New Non-Ideal Properties of AES-Based Permutations: Applications to ECHO and Grøstl

Yu Sasaki[1], Yang Li[2], Lei Wang[2],
Kazuo Sakiyama[2], Kazuo Ohta[2]

1: NTT Corporation
2: The University of Electro-Communications

NTT

Summary

- New analysis on AES-based design.
  - Targets in 2nd round candidates: ECHO, Grøstl
- Find Non-ideal properties with a low complexity.

Target	#rounds	Attack	Previous (Time, Mem.)	Ours (Time, Mem.)
ECHO-256 (permutation)	8 full	distinguish	$(2^{512},2^{512})$	$(2^{182},2^{37})$
Grøstl-256 (permutation)	8	distinguish	$(2^{112},2^{64})$	$(2^{48},2^{8})$
Grøstl-512 (comp. func.)	7	semi-free-start collision	$(2^{152},2^{64})$	$(2^{152},2^{56})$

- No impact on the security of “hash function.”

NTT

Contents

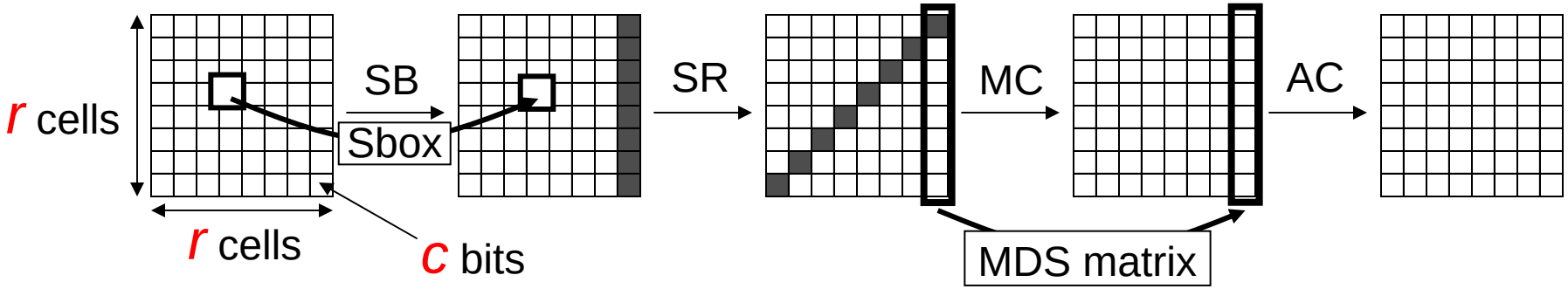
- Previous technique
  - Rebound attack
  - Super-Sbox analysis on rebound attack
- New approach for AES-based permutations
  - Idea of non-full-active Super-Sbox analysis
- Application Results
  - full-round ECHO-256 permutation
  - Grøstl-512 compression function

NTT

AES-Based Permutation

- Use AES-round as a component of a permutation. Naturally motivated design:

  security / implementation / simplicity



- 128-bit state of AES ($r = 4$, $c = 8$) is too small to produce 512-bit digests.

  ➡ r is often enlarged.

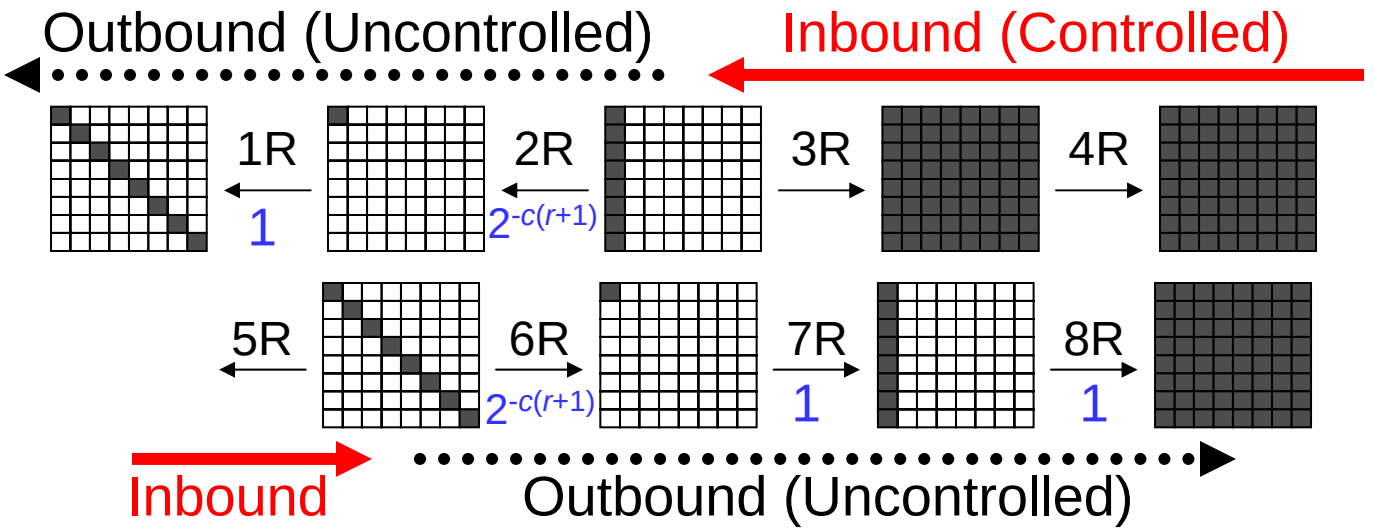
Attack on AES-Based Permutation

- Rebound Attack [MRST09]
  - Analytic tool useful for AES-based design
  - Truncated Diff. Attack optimized for key-less primitive *e.g.* hash function and permutation
- Super-Sbox Analysis [GP10]
  - Differential property of 2 AES-rounds
  - Extends the number of attacked rounds when applied to rebound attack

NTT

Rebound Attack Procedure

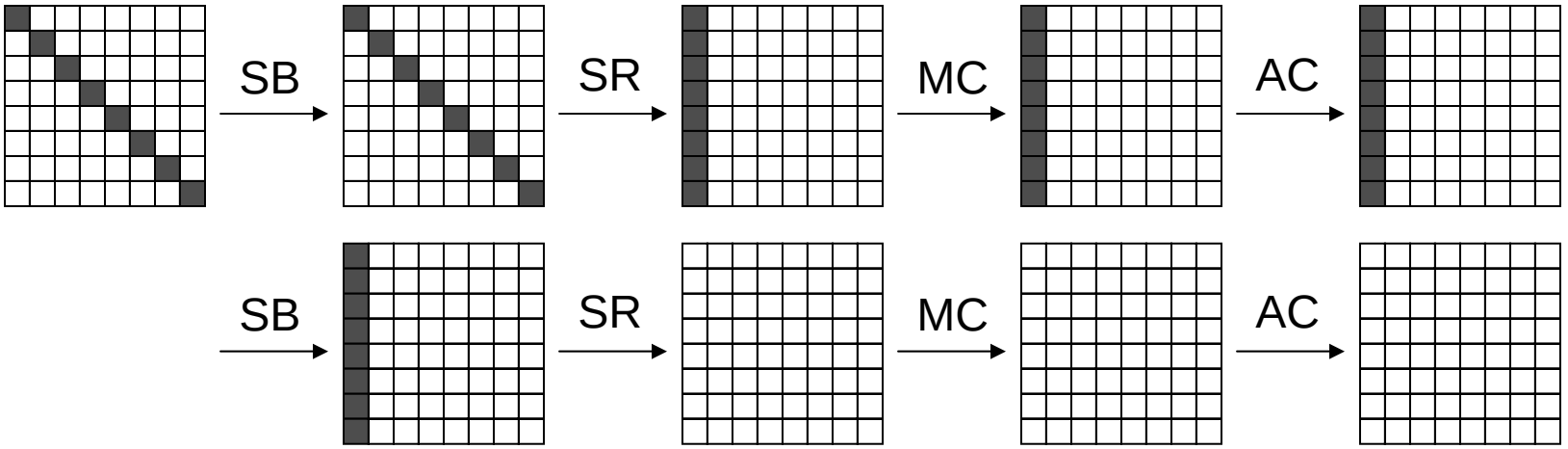
Goal: Efficiently find pairs satisfying the diff. path



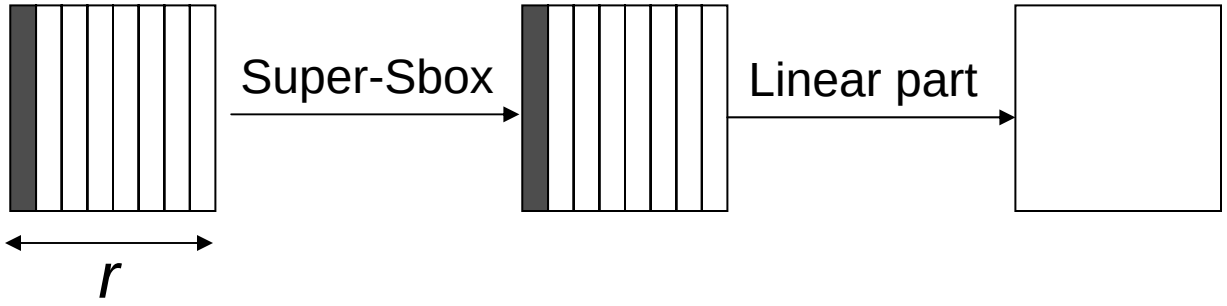
1. Prepare the differential path.
2. Efficiently find pairs satisfying inbound phase.
   Such pairs are called “Starting Points (SPs)”
3. For all SPs, exhaustively try outbound phase.

NTT

Super-Sbox

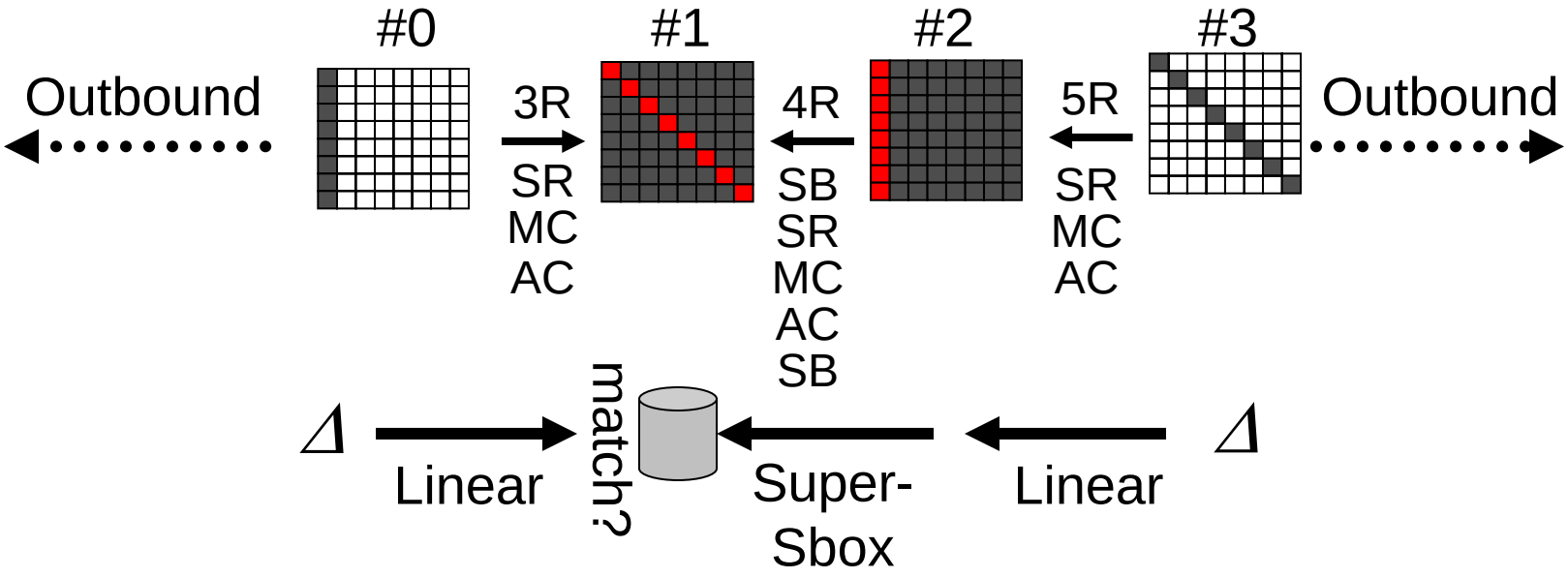


- Non-linear part of 2 AES-rounds can be computed column by column.
- Regard it as an Sbox of 2^{rc} size.



NTT

Super-Sbox in Rebound Attack

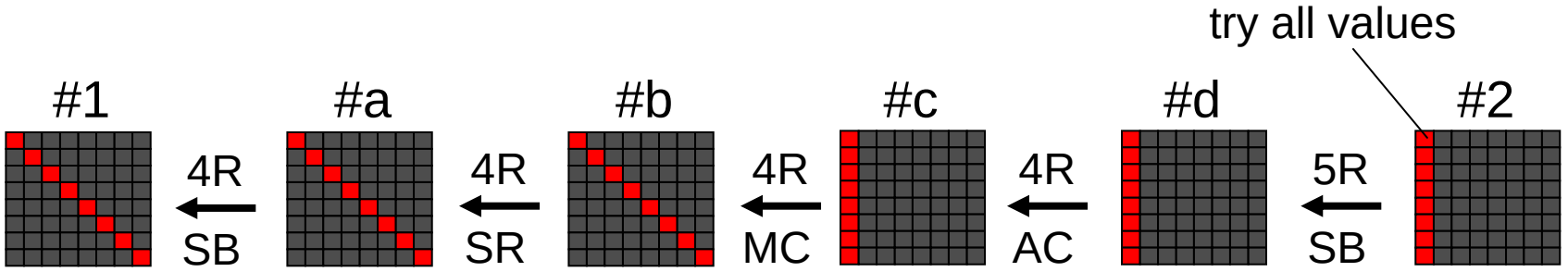


Back
- Fix a diff at #3 and compute diff at #2.
- For all 2^{rc} values of each column at #2, compute Super-Sbox to obtain diff at #1.

For
- Choose a diff at #0 and compute diff at #1.
- Find the matched value in the table.

NTT

Problems of Super-Sbox Analysis



Goal: For each column, from a given difference at #2, compute all possible differences at #1.

- To compute the difference at #1, we need to know values of all (r) bytes in a column.
- Thus, 2^{rc} trials are required, which makes both time and memory at least 2^{rc}.

Our Approach

Non-Full-Active Super-Sbox Analysis

- Avoid activating all bytes.
- Through the S-box, if input diff is 0, output diff is always 0 regardless of the value.

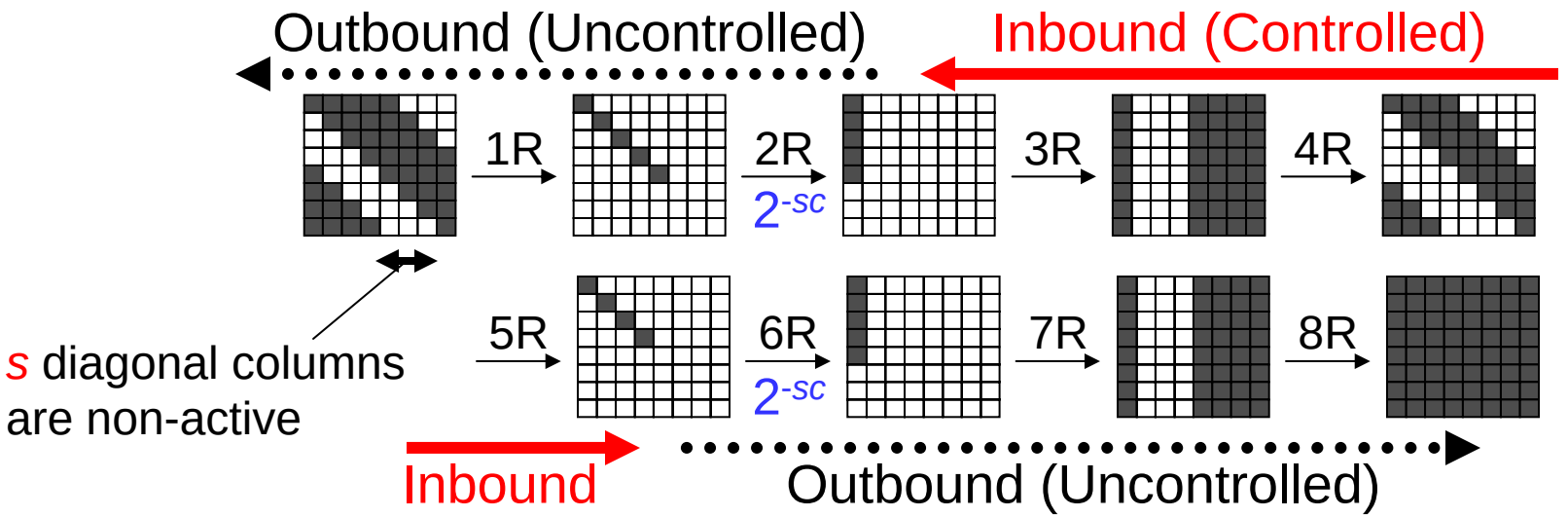
$\Delta=0 \longrightarrow$ S-box $\longrightarrow \Delta=0$

- Freedom of values can be used to adjust other bytes inside the Super-Sbox.

➡ complexity is reduced from 2^{rc}.

NTT

New Differential Path



- All states are non-full-active.
- *s* is a parameter. Different *s* is possible as long as it satisfies a certain condition.
- The path must satisfy MDS property when Mix-Columns is operated.

NTT

Non-Full-Active Super-Sbox Analysis (Summary)

- Inbound phase
  - Generate 2^c starting points with 2^c time and 2^c memory. (1 time for 1 SP, in average)
- Outbound phase
  - Succ. Pr. is $(2^{-sc})^2$. Thus 2^{2sc} SPs are needed. Cond. 1
- Freedom Degrees
  - $2^{c(r+1)}$ SPs can be generated in maximum. Cond. 2
- Complexity for the ideal case.
  - Complexity to find *crs*-bit collision; $2^{crs/2}$. Cond. 3

Conditions 1 and 2 lead to the range of s : $s \leq \frac{r+1}{2}$

Condition 3 leads to the range of r : $r > 4$

Do not work for AES, but usually state is enlarged.

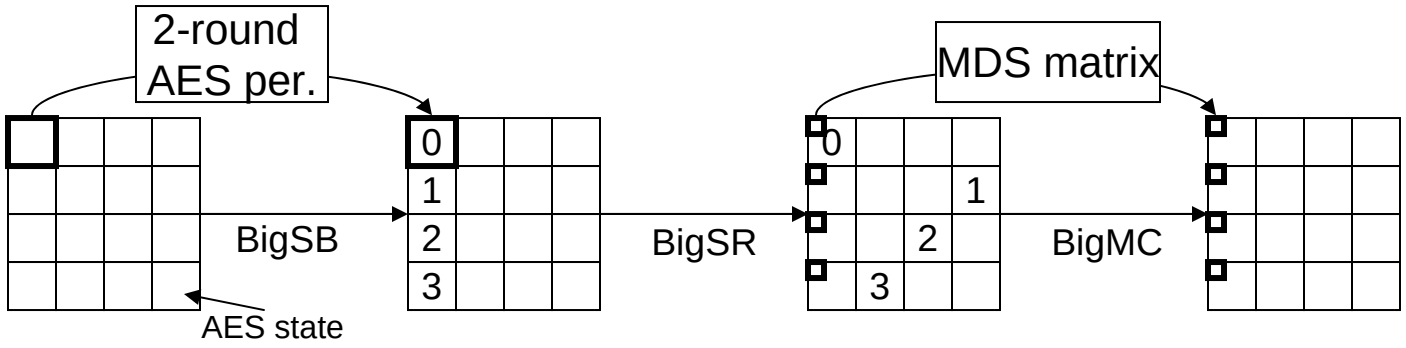
NTT

Applications

Permutations in ECHO and Grøstl

ECHO-256

2048-bit permutation with $r = 4$, $c = 128$, 8-rounds.



Grøstl-256

512-bit permutation with $r = 8$, $c = 8$, 10-rounds.

In the end, both compress the data to 256-bits.

NTT

Application: 8-round Grøstl-256

- Because Grøstl-256 uses a standard AES-based permutation with $r = 8$ and $c = 8$, application is straight-forward.

	s = 1	s = 2	s = 3	s = 4
Grøstl-256	$(2^{16}, 2^{8})$	$(2^{32}, 2^{8})$	$(2^{48}, 2^{8})$	$(2^{64}, 2^{8})$
Ideal permu.	$(2^{32}, -)$	$(2^{64}, -)$	$(2^{96}, -)$	$(2^{128}, -)$

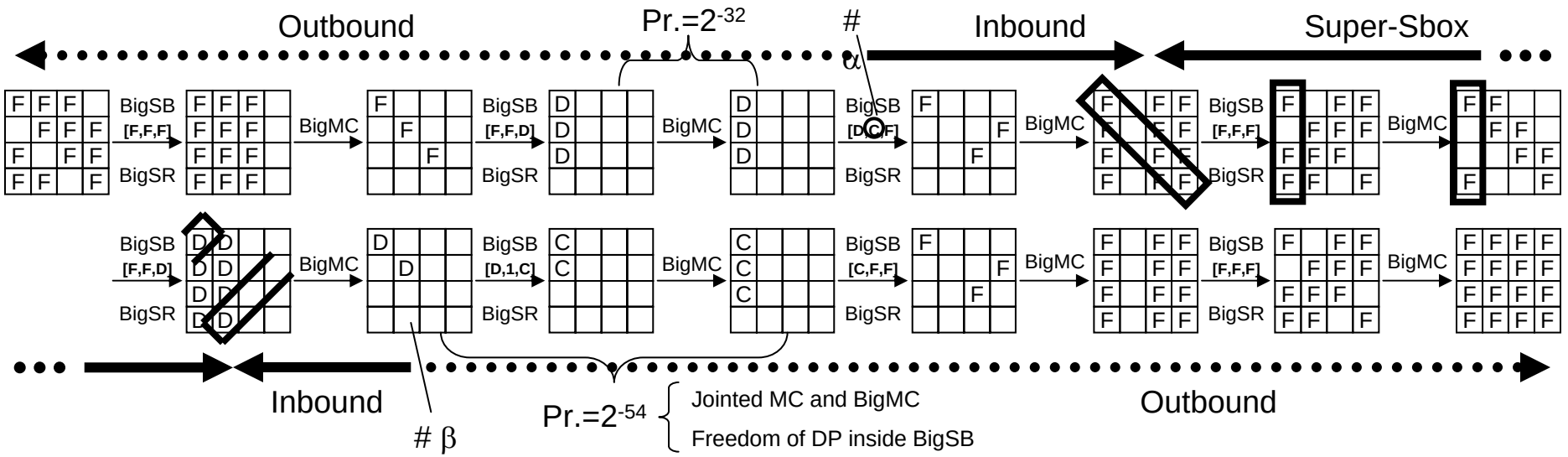
NTT

Application: Full-round ECHO-256

- Because ECHO is not a pure AES-based permutations, and has other internal structures, we can greatly reduce the complexity.
- For example, we considered
  - Increased granularity of word-size difference [P10]
  - Jointed MC and BigMC (also considered by [M10] independently.)
  - Freedom degrees of diff paths inside BigWords.
- As a result, we can find a non-ideal property with 2^{182} in time and 2^{37} in memory.

NTT

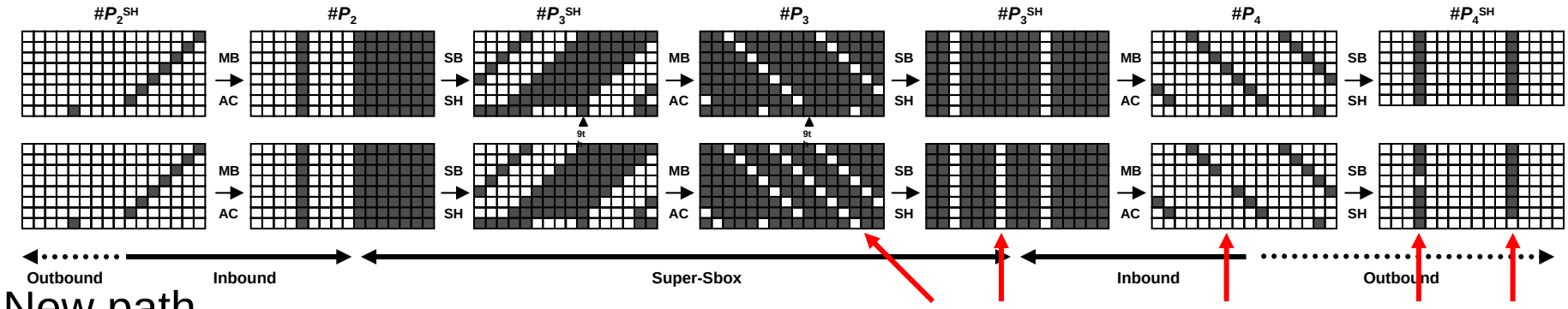
Differential Path for ECHO-256



Application: 7-round Grøstl-512

- Non-full-active Super-Sbox can also improve memory complexity of Grøstl-512 semi-free-start collision attack.
- This attack is specific for the rectangle size state.
- Due to the time limitation, we omit details.

Previous path



New path

We increased non-active bytes.

Conclusions

- We proposed non-full-active Super-Sbox analysis.
- Non-ideal properties of AES-based permutations can be found with a low complexity.
- No impact for the security of hash functions.

Target	#rounds	Attack	Previous (Time, Mem.)	Ours (Time, Mem.)
ECHO-256 (permutation)	8 full	distinguish	$(2^{512},2^{512})$	$(2^{182},2^{37})$
Grøstl-256 (permutation)	8	distinguish	$(2^{112},2^{64})$	$(2^{48},2^{8})$
Grøstl-512 (comp. func.)	7	semi-free-start collision	$(2^{152},2^{64})$	$(2^{152},2^{56})$

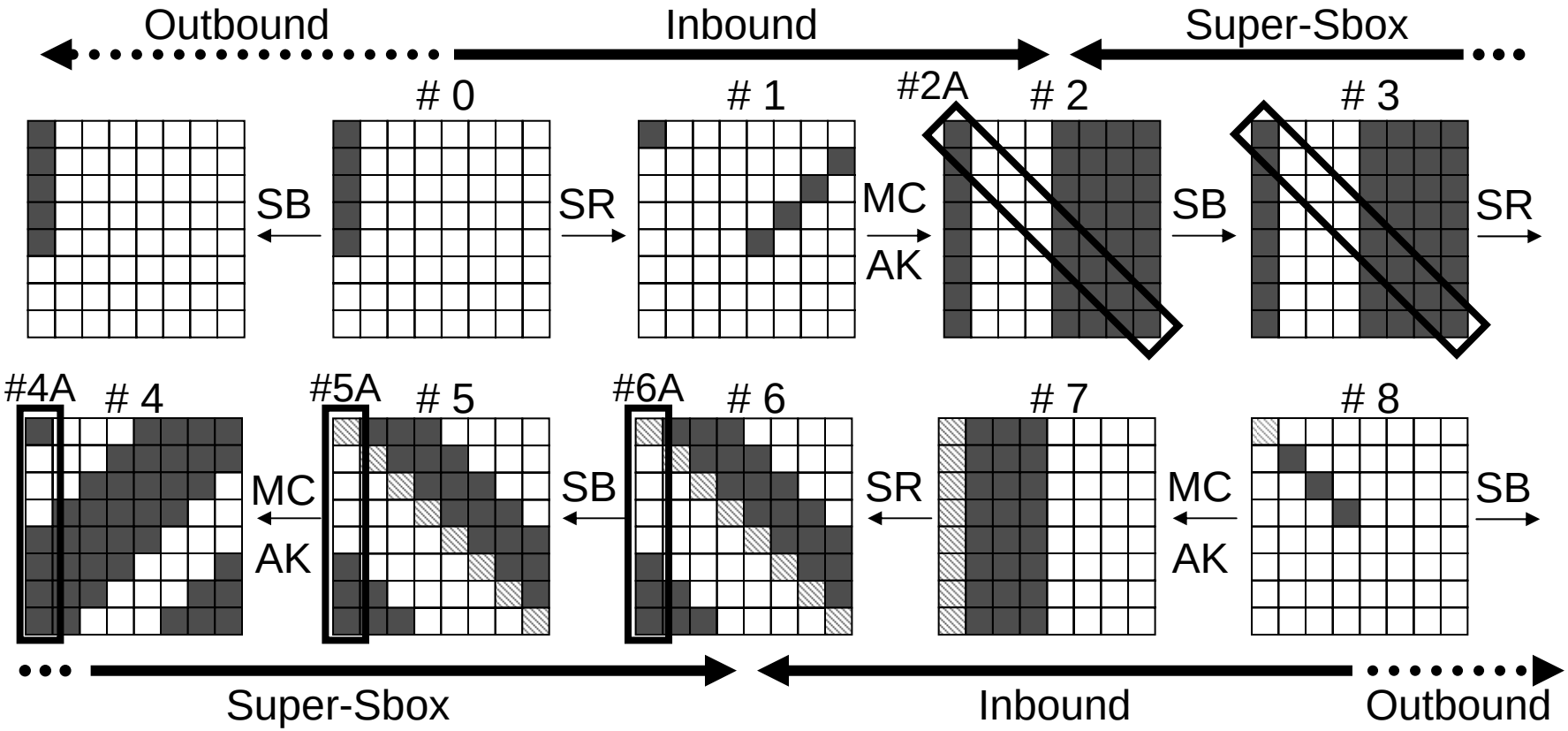
NTT

Thank you for your attention!!

Appendix

Detailed procedure of the
non-full-active Super-Sbox

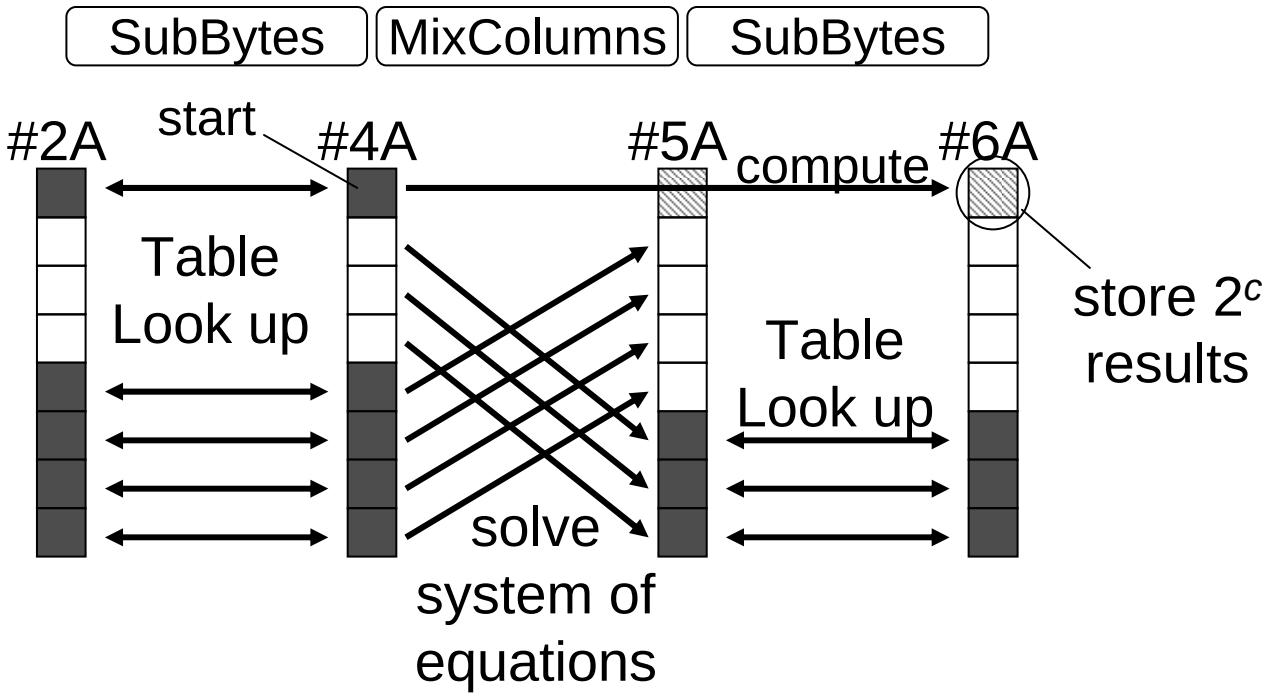
Inbound Phase



□ : non-active ■ : active (difference is fixed)

▨ : active (produce all possible differences)

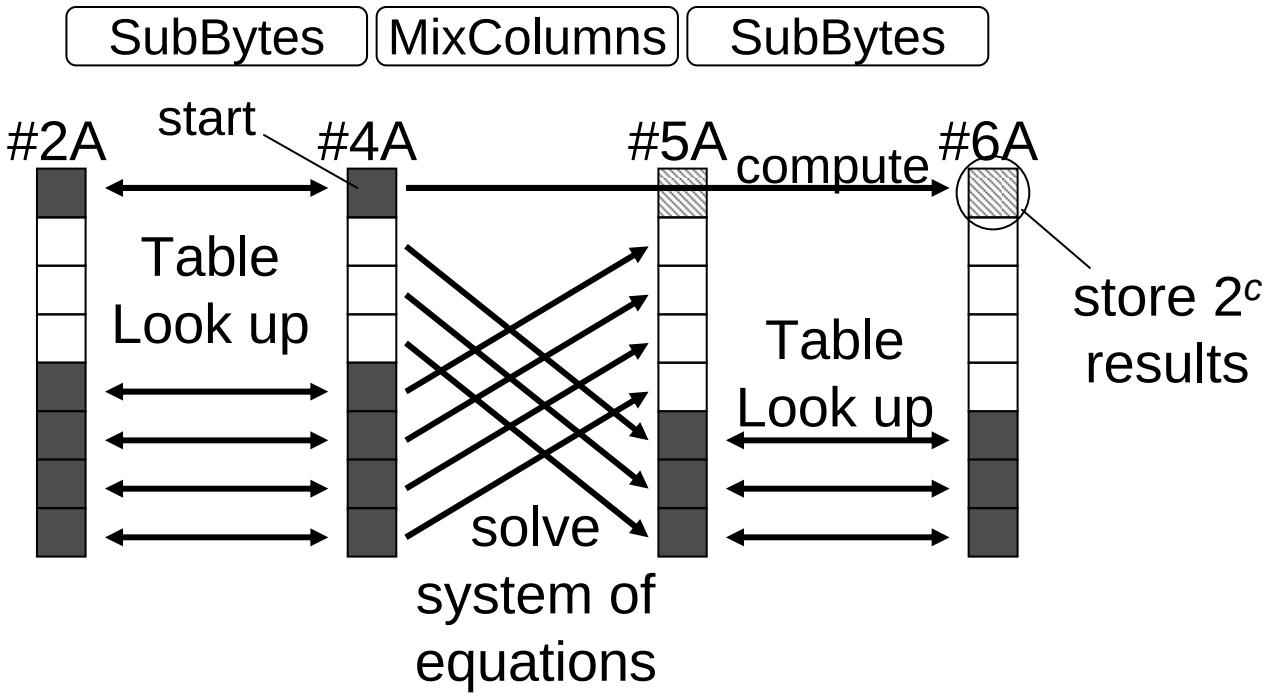
Super-Sbox Computation 1



- Choose a diff of the start byte at #4A.
- Calculate for other r-(s+1) active byte differences of #4A so that r-(s+1) bytes can be non-active at #5A.

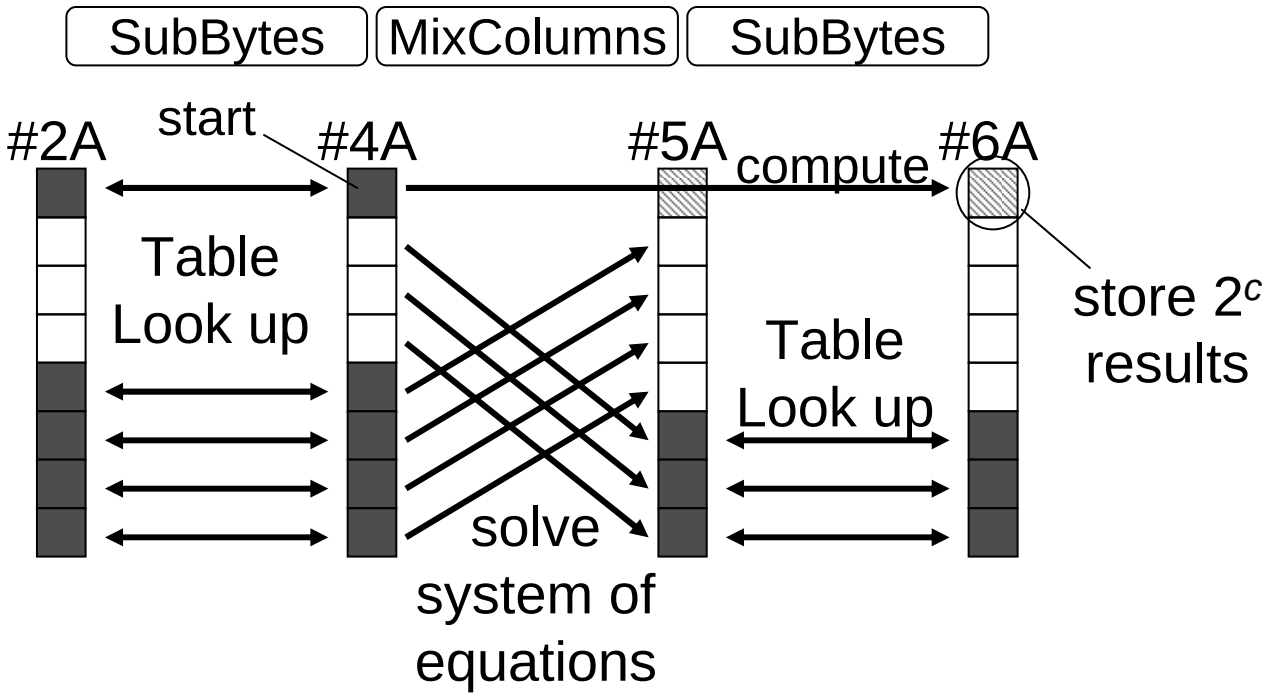
  This is achieved by solving a system of equations.

Super-Sbox Computation 2



- Generate a look up table for each active byte of #4A and #5A by computing diff of all 2^c values.
- By looking up tables, find values of all fixed active bytes between #2A and #4A, and between #5A and #6A.
- Note that values of non-active bytes are still free.

Super-Sbox Computation 3



- By choosing values of non-active bytes of #4A, guarantee that values of active bytes at #5A are achieved. This is done by solving a system of equations.
- Compute the active but non-fixed bytes at #6A. Then, store the differences and values.
- Repeat the procedure 2^c times for diff of start byte at #4A.

Complexity

- In previous page, preparation of look-up tables requires 2^c time and 2^c memory.
- Choosing a start byte, solving systems of equations, table look up cost time 1 and negligible memory.
- Storing the result at #6A requires 1 memory.
- By repeating the attack 2^c times, we spend 2^c time and 2^c memory, and 2^c SPs are generated.
- Hence, the average complexity for 1 SP is time 1.

NTT