solution

$\Rightarrow$ Complexity is too high for a collision attack

From Collisions to Near-Collisions

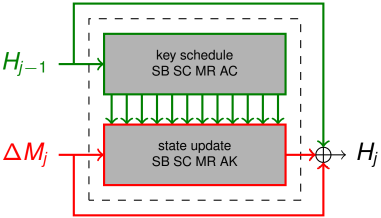


- Add one round at input and output
  - no additional complexity
  - MixRows: 1 $\rightarrow$ 8 with probability 1

$\Rightarrow$ Near-collision attack for 7 rounds

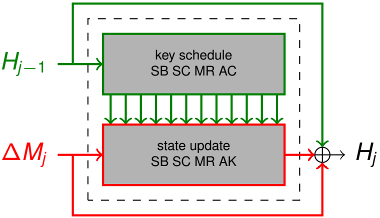
  - time complexity 2^{112} and 2^{64} memory

Compression Function Attacks



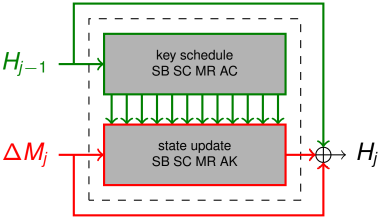
- We can freely choose the chaining input H_{j-1}
  - no differences in H_{j-1}
  - semi-free-start (near-) collisions

Compression Function Attacks



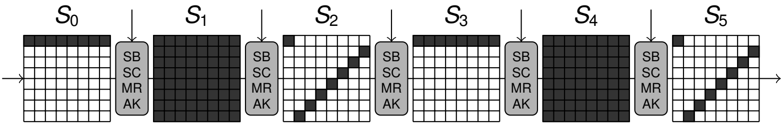
- We can freely choose the chaining input H_{j-1}
  - no differences in H_{j-1}
  - semi-free-start (near-) collisions
- Extend previous attacks by 2 rounds
  - using multiple inbound phases

Compression Function Attacks



- We can freely choose the chaining input H_{j-1}
  - no differences in H_{j-1}
  - semi-free-start (near-) collisions
- Extend previous attacks by 2 rounds
  - using multiple inbound phases
- Outbound phases of attacks stay the same

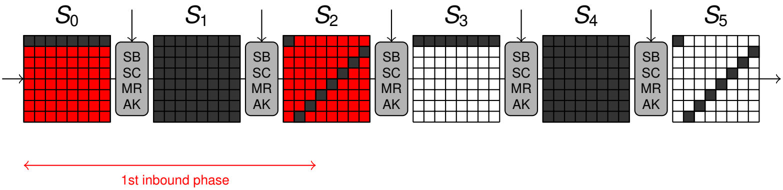
Inbound Phase



Basic Idea:

- use two independent inbound phases

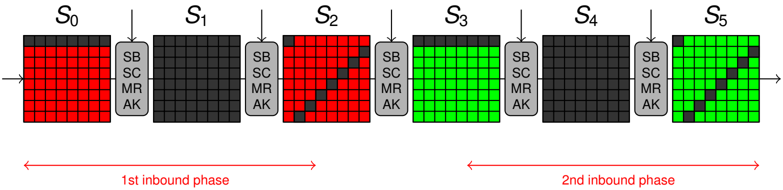
Inbound Phase



Basic Idea:

- use two independent inbound phases

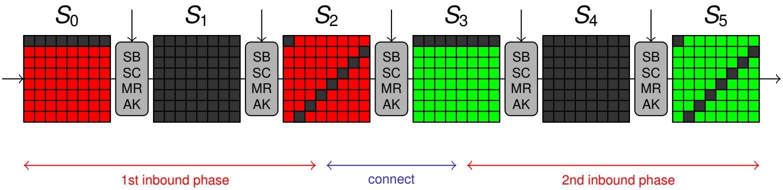
Inbound Phase



Basic Idea:

- use two independent inbound phases

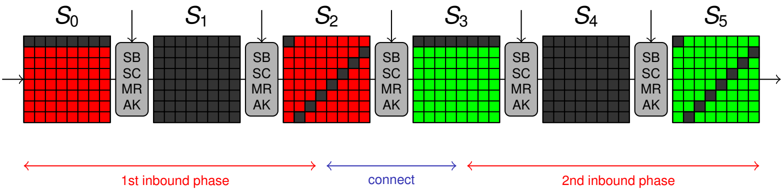
Inbound Phase



Basic Idea:

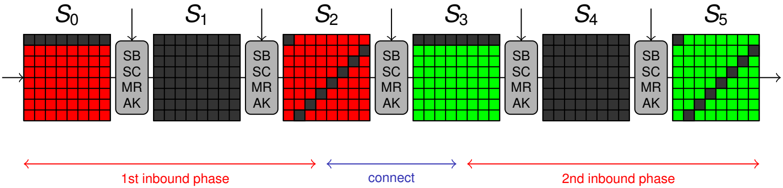
- use two independent inbound phases
- connect them using 512-bit freedom of key input
  - ($S_3 = S_2^{MR} \oplus K_3$)

Inbound Phase



In practice: Slightly more tricky than that (3 key inputs involved)

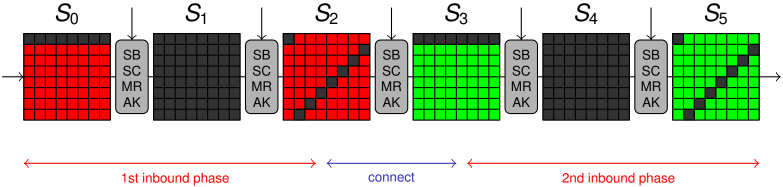
Inbound Phase



In practice: Slightly more tricky than that (3 key inputs involved)

- connect rows independently

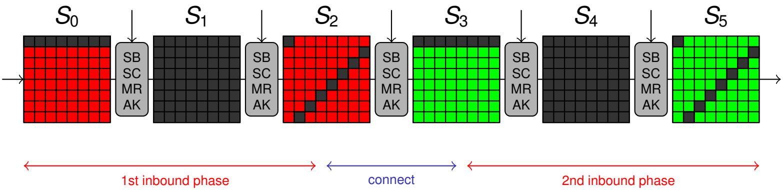
Inbound Phase



In practice: Slightly more tricky than that (3 key inputs involved)

- connect rows independently
- find 2^{64} solutions with complexity 2^{128}

Inbound Phase



In practice: Slightly more tricky than that (3 key inputs involved)

- connect rows independently
- find 2^{64} solutions with complexity 2^{128}

$\Rightarrow$ Collision on 7 and near-collision on 9 rounds

Summary of Results on Whirlpool

target	rounds	computational complexity	memory requirements	type
hash function	5	2^{184-s}	2^s	collision
	7	2^{176-s}	2^s	near-collision
compression function	7	2^{184}	2^{64}	collision
	9	2^{176}	2^{64}	near-collision

Summary of Results on Whirlpool

target	rounds	computational complexity	memory requirements	type
hash function	5.5	2^{184-s}	2^s	collision
	7.5	2^{176-s}	2^s	near-collision
compression function	7.5	2^{184}	2^{64}	collision
	9.5	2^{176}	2^{64}	near-collision

Summary of Results on Whirlpool

target	rounds	computational complexity	memory requirements	type
hash function	5.5	2^{184-s}	2^s	collision
	7.5	2^{176-s}	2^s	near-collision
compression function	7.5	2^{184}	2^{64}	collision
	9.5	2^{176}	2^{64}	near-collision
	10	2^{188}	2^{64}	distinguisher

Summary

- Basic principle not that difficult
  - efficient inbound phase (average 1)
  - probabilistic outbound phase (determined by linear layer)
- Difficulty in constructing and merging inbound phases
  - finding good and sparse truncated differential paths
  - efficient way to use available freedom for merge

$\Rightarrow$ powerful tool in the cryptanalysis of hash functions

Thank you for your attention!

http://eprint.iacr.org/2010/198

References I

Paulo S. L. M. Barreto and Vincent Rijmen.
The WHIRLPOOL Hashing Function.
Submitted to NESSIE, September 2000, revised May 2003, 2000.
Available online: `http://www.larc.usp.br/~pbarreto/WhirlpoolPage.html`.

Henri Gilbert and Thomas Peyrin.
Super-Sbox Cryptanalysis: Improved Attacks for AES-Like Permutations.
In Seokhie Hong and Tetsu Iwata, editors, *FSE*, volume 6147 of *LNCS*, pages 365–383. Springer, 2010.

Jérémy Jean, María Naya-Plasencia, and Thomas Peyrin.
Improved Rebound Attack on the Finalist Grøstl.
In Anne Canteaut, editor, *FSE*, volume 7549 of *LNCS*, pages 110–126. Springer, 2012.

Jérémy Jean, María Naya-Plasencia, and Martin Schläffer.
Improved Analysis of ECHO-256.
In Ali Miri and Serge Vaudenay, editors, *Selected Areas in Cryptography*, volume 7118 of *LNCS*, pages 19–36. Springer, 2011.

Stefan Kölbl and Florian Mendel.
Practical Attacks on the Maelstrom-0 Compression Function.
In Javier Lopez and Gene Tsudik, editors, *ACNS*, volume 6715 of *LNCS*, pages 449–461, 2011.

Dmitry Khovratovich, María Naya-Plasencia, Andrea Röck, and Martin Schläffer.
Cryptanalysis of *Luffa* v2 Components.
In Alex Biryukov, Guang Gong, and Douglas R. Stinson, editors, *Selected Areas in Cryptography*, volume 6544 of *LNCS*, pages 388–409. Springer, 2010.

References II

Mario Lamberger, Florian Mendel, Christian Rechberger, Vincent Rijmen, and Martin Schläffer.
Rebound Distinguishers: Results on the Full Whirlpool Compression Function.
In Mitsuru Matsui, editor, *ASIACRYPT*, volume 5912 of *LNCS*, pages 126–143. Springer, 2009.

Mario Lamberger, Florian Mendel, Christian Rechberger, Vincent Rijmen, and Martin Schläffer.
The Rebound Attack and Subspace Distinguishers: Application to Whirlpool.
Cryptology ePrint Archive, Report 2010/198, 2010.

Krystian Matusiewicz, María Naya-Plasencia, Ivica Nikolić, Yu Sasaki, and Martin Schläffer.
Rebound Attack on the Full Lane Compression Function.
In Mitsuru Matsui, editor, *ASIACRYPT*, volume 5912 of *LNCS*, pages 106–125. Springer, 2009.

Marine Minier, María Naya-Plasencia, and Thomas Peyrin.
Analysis of Reduced-SHAvite-3-256 v2.
In Antoine Joux, editor, *FSE*, volume 6733 of *LNCS*, pages 68–87. Springer, 2011.

Florian Mendel, Thomas Peyrin, Christian Rechberger, and Martin Schläffer.
Improved Cryptanalysis of the Reduced Grøstl Compression Function, ECHO Permutation and AES Block Cipher.
In Michael J. Jacobson Jr., Vincent Rijmen, and Reihaneh Safavi-Naini, editors, *Selected Areas in Cryptography*, volume 5867 of *LNCS*, pages 16–35. Springer, 2009.

Florian Mendel, Christian Rechberger, and Martin Schläffer.
Cryptanalysis of Twister.
In Michel Abdalla, David Pointcheval, Pierre-Alain Fouque, and Damien Vergnaud, editors, *ACNS*, volume 5536 of *LNCS*, pages 342–353, 2009.

References III

- Florian Mendel, Christian Rechberger, Martin Schläffer, and Søren S. Thomsen.
  The Rebound Attack: Cryptanalysis of Reduced Whirlpool and Grøstl.
  In Orr Dunkelman, editor, *FSE*, volume 5665 of *LNCS*, pages 260–276. Springer, 2009.
- Florian Mendel, Christian Rechberger, Martin Schläffer, and Søren S. Thomsen.
  Rebound Attacks on the Reduced Grøstl Hash Function.
  In Josef Pieprzyk, editor, *CT-RSA*, volume 5985 of *LNCS*, pages 350–365. Springer, 2010.
- María Naya-Plasencia.
  How to Improve Rebound Attacks.
  In Phillip Rogaway, editor, *CRYPTO*, volume 6841 of *LNCS*, pages 188–205. Springer, 2011.
- María Naya-Plasencia, Deniz Toz, and Kerem Varici.
  Rebound Attack on JH42.
  In Dong Hoon Lee and Xiaoyun Wang, editors, *ASIACRYPT*, volume 7073 of *LNCS*, pages 252–269. Springer, 2011.
- Thomas Peyrin.
  Improved Differential Attacks for ECHO and Grøstl.
  In Tal Rabin, editor, *CRYPTO*, volume 6223 of *LNCS*, pages 370–392. Springer, 2010.
- Yu Sasaki, Yang Li, Lei Wang, Kazuo Sakiyama, and Kazuo Ohta.
  Non-full-active Super-Sbox Analysis: Applications to ECHO and Grøstl.
  In Masayuki Abe, editor, *ASIACRYPT*, volume 6477 of *LNCS*, pages 38–55. Springer, 2010.
- Yu Sasaki, Lei Wang, Shuang Wu, and Wenling Wu.
  Investigating Fundamental Security Requirements on Whirlpool: Improved Preimage and Collision Attacks.
  In Xiaoyun Wang and Kazue Sako, editors, *ASIACRYPT*, volume 7658 of *LNCS*, pages 562–579. Springer, 2012.