# Automated Meet-in-the-Middle Attack Goes to Feistel


**Qingliang Hou** [1] Xiaoyang Dong[2,4,5](✉) Lingyue Qin[2,4] (✉)
Guoyan Zhang[1,3,5] (✉) Xiaoyun Wang[1,2,3,4,5] (✉)

[1]School of Cyber Science and Technology, Shandong University, Qingdao, China

[2]Institute for Advanced Study, BNRist, Tsinghua University, Beijing, China

[3]Key Laboratory of Cryptologic Technology and Information Security, Ministry of Education, Shandong University, Jinan, China

[4]Zhongguancun Laboratory, Beijing, China

[5]Shandong Institute of Blockchain, Jinan, China

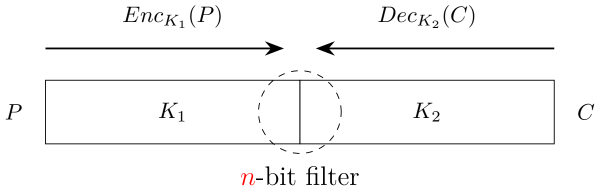


Asiacrypt 2023 / December 4 - 8, 2023

# Outline

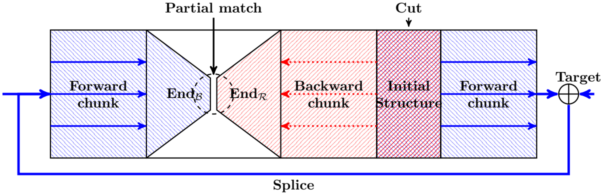
# Meet-in-the-Middle(MitM) attack

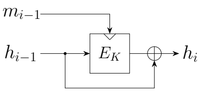
Meet-in-the-Middle(MitM) attack was first introduced by Diffie and Hellman[DH77] in 1977.



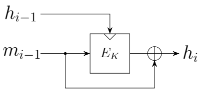

- Forward Encryption: store $K_1$ in table $L_1[Enc_{K_1}(P)]$.
- Backward Decryption: store $K_2$ in table $L_2[Dec_{K_2}(C)]$.
- Find match between $L_1$ and $L_2$.
- $2^{|K_1|+|K_2|-n}(< 2^{|K_1|+|K_2|})$ keys survived.

# MitM Preimage attack on compression function



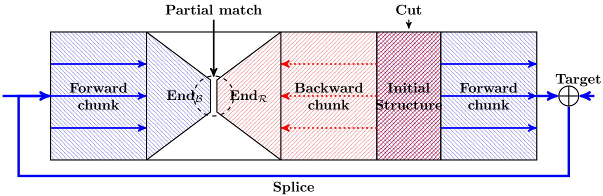

Davies-Meyer(DM) Mode




Matyas-Meyer-Oseas (MMO) Mode

- Splice-and-Cut[SAC:AS08]
  - Splice: the first and last steps are consecutive with feed-forward mechanism
  - Cut: chunk separation with *neutral sets* (■/■)
- Initial structure[EC:SA09]
  - starting steps before chunks where constraints are imposed on neutral sets
- Partial or indirect match[SAC:AS08]
  - match through m(<n) bits
  - match through deterministic relation(i.e. `MixColumns` in AES-like hash).

# MitM Preimage attack on compression function

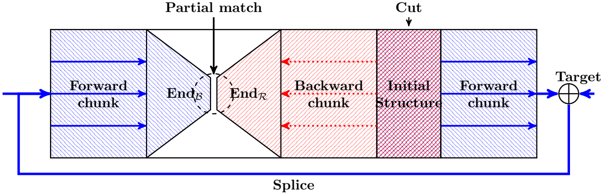



- Initial structure:

$$\begin{cases} \lambda_\mathcal{B}\ \blacksquare,\ \lambda_\mathcal{R}\ \blacksquare,\ \lambda_\mathcal{G}\ \blacksquare \text{ in starting states.} \\ l_\mathcal{B} \text{ constraints on } \blacksquare,\ l_\mathcal{R} \text{ constraints on } \blacksquare. \end{cases}$$

- Forward: $\mathrm{DoF}_\mathcal{B} = \lambda_\mathcal{B} - l_\mathcal{B}$ values of $\blacksquare$.
- Backward: $\mathrm{DoF}_\mathcal{R} = \lambda_\mathcal{R} - l_\mathcal{R}$ values of $\blacksquare$.
- Match: $\mathrm{DoM}$ bytes from $End_\mathcal{B} = End_\mathcal{R}$.

# MitM Preimage attack on compression function




- Initial structure:

$$\begin{cases} \lambda_\mathcal{B}\ \blacksquare,\ \lambda_\mathcal{R}\ \blacksquare,\ \lambda_\mathcal{G}\ \blacksquare \text{ in starting states.} \\ l_\mathcal{B} \text{ constraints on } \blacksquare,\ l_\mathcal{R} \text{ constraints on } \blacksquare. \end{cases}$$

- Forward: $\mathrm{DoF}_\mathcal{B} = \lambda_\mathcal{B} - l_\mathcal{B}$ values of ■.
- Backward: $\mathrm{DoF}_\mathcal{R} = \lambda_\mathcal{R} - l_\mathcal{R}$ values of ■.
- Match: DoM bytes from $End_\mathcal{B} = End_\mathcal{R}$.

## Attack Framework

1. Choose constants for $\lambda_\mathcal{G}$ and $l_\mathcal{B} + l_\mathcal{R}$ bytes.
2. For $\mathrm{DoF}_\mathcal{B}$ values of ■, compute to $End_\mathcal{B}$ and store them in $L_1[End_\mathcal{B}]$.
3. For $\mathrm{DoF}_\mathcal{R}$ values of ■, compute to $End_\mathcal{R}$ and store them in $L_2[End_\mathcal{R}]$.
4. Find match between $L_1$ and $L_2$.

# MitM Preimage attack on compression function

## Attack Framework

1. Choose constants for $\lambda_{\mathcal{G}}$ and $l_{\mathcal{B}} + l_{\mathcal{R}}$ bytes.
2. For $\mathrm{DoF}_{\mathcal{B}}$ values of $\color{blue}\blacksquare$, compute to $End_{\mathcal{B}}$ and store them in $L_1[End_{\mathcal{B}}]$.
3. For $\mathrm{DoF}_{\mathcal{R}}$ values of $\color{red}\blacksquare$, compute to $End_{\mathcal{R}}$ and store them in $L_2[End_{\mathcal{R}}]$.
4. Find match between $L_1$ and $L_2$.

## Cplx.

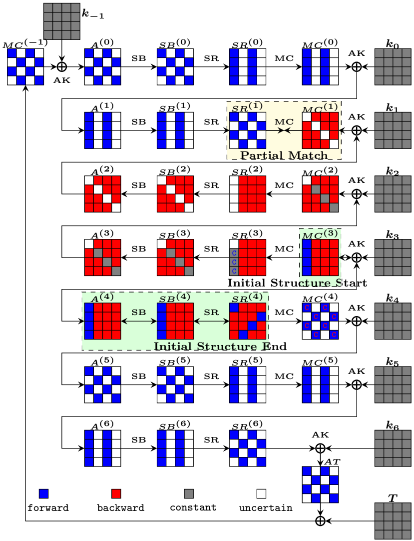
According to $\lambda_{\mathcal{G}} = n - \lambda_{\mathcal{B}} - \lambda_{\mathcal{R}}$, then $\lambda_{\mathcal{G}} + l_{\mathcal{B}} + l_{\mathcal{R}} = n - \mathrm{DoF}_{\mathcal{B}} - \mathrm{DoF}_{\mathcal{R}}$,

$$T = 2^{8\times(n-\mathrm{DoF}_{\mathcal{B}}-\mathrm{DoF}_{\mathcal{R}})} \times \left(2^{8\times \mathrm{DoF}_{\mathcal{B}}} + 2^{8\times \mathrm{DoF}_{\mathcal{R}}} + 2^{8\times(\mathrm{DoF}_{\mathcal{B}}+\mathrm{DoF}_{\mathcal{R}}-\mathrm{DoM})}\right)$$

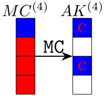
$$= 2^{8\times(n-\min\{\mathrm{DoF}_{\mathcal{B}},\mathrm{DoF}_{\mathcal{R}},\mathrm{DoM}\})}$$

$$M = \min\{2^{8\times \mathrm{DoF}_{\mathcal{B}}},\ 2^{8\times \mathrm{DoF}_{\mathcal{R}}}\}$$

# MitM (pseudo)Preimage attack on 7-round AES-like Hash[FSE:Sas11]

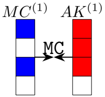



- Starting states($MC^3$): $\lambda_{\mathcal{B}} = 4$ ■, $\lambda_{\mathcal{R}} = 12$ ■
- Initial structure: $l_{\mathcal{B}} = 3$, $l_{\mathcal{R}} = 8$




$$\begin{bmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{bmatrix} \times \begin{bmatrix} MC^{(4)}[0] \\ MC^{(4)}[1] \\ MC^{(4)}[2] \\ MC^{(4)}[3] \end{bmatrix} = \begin{bmatrix} 2 \cdot MC^{(4)}[0] + c_0 \\ - \\ MC^{(4)}[0] + c_1 \\ - \end{bmatrix}$$

- Match through `MixColumn(MC)`




$$\begin{bmatrix} MC^{(1)}[0] \\ - \\ MC^{(1)}[2] \\ - \end{bmatrix} = \begin{bmatrix} e & b & d & 9 \\ 9 & e & b & d \\ d & 9 & e & b \\ b & d & 9 & e \end{bmatrix} \times \begin{bmatrix} - \\ AK^{(1)}[1] \\ AK^{(1)}[2] \\ AK^{(1)}[3] \end{bmatrix}$$

- $T = 2^{8\times(3+8)} \times (2^8 + 2^{8\times 4} + 2^{8\times(1+4-4)}) = 2^{120}$

# Table-based method to solve nonlinear constraints[C:DHSLWH21]

- Initial structure
  - Linear constraints: Solving linear systems of equations
  - Nonlinear constraints: Precomputing the concrete value of constraint constants $\mathfrak{c}_{\mathcal{B}}$ by traversing $\lambda_{\mathcal{B}}$ ■ and store $\mathrm{DoF}_{\mathcal{B}}$ ■ in table $U[\mathfrak{c}_{\mathcal{B}}]$. Similar to ■ and $V[\mathfrak{c}_{\mathcal{R}}]$.

# Table-based method to solve nonlinear constraints[C:DHSLWH21]

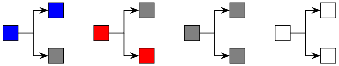
- Initial structure
  - Linear constraints: Solving linear systems of equations
  - Nonlinear constraints: Precomputing the concrete value of constraint constants $\mathfrak{c}_{\mathcal{B}}$ by traversing $\lambda_{\mathcal{B}}$ ■ and store $\mathrm{DoF}_{\mathcal{B}}$ ■ in table $U[\mathfrak{c}_{\mathcal{B}}]$. Similar to ■ and $V[\mathfrak{c}_{\mathcal{R}}]$.

## Improved attack framework

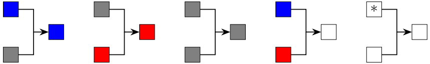
1. Choose constants for the starting states.
2. Build two table to solve nonlinear constraints $\mathfrak{c}_{\mathcal{B}}/\mathfrak{c}_{\mathcal{R}}$ with additional time $\max\{2^{8\times\lambda_{\mathcal{B}}},\ 2^{8\times\lambda_{\mathcal{R}}}\}$ and memory $\max\{2^{8\times\lambda_{\mathcal{B}}},\ 2^{8\times\lambda_{\mathcal{R}}}\}$.
3. For each $\mathfrak{c}_{\mathcal{B}}$, $\mathrm{DoF}_{\mathcal{B}}$ values of ■ are computed to $End_{\mathcal{B}}$ and store in $L_1[End_{\mathcal{B}}]$.
4. For each $\mathfrak{c}_{\mathcal{R}}$, $\mathrm{DoF}_{\mathcal{R}}$ values of ■ are computed to $End_{\mathcal{R}}$ and store in $L_2[End_{\mathcal{R}}]$.
5. Find match between $L_1$ and $L_2$.

# Superposition states and Bi-direction attribute-propagation[C:BGST22]

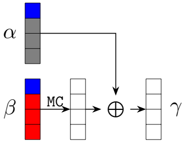
- Intermediate states are separated into two Superposition(SupP) states.
- SupP states are handled independently in linear operation and can be propagated equally in Bi-directions(BiDir).
- SupP states are combined before the next nonlinear operation.



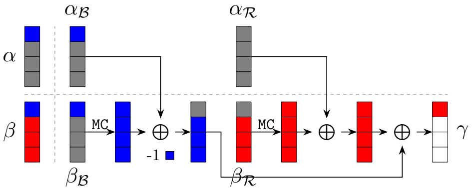
(a) Rules for separation




(b) Rules for combination

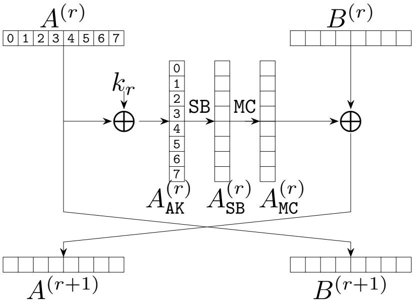



☒ MC and Xor



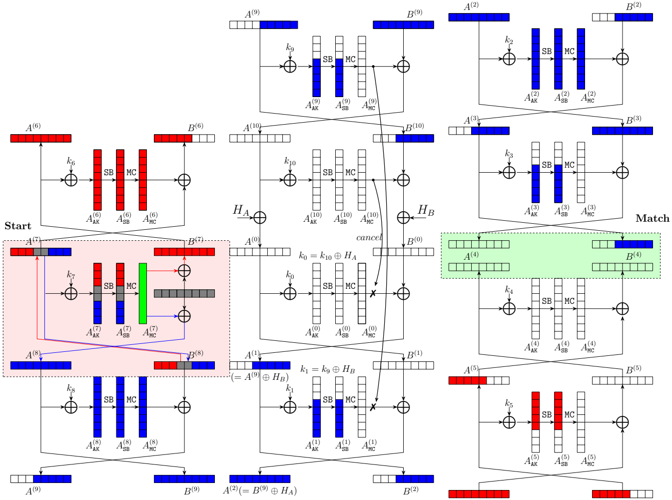

☑ SupP MC and Xor

# Description of Feistel-SP



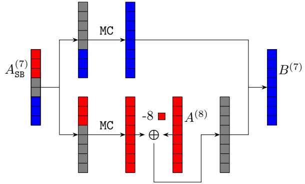

- **SubBytes(SB).** Substitute each byte according to an S-box.
- **ShiftRows$_\pi$(SR).** Permute byte positions according to the permutation $\pi$.
- **MixColumns(MC).** Left-multiply each column by one MDS matrix.
- **AddRoundKey(AK).** Xor with a round key or a round constant.

# MitM Preimage attack on 11-round Feistel-SP[ACNS:Sas13]

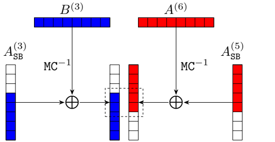



(a) Initial Structure (b) 4-round shrink and link (c) Matching point

- $\mathrm{DoF}_{\mathcal{B}} = 3, \ \mathrm{DoF}_{\mathcal{R}} = 11 - 8 = 3$




- $\mathrm{DoM} = 2$

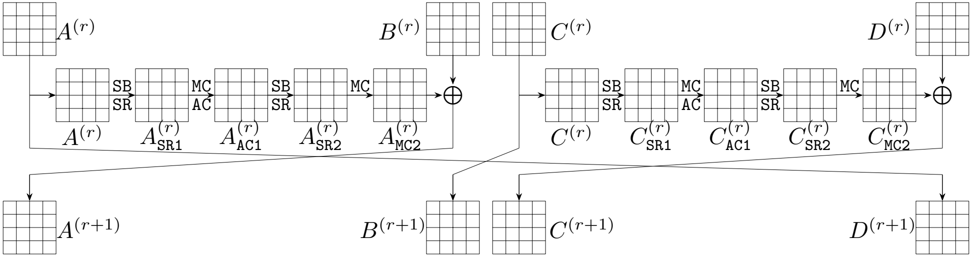



**Cplx.**

$$T = 2^{8\times(16-\min\{3,3,2\})} = 2^{112}$$

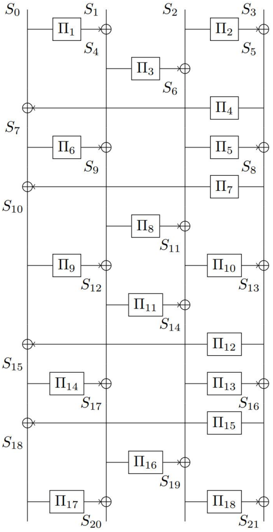
# Guess and Determine Preimage attack on 9-round Simpira-4[C:SS22]

- Simpira-4: 4-brance Type-2 GFN with Double-SP round function




- **Output**: $\text{trunc}_{256}\left(\texttt{Simpira-4}(\texttt{x}) \oplus \texttt{x}\right) = H_A \| H_B$

# Guess and Determine Preimage attack on 9-round Simpira-4[C:SS22]




- **Simplified:**

$$
\begin{array}{lll}
S_0 \oplus S_7 = \Pi_4(S_5) & S_5 \oplus S_8 = \Pi_5(S_6) & S_7 \oplus S_{10} = \Pi_7(S_8) \\
S_6 \oplus S_{11} = \Pi_8(S_9) & S_9 \oplus S_{12} = \Pi_9(S_{10}) & S_8 \oplus S_{13} = \Pi_{10}(S_{11}) \\
S_{11} \oplus S_{14} = \Pi_{11}(S_{12}) & S_{10} \oplus S_{15} = \Pi_{12}(S_{13}) & S_{13} \oplus S_{16} = \Pi_{13}(S_{14}) \\
S_{15} \oplus S_{18} = \Pi_{15}(S_{16}) & S_0 = S_{18} \oplus H_A &
\end{array}
$$

- **Expanded:**

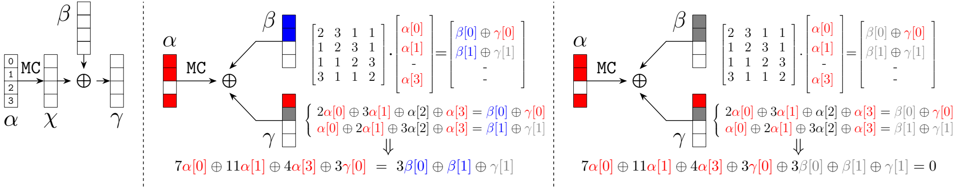
$$\Pi_4(S_5) \oplus \Pi_7(S_8) \oplus \Pi_{12}(S_{13}) \oplus \Pi_{15}(S_{16}) = H_A$$

- **MitM language:**
  - ■ cells: $S_9$, $S_{11}$, ■ cells: $S_8$, ■ cells: $S_{10}$
  - match: $\cancel{S_{10}} \oplus \Pi_4(S_5) \oplus \Pi_7(S_8) = H_A \oplus \Pi_{12}(S_{13}) \oplus \Pi_{15}(S_{16}) \oplus \cancel{S_{10}}$
  - no table is needed(only ■ and ■ appeared in direct matching)

# Outline

# New view of Matching Strategy in MitM attack

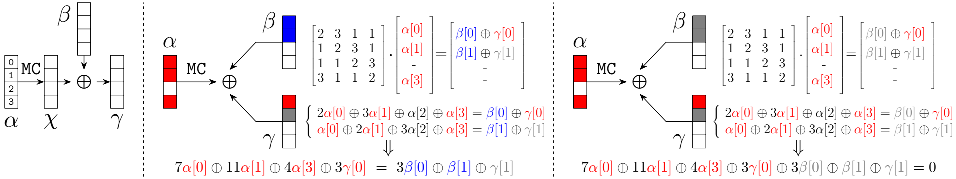
- Detect $\mathrm{M}_{\mathcal{B}}$ bytes filter $\pi_{\mathcal{B}} = 0$ and $\mathrm{M}_{\mathcal{R}}$ bytes filter $\pi_{\mathcal{R}} = 0$




(a) MC then XOR (b) DoM $= 1$ bytes matching (c) $\mathrm{M}_{\mathcal{R}} = 1$ bytes matching

# New view of Matching Strategy in MitM attack

- Detect $\mathrm{M}_{\mathcal{B}}$ bytes filter $\pi_{\mathcal{B}} = 0$ and $\mathrm{M}_{\mathcal{R}}$ bytes filter $\pi_{\mathcal{R}} = 0$




(a) MC then XOR (b) $\mathrm{DoM} = 1$ bytes matching (c) $\mathrm{M}_{\mathcal{R}} = 1$ bytes matching

## New attack framework

1. Choose constants for the Initial structure.
2. For $\mathrm{DoF}_{\mathcal{B}}$ values of ■, compute to $End_{\mathcal{B}}$. If $\pi_{\mathcal{B}} = 0$ holds, store them in $L_1[End_{\mathcal{B}}]$(Filter before store).
3. For $\mathrm{DoF}_{\mathcal{R}}$ values of ■, compute to $End_{\mathcal{R}}$. If $\pi_{\mathcal{R}} = 0$ holds, store them in $L_2[End_{\mathcal{R}}]$.
4. Find match between $L_1$ and $L_2$.

# Comparison with the traditional attack framework

- Traditional:
  1. Choose constants ..
  2. For $2^{\mathrm{DoF}_\mathcal{B}}$ $\color{blue}\blacksquare$, .. store in $L_1[End_\mathcal{B}]$.
  3. For $2^{\mathrm{DoF}_\mathcal{R}}$ $\color{red}\blacksquare$, .. store in $L_2[End_\mathcal{R}]$.
  4. Find match between $L_1$ and $L_2$.
- $M = 2^{8\times\min\{\mathrm{DoF}_\mathcal{B},\mathrm{DoF}_\mathcal{R}\}}$

- Ours:
  1. Choose constants ..
  2. For $2^{\mathrm{DoF}_\mathcal{B}}$ $\color{blue}\blacksquare$, .. if $\pi_\mathcal{B} = 0$, store in $L_1[End_\mathcal{B}]$.
  3. For $2^{\mathrm{DoF}_\mathcal{R}}$ $\color{red}\blacksquare$, .. if $\pi_\mathcal{R} = 0$, store in $L_2[End_\mathcal{R}]$.
  4. Find match between $L_1$ and $L_2$.
- $M = 2^{8\times\min\{\mathrm{DoF}_\mathcal{B}-\mathrm{M}_\mathcal{B},\mathrm{DoF}_\mathcal{R}-\mathrm{M}_\mathcal{R}\}}$

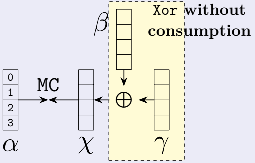
**An extreme example**

Let $\mathrm{DoF}_\mathcal{B} = \mathrm{DoF}_\mathcal{R} = 1$, and $\mathrm{M}_\mathcal{B} = 1$. After 1-byte filter $\pi_\mathcal{B} = 0$, only one solution of $\color{blue}\blacksquare$ is derived. Therefore, no table is needed.

# MILP Model to detect $\pi_{\mathcal{B}} = 0$ and $\pi_{\mathcal{R}} = 0$

## Encoded with a pair 0-1 binary variables $(x_i^\alpha, y_i^\alpha)$

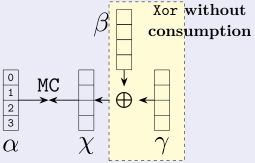
$$\begin{cases} \blacksquare\ \text{Gray}(\mathcal{G}): & (x_i^\alpha, y_i^\alpha) = (1,1), \text{ known in both chunks.} \\ \blacksquare\ \text{Blue}(\mathcal{B}): & (x_i^\alpha, y_i^\alpha) = (1,0), \text{ computed from } \blacksquare \text{ or } \blacksquare. \\ \blacksquare\ \text{Red}(\mathcal{R}): & (x_i^\alpha, y_i^\alpha) = (0,1), \text{ computed from } \blacksquare \text{ or } \blacksquare. \\ \square\ \text{White}(\mathcal{W}): & (x_i^\alpha, y_i^\alpha) = (0,0), \text{ uknown in both chunks.} \end{cases}$$



# MILP Model to detect $\pi_{\mathcal{B}} = 0$ and $\pi_{\mathcal{R}} = 0$

## Encoded with a pair 0-1 binary variables $(x_i^\alpha, y_i^\alpha)$

$$\begin{cases} \blacksquare\ \text{Gray}(\mathcal{G}): & (x_i^\alpha, y_i^\alpha) = (1,1), \text{ known in both chunks.} \\ \blacksquare\ \text{Blue}(\mathcal{B}): & (x_i^\alpha, y_i^\alpha) = (1,0), \text{ computed from } \blacksquare \text{ or } \blacksquare. \\ \blacksquare\ \text{Red}(\mathcal{R}): & (x_i^\alpha, y_i^\alpha) = (0,1), \text{ computed from } \blacksquare \text{ or } \blacksquare. \\ \square\ \text{White}(\mathcal{W}): & (x_i^\alpha, y_i^\alpha) = (0,0), \text{ uknown in both chunks.} \end{cases}$$

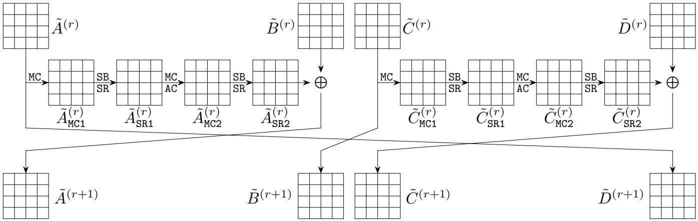



$$\begin{cases} n_{\mathcal{B}}^\alpha = \sum_{i=0}^{3} x_i^\alpha; \\ n_{\mathcal{R}}^\alpha = \sum_{i=0}^{3} y_i^\alpha; \end{cases} \quad \begin{cases} n_{\mathcal{B}}^\chi = \sum_{i=0}^{3} x_i^\chi; \\ n_{\mathcal{R}}^\chi = \sum_{i=0}^{3} y_i^\chi; \end{cases} \quad \begin{cases} n_{\mathcal{G}} = \sum_{i=0}^{3} \mathtt{AND}(x_i^\alpha, y_i^\alpha) + \mathtt{AND}(x_i^\chi, y_i^\chi); \\ \mathrm{M}_{\mathcal{G}} = \max\{0, n_{\mathcal{G}} - 4\}. \end{cases}$$

$$\mathrm{M}_{\mathcal{B}} = \max\left\{0,\ n_{\mathcal{B}}^\alpha + n_{\mathcal{B}}^\chi - 4\right\}, \quad \mathrm{M}_{\mathcal{R}} = \max\left\{0,\ n_{\mathcal{R}}^\alpha + n_{\mathcal{R}}^\chi - \mathrm{M}_{\mathcal{G}} - 4\right\}.$$

# Generalization of Sasaki's Matching Strategy for Feistel

- Equivalent transformation of `Simpira-4`



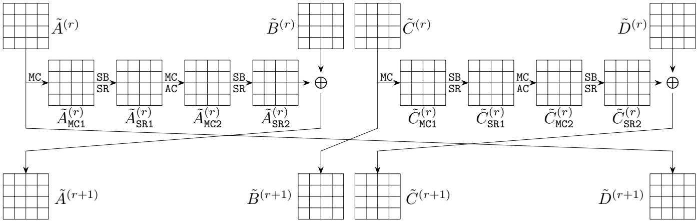

where the round function becomes

$$\mathcal{R}'_i = \mathtt{SR} \circ \mathtt{SB} \circ \mathtt{AC} \circ \mathtt{MC} \circ \mathtt{SR} \circ \mathtt{SB} \circ \mathtt{MC},$$

and the input becomes $\left(\mathtt{MC}^{-1}(A^{(r)}), \mathtt{MC}^{-1}(B^{(r)}), \mathtt{MC}^{-1}(C^{(r)}), \mathtt{MC}^{-1}(D^{(r)})\right)$.

# Generalization of Sasaki's Matching Strategy for Feistel

- Equivalent transformation of `Simpira-4`



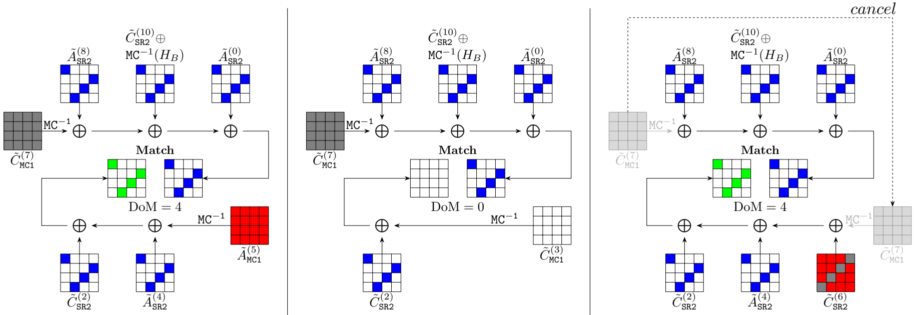

where the round function becomes

$$\mathcal{R}'_i = \texttt{SR} \circ \texttt{SB} \circ \texttt{AC} \circ \texttt{MC} \circ \texttt{SR} \circ \texttt{SB} \circ \texttt{MC},$$

and the input becomes $\left(\texttt{MC}^{-1}(A^{(r)}), \texttt{MC}^{-1}(B^{(r)}), \texttt{MC}^{-1}(C^{(r)}), \texttt{MC}^{-1}(D^{(r)})\right)$.

**Remark.** This is equivalent to searching for the MitM characteristic of the linear transformation of Preimage (or Collision).

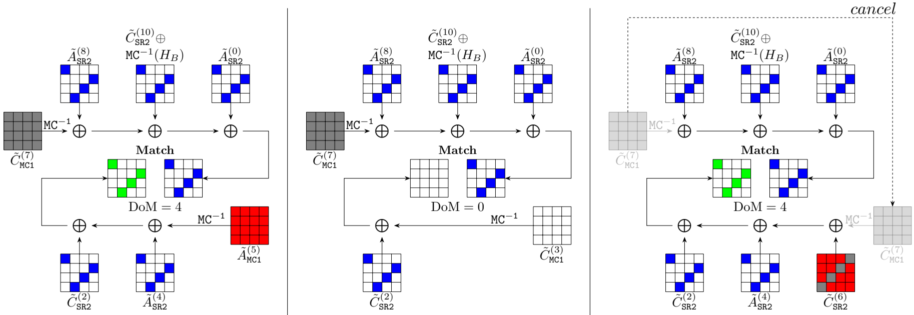
- Full Round match of `Simpira-4`




**(a)** (6,4)-round matching    **(b)** No matching in (6,2)-round    **(c)** Matching through full-round

# Generalization of Sasaki's Matching Strategy for Feistel

- Full Round match of `Simpira-4`




**(a)** (6,4)-round matching **(b)** No matching in (6,2)-round **(c)** Matching through full-round

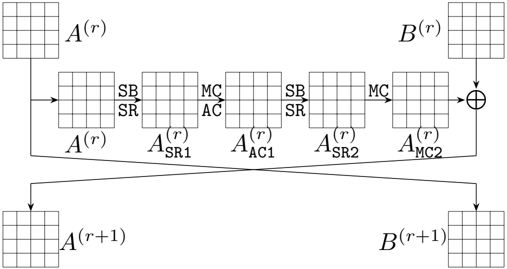
## Advantage

1. Initial structure preserves more useful information or even can be canceled.
2. Only `Xor` operation remained is friendly to program.

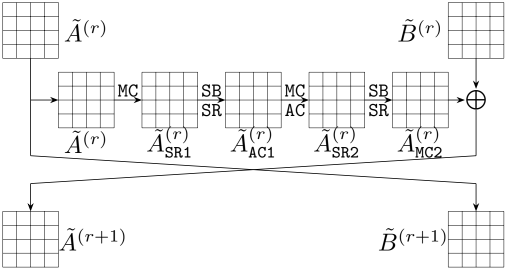
# Outline

# Preimage attack on 7-round Simpira-2

- Description of Simpira-2

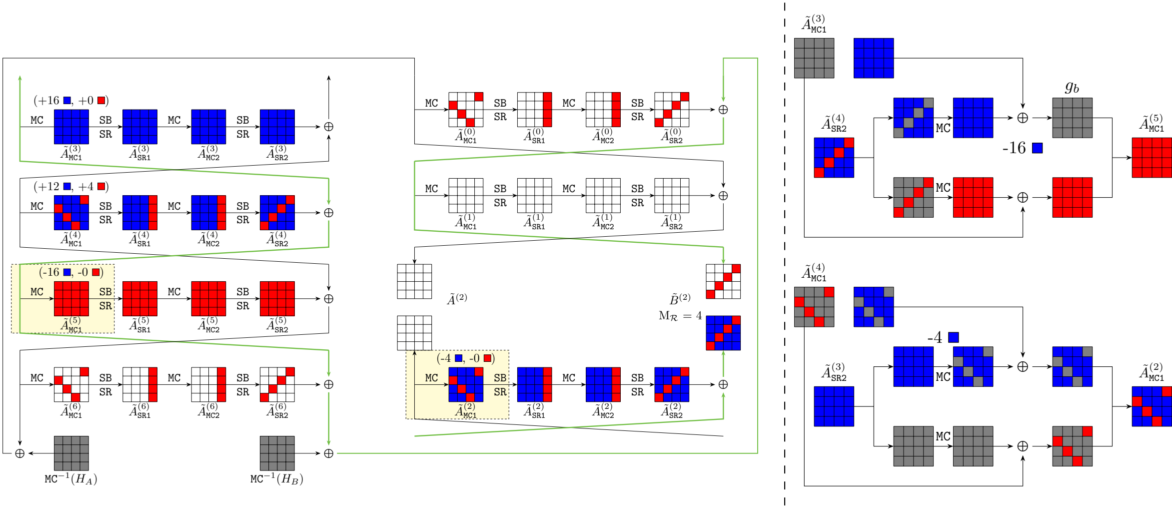



(a) The round function of Simpira-2

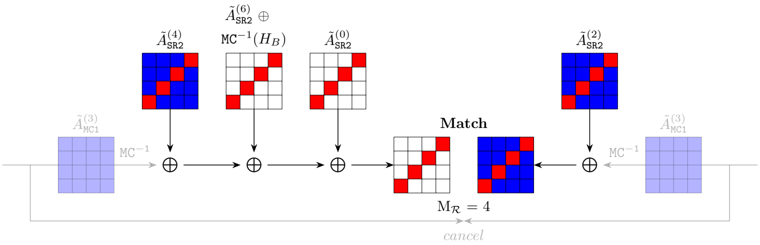



(b) Equivalent transform of Simpira-2

# Preimage attack on 7-round Simpira-2



# Preimage attack on 7-round Simpira-2




## Cplx.

- $\lambda_{\mathcal{B}} = 28$ ■, $\lambda_{\mathcal{R}} = 4$ ■, $l_{\mathcal{B}} = 20$, $l_{\mathcal{R}} = 0$
- $\mathrm{DoF}_{\mathcal{B}} = \lambda_{\mathcal{B}} - l_{\mathcal{B}} = 8$, $\mathrm{DoF}_{\mathcal{R}} = \lambda_{\mathcal{R}} - l_{\mathcal{R}} = 4$, $\mathrm{DoM} = 0$, $\mathrm{M}_{\mathcal{B}} = 0$, $\mathrm{M}_{\mathcal{R}} = 4$

$$T = 2^{8\times 28} + 2^{8\times(32-\min\{8,4,4\})} \approx 2^{225}$$

# Summary of applications to preimage and collision attacks

| Target | Attacks | Settings | Rounds | Time | Memory | Generic | Ref. |
|---|---|---|---|---|---|---|---|
| Feistel-SP-128 | Preimage | Classical | 11 | $2^{112}$ | $2^{24}$ | $2^{128}$ | [ACNS:Sas13] |
| | | Classical | **12** | $\mathbf{2^{113}}$ | $\mathbf{2^{48}}$ | $2^{128}$ | [This] |
| Simpira-2 | Preimage | Classical | 5/15 | $2^{128}$ | - | $2^{256}$ | [C:SS22] |
| | | Quantum | 5/15 | $2^{64}$ | - | $2^{128}$ | [C:SS22] |
| | | Classical | **7/15** | $\mathbf{2^{225}}$ | $\mathbf{2^{96}}$ | $2^{256}$ | [This] |
| Simpira-4 | Preimage | Classical | 9/15 | $2^{128}$ | - | $2^{256}$ | [C:SS22] |
| | | Quantum | 9/15 | $2^{64}$ | - | $2^{128}$ | [C:SS22] |
| | | Classical | **11/15** | $\mathbf{2^{225}}$ | $\mathbf{2^{160}}$ | $2^{256}$ | [This] |
| Simpira-6 | Preimage | Classical | **11/15** | $\mathbf{2^{193.6}}$ | $\mathbf{2^{193}}$ | $2^{256}$ | [This] |
| Lesamnta-LW | Collision | Classical | 11/64 | $2^{97}$ | $2^{96}$ | $2^{128}$ | [ICISC:HIK10] |
| | | Classical | **17/64** | $\mathbf{2^{113.58}}$ | $\mathbf{2^{112}}$ | $2^{128}$ | [This] |
| | | Classical | **20/64** | $\mathbf{2^{124}}$ | $\mathbf{2^{124}}$ | $2^{128}$ | [This] |
| Areion256-DM | Preimage | Classical | 5/10 | $2^{248}$ | $2^{8}$ | $2^{256}$ | [CHES:IIL23] |
| | | Classical | 5/10 | $\mathbf{2^{193}}$ | $\mathbf{2^{88}}$ | $2^{256}$ | [This] |
| | | Classical | **7/10** | $\mathbf{2^{240}}$ | $\mathbf{2^{64}}$ | $2^{256}$ | [This] |
| Areion512-DM | Preimage | Classical | 10/15 | $2^{248}$ | $2^{8}$ | $2^{256}$ | [CHES:IIL23] |
| | | Classical | **11/15** | $\mathbf{2^{241}}$ | $\mathbf{2^{48}}$ | $2^{256}$ | [This] |

# Outline

# Conclusion

- Automated MitM Preimage and Collision attack on the linear transformation of Feistel-based hash functions with full-round match.
- Detect matching bytes filter with same color to reduce memory.
- Leading to improved or first MitM attack on `Simpira`, `Areion`, and `Lesamnta-LW`.
- Limitation:
  - In Double-SP round function, the output can be easily unknown with only one input unknown byte, so how to avoid it in SupP states even with additional cost?

**Full Version:** `https://eprint.iacr.org/2023/1359.pdf`
**Source Code:** `https://github.com/Hql-code/MitM-Feistel.git`

Thank you!