Improved Analysis of ECHO-256

Jérémy Jean[1] María Naya-Plasencia[2] Martin Schläffer[3]

[1]École Normale Supérieure, France

[2]FHNW, Windisch, Switzerland

[3]IAIK, Graz University of Technology, Austria

SAC'2011 – August 11, 2011

Outline of the talk

Outline

- Previous cryptanalysis
- Description of ECHO-256
- Collision attack on the 5-round hash function
- Distinguisher on the 7-round compression function
- Conclusion

Previous cryptanalysis of ECHO-256

Hash function

Rounds	Time	Memory	Type	Reference
4/8	2^{64}	2^{64}	**collision**	**new** (Extended Version)
5/8	2^{112}	$2^{85.3}$	**collision**	**new**

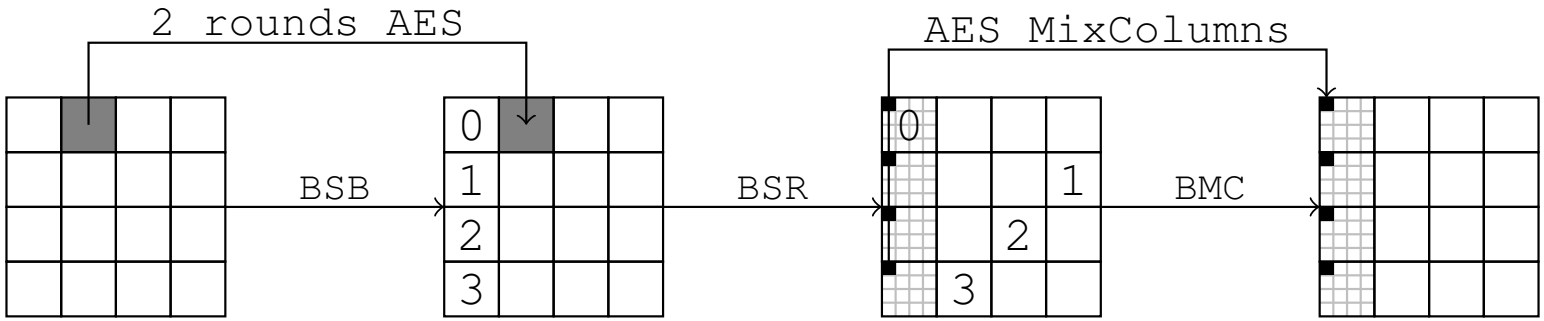
Compression function

Rounds	Time	Memory	Type	Reference
3/8	2^{64}	2^{32}	free-start collision	[Peyrin-C10]
3/8	2^{96}	2^{32}	semi-free-start collision	[Peyrin-C10]
4/8	2^{96}	2^{32}	distinguisher	[Peyrin-C10]
4/8	2^{36}	2^{16}	distinguisher	[JeanFouque-FSE11]
4/8	2^{52}	2^{16}	semi-free-start collision	[JeanFouque-FSE11]
6/8	$\mathbf{2^{160}}$	$\mathbf{2^{128}}$	**collision, chosen salt**	**new** (Extended Version)
6/8	$\mathbf{2^{193}}$	$\mathbf{2^{128}}$	**collision**	**new**
7/8	$\mathbf{2^{160}}$	$\mathbf{2^{128}}$	**distinguisher, chosen salt**	**new** (Extended Version)
7/8	$\mathbf{2^{193}}$	$\mathbf{2^{128}}$	**distinguisher**	**new**

Description of the hash function

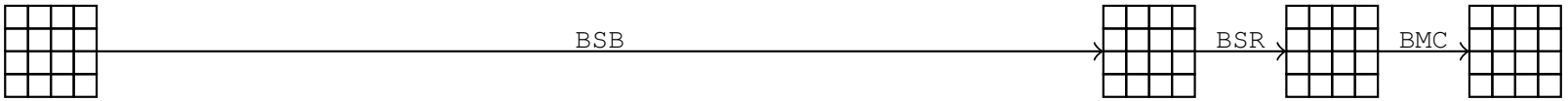
ECHO-256

- Submitted to SHA-3 by Gilbert et al.
- Merkle-Damgård construction
- HAIFA design (counter & salt)
- 2048-bit internal state as a 4×4 matrix of AES states
- 8-round AES-based permutation : BSB, BSR, BMC
- Output transformation : compress and truncate



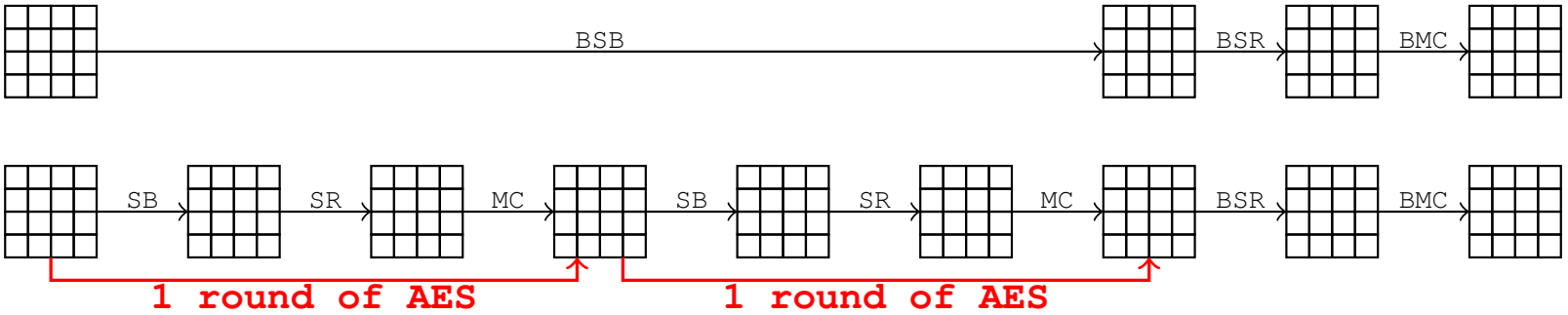
Alternative view

- Breaking down to the AES-state level of operations
  - **SuperSBox** = **SB – MC – SB** [LMRRS-A09, GP-FSE10]
  - **SuperMixColumns** = **MC – BMC** [Schläffer-SAC10]



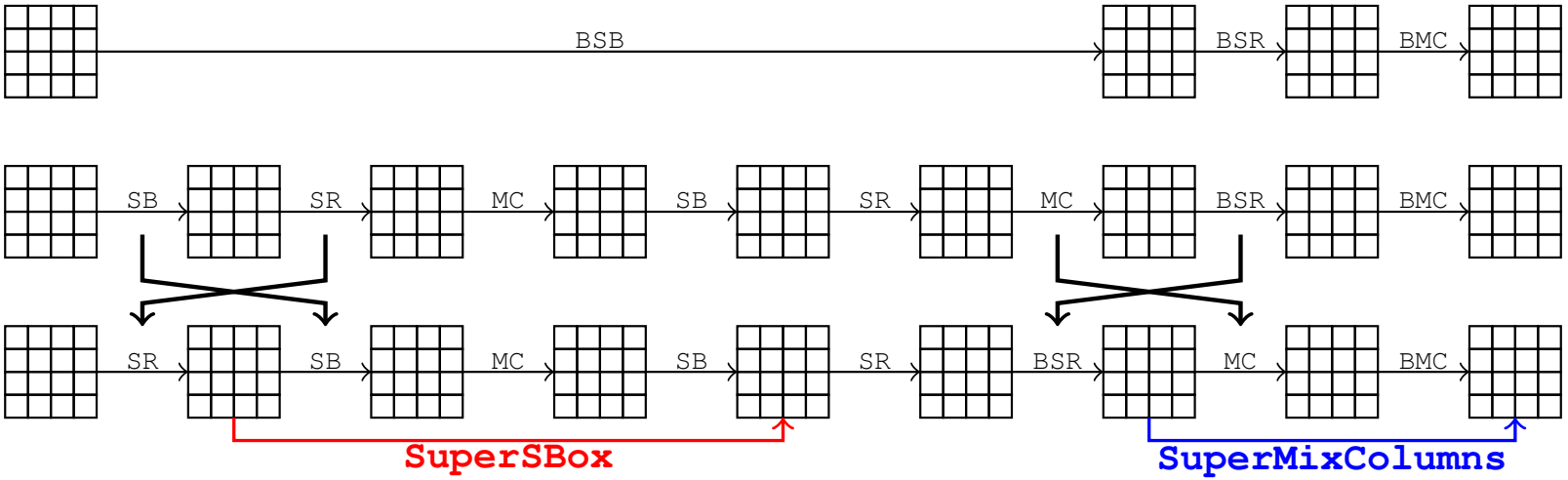
Alternative view

- Breaking down to the AES-state level of operations
  - **SuperSBox** = **SB – MC – SB** [LMRRS-A09, GP-FSE10]
  - **SuperMixColumns** = **MC – BMC** [Schläffer-SAC10]



Alternative view

- Breaking down to the AES-state level of operations
  - **SuperSBox** = **SB – MC – SB** [LMRRS-A09, GP-FSE10]
  - **SuperMixColumns** = **MC – BMC** [Schläffer-SAC10]



Super Transformations

SuperSBox

- Introduced by Rijmen and Daemen in 2006
- Used in [LMRRS-A09, GP-FSE10]
- **SuperSBox = SB – MC – SB**
- Works on 32-bit AES-columns
- $\mathbb{P}(\Delta_{IN} \rightarrow \Delta_{OUT} \text{ exists}) \approx 1/2$

Super Transformations

SuperSBox

- Introduced by Rijmen and Daemen in 2006
- Used in [LMRRS-A09, GP-FSE10]
- **SuperSBox = SB – MC – SB**
- Works on 32-bit AES-columns
- $\mathbb{P}(\Delta_{IN} \rightarrow \Delta_{OUT} \text{ exists}) \approx 1/2$

SuperMixColumns

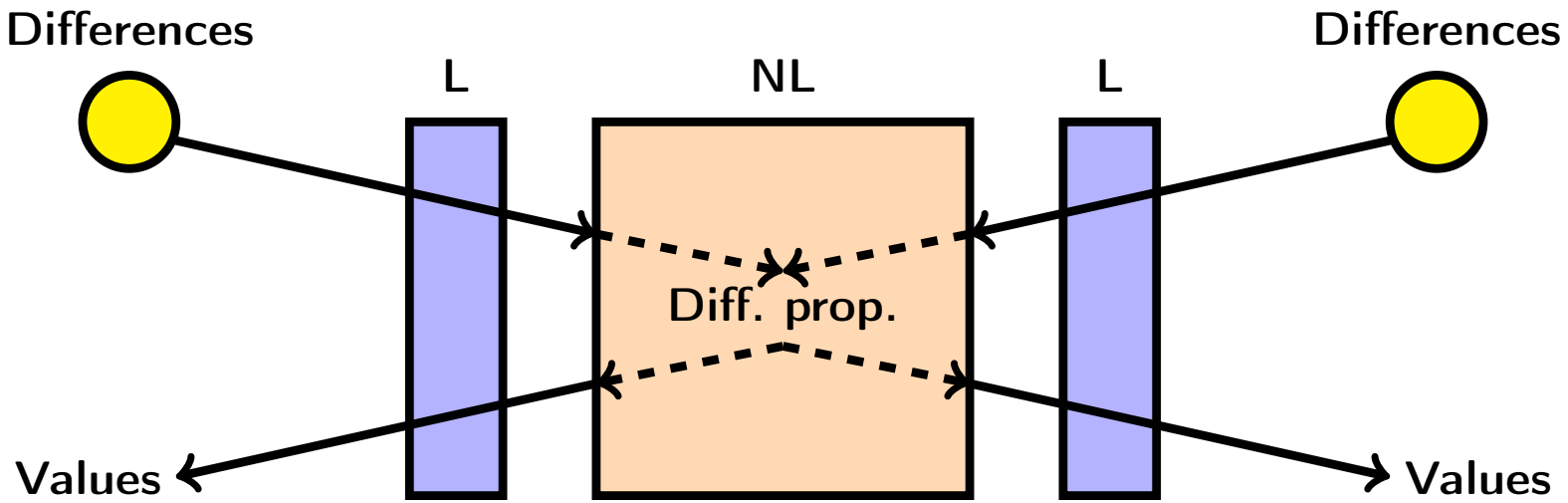
- Super transformation introduced in [Schläffer-SAC10]
- **SuperMixColumns = MC – BigMC**
- Works on 16×1 byte-slices
- $\mathbf{M}_{SMC} = \mathbf{M} \otimes \mathbf{M}$ ($\mathbf{M}$ from MixColumns)

Rebound technique [MRST-FSE09]

For a given truncated differential path

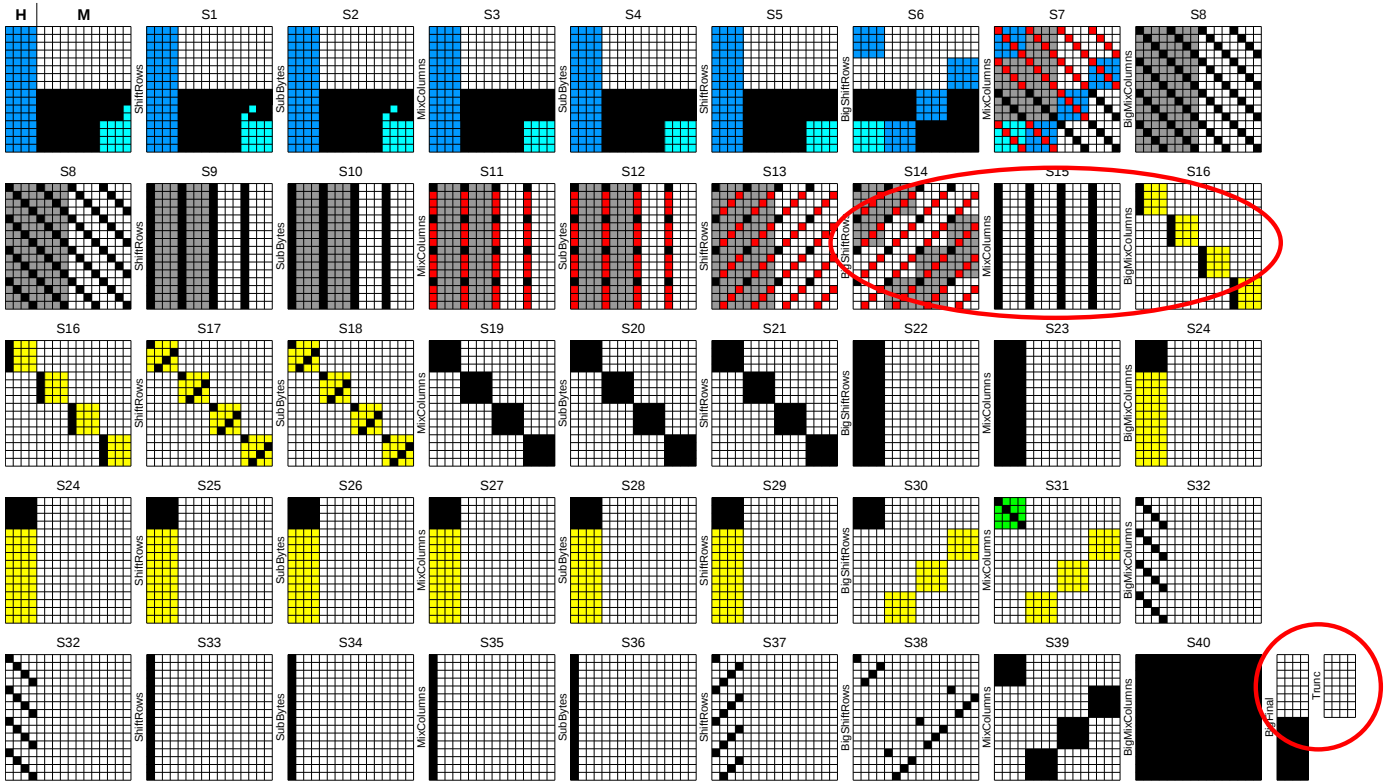
Set differences and values around a non-linear layer using its differential properties with amortized complexity one

NL = AES SBox or SuperSBox



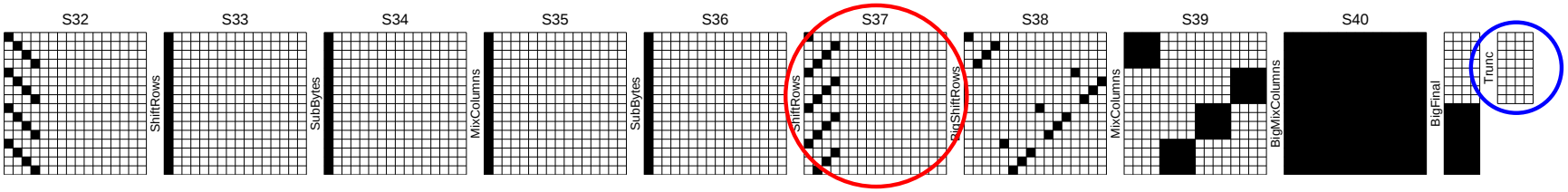
5-round Hash Function Collision Attack

Truncated Differential Path



- Almost the same path as in [Schläffer-SAC10]
- Improved attack to get collisions instead of distinguisher
- Corrected attack to find solutions also in the hash function

How to get Collisions



For some differences a, b, c, d of the first column slice (16x1) of state S37, we get a collision in the first column slice at the output (8 bytes) if

$$\mathbf{M_{trunc}} \cdot \mathbf{M_{SMC}} \cdot \begin{bmatrix} a\,0\,0\,0\,b\,0\,0\,0\,c\,0\,0\,0\,d\,0\,0\,0 \end{bmatrix}^{\mathrm{T}} =$$

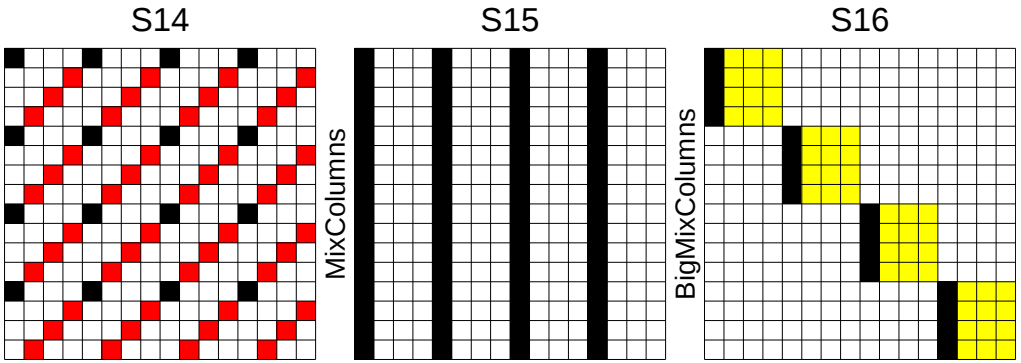
$$\underbrace{\begin{bmatrix} 4 & 6 & 2 & 2 & 6 & 5 & 3 & 3 \\ 2 & 3 & 1 & 1 & 4 & 6 & 2 & 2 \\ 2 & 3 & 1 & 1 & 2 & 3 & 1 & 1 \\ 6 & 5 & 3 & 3 & 2 & 3 & 1 & 1 \end{bmatrix}^{\mathrm{T}}}_{\mathbf{M_{comb}}} \cdot \begin{bmatrix} a\,b\,c\,d \end{bmatrix}^{\mathrm{T}} = \begin{bmatrix} 0\,0\,0\,0\,0\,0\,0\,0 \end{bmatrix}^{\mathrm{T}}$$

$\mathrm{rank}(M_{comb}) = 2 \implies \mathbb{P}(\text{one slice}) = 2^{-16}$.
So : $\mathbb{P}(\text{collision}) = 2^{-16\times 4} = 2^{-64}$.

Problem of the Previous Attack

SuperMixColumns Problem [JeanFouque-FSE11]

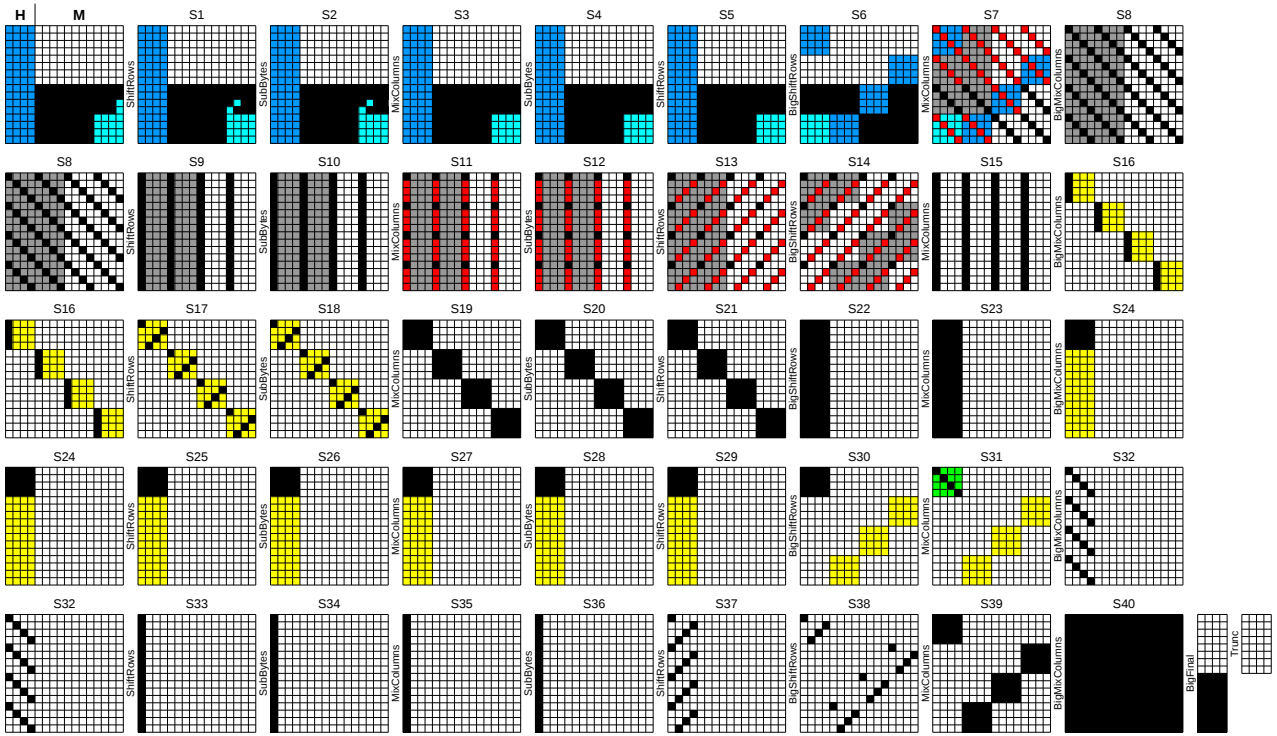
For given differences in all bytes ■ and given values in bytes ■ ■ ■ of state S14 and S16, a solution exists only with probability 2^{-128}.



Solution

- Solved for compression function attacks [JeanFouque-FSE11]
- More difficult for the hash function (larger constraints)

Outline of the Attack



1. 1st inbound
2. 1st outbound
3. 2nd inbound
4. 1st merge inbound $\Leftrightarrow$
5. merge chaining $\Leftrightarrow$
6. 2nd merge inbound
7. 3nd merge inbound
8. 2nd outbound to get collision

Improvements Compared to Previous Attacks

1. 1st inbound ■
2. 1st outbound ■
3. 2nd inbound ■
4. 1st merge inbound ■ ⇔ ■
5. merge chaining ■ ■ ⇔ ■
6. 2nd merge inbound ■
7. 3nd merge inbound □
8. 2nd outbound to get collision

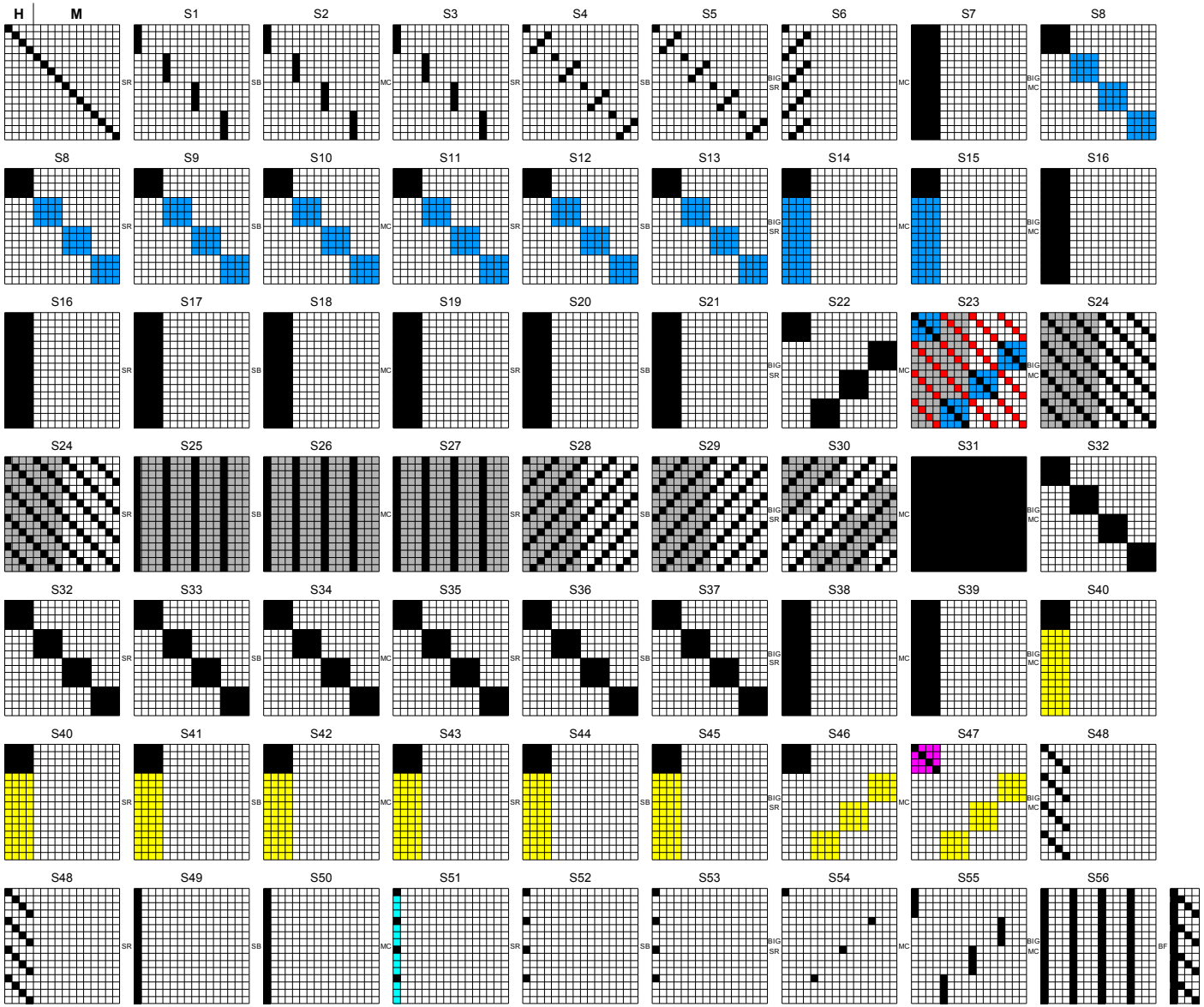
- inbound/outbound phases are largely the same as in previous attacks on ECHO
- new : separate merging phase into 3 parts :
  - solve first 128-bit condition using birthday effect and by generating enough solutions for the 2nd inbound (■)
  - solve second 128-bit condition by choosing gray values (■)
  - solve final 192-bit condition by choosing white values (□)
- drawback : all phases have time/memory complexities above 2^{64}

Complexity Analysis

1. 1st inbound ■
   - time 2^{96}, memory 2^{64} to get 2^{96} solutions
2. 1st outbound ■
   - time 2^{96}, memory 2^{64} to get 1 solution
3. 2nd inbound ■
   - time 2^{64}, memory 2^{64} to get $2^{32} \times 2^{32} \times 2^{32} \times 2^{64} = 2^{160}$ solutions
4. 1st merge inbound ■ $\Leftrightarrow$ ■
   - time 2^{96}, memory 2^{64} to get 2^{32} solutions
5. merge chaining ■ ■ $\Leftrightarrow$ ■
   - time 2^{112}, memory 2^{48} to get 1 solution
6. 2nd merge inbound ■
   - time 2^{64}, memory 2^{64} to get 1 solution
7. 3nd merge inbound □
   - time $2^{85.3}$, memory $2^{85.3}$ to get 2^{64} solutions
8. 2nd outbound to get collision
   - time 2^{64}, memory 1 to get 1 collision

7-round Compression Function Attack

Truncated Differential Path



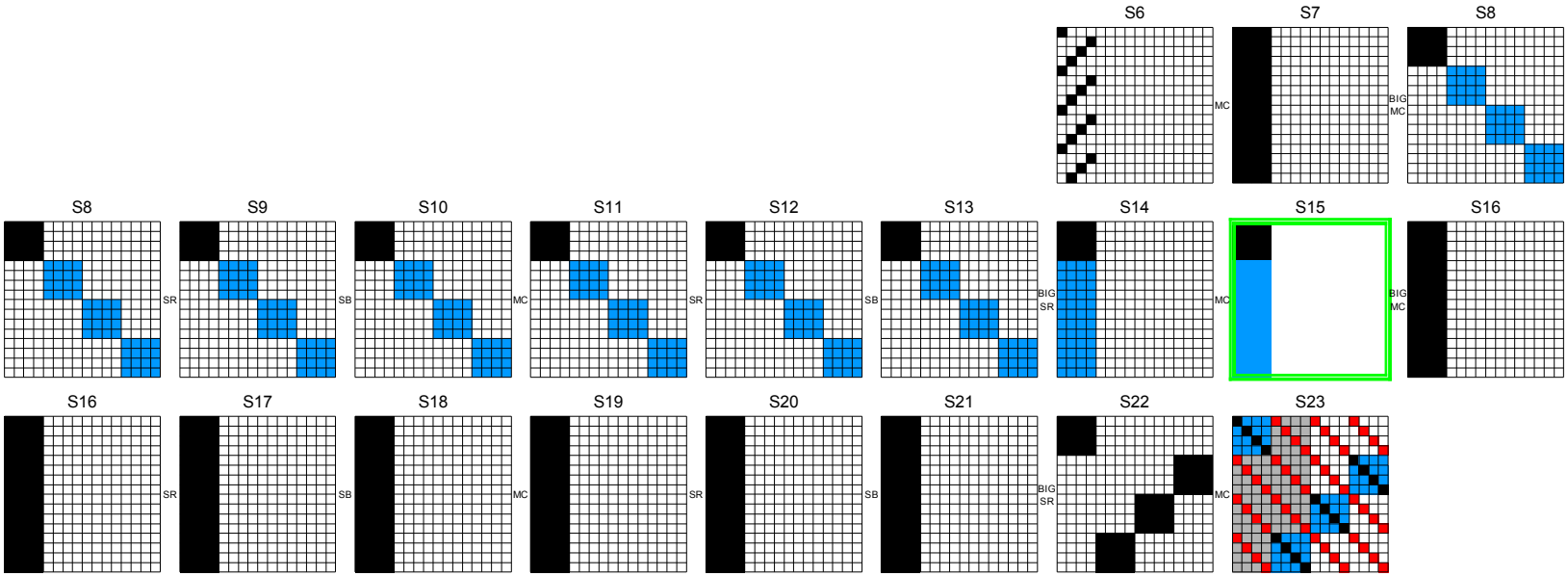
Finding solutions for the path

Finding a right pair

1. Solutions for S6 to S23 (stop-in-the-middle [NayaPlasencia-C11])
2. Solutions for S30 to S48 (idem)
3. Merge both partial solutions
4. Find the remaining values with the same method as before

Differential solutions for S6 to S23

First, we consider the *first half*.



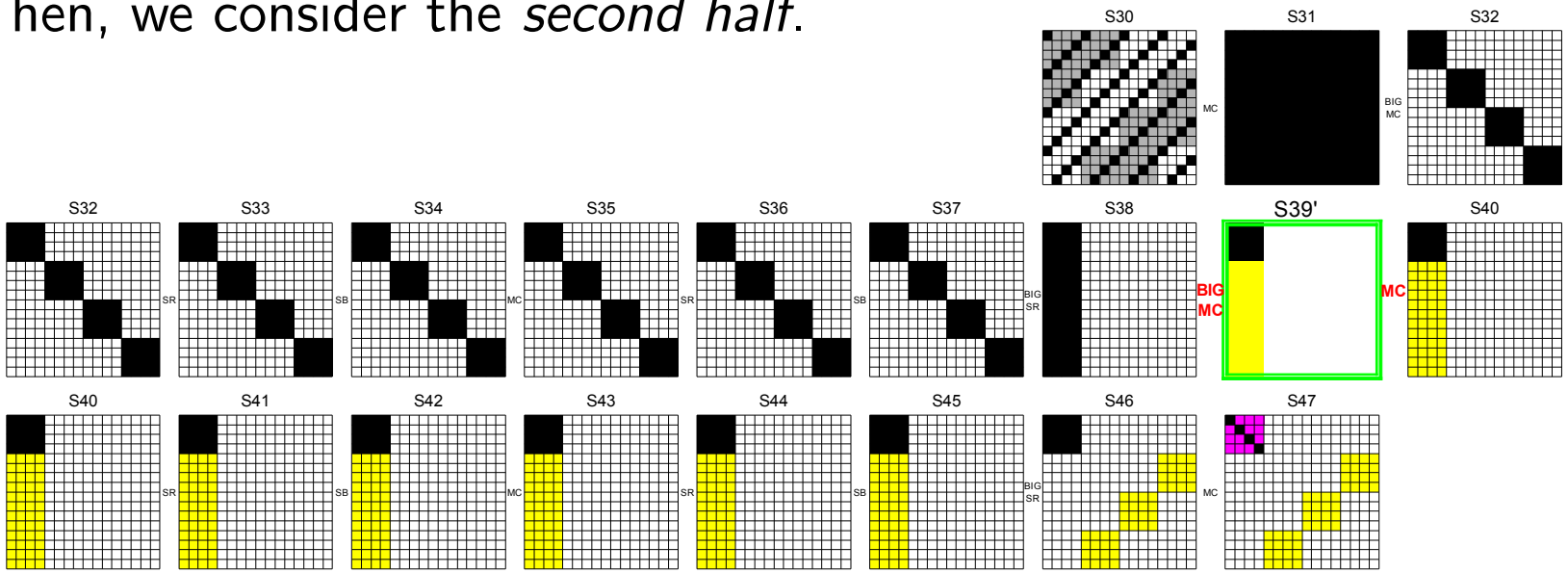
Compute partial values and differences

- Stop-in-the-middle algorithm where S15 is the *middle*
- 2^{64} solutions for blue and black bytes

  $\Longrightarrow$ 2^{129} in time and 2^{64} in memory

Differential solutions for S30 to S47

Then, we consider the *second half*.

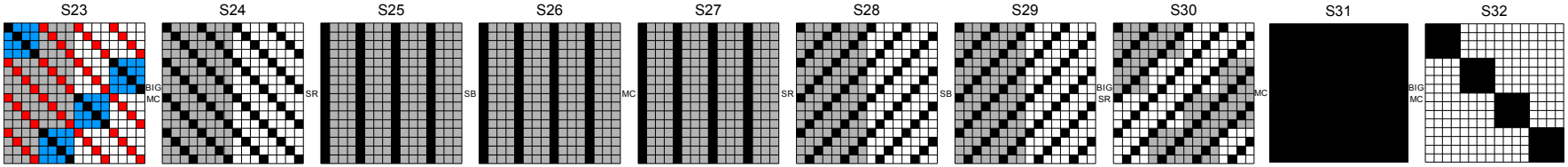


Compute partial values and differences

- MixColumns and BigMixColumns commute
- Stop-in-the-middle algorithm, where S39 is the *middle*
- 2^{64} solutions for yellow and black bytes

  $\Longrightarrow$ 2^{129} in time and 2^{64} in memory

Merging solutions from S23 to S30



Merge

- Blue/black part fixed from the first half (S6 to S23)
- Yellow/black part fixed from the second half (S32 to S47)
- Find values and differences that match
- Consider the SuperSBoxes separately
- Match step-by-step in 2^{193} time and 2^{128} memory

Solutions for the whole path

Solutions

- Using the method from [Schläffer-SAC10], we find solutions completing the part of the path without differences
- 128-bit condition from [JeanFouque-FSE11] verified
- In the generic case, finding such a pair of input/output costs 2^{240} in time
- Ours : 2^{193} in time and 2^{128} in memory
- Can also produce compression function collisions on 6 rounds with the same complexity

Conclusion I

Attack Property

- Attack split in small parts/phases
- Each part has complexity below generic scenario
- Parts are merged with complexity below the generic one
- We may even split parts into sub-parts

Conclusion II

- Results on the 5-round Hash Function
  $\Longrightarrow$ Collision in time 2^{112} and memory $2^{85.3}$
- Results on the 6- and 7-round Compression Function
  $\Longrightarrow$ 6R Collision in time 2^{193} and memory 2^{128}
  $\Longrightarrow$ 7R Distinguisher in time 2^{193} and memory 2^{128}
- Extended version on ePrint : **ePrint/2011/422**
  $\Longrightarrow$ 4R hash function collision attack in time 2^{64} and memory 2^{64}
  $\Longrightarrow$ 6R compression function collision attack in the chosen-salt model in time 2^{160} and memory 2^{128}
  $\Longrightarrow$ 7R compression function distinguisher in the chosen-salt model in time 2^{160} and memory 2^{128}

Conclusion II

- Results on the 5-round Hash Function
  $\Longrightarrow$ Collision in time 2^{112} and memory $2^{85.3}$
- Results on the 6- and 7-round Compression Function
  $\Longrightarrow$ 6R Collision in time 2^{193} and memory 2^{128}
  $\Longrightarrow$ 7R Distinguisher in time 2^{193} and memory 2^{128}
- Extended version on ePrint : **ePrint/2011/422**
  $\Longrightarrow$ 4R hash function collision attack in time 2^{64} and memory 2^{64}
  $\Longrightarrow$ 6R compression function collision attack in the chosen-salt model in time 2^{160} and memory 2^{128}
  $\Longrightarrow$ 7R compression function distinguisher in the chosen-salt model in time 2^{160} and memory 2^{128}

Thank you !