The State-Test Technique on Differential Attacks:

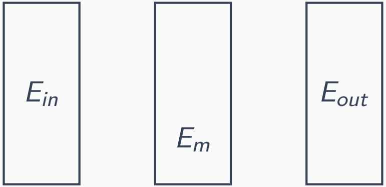
a 26-Round Attack on CRAFT and Other Applications

Dounia M'Foukh   María Naya-Plasencia   Patrick Neumann

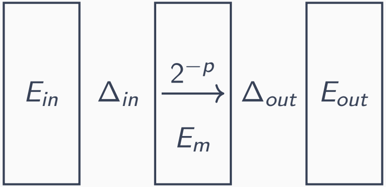
Asiacrypt, December 10, 2025

Inria Paris, France

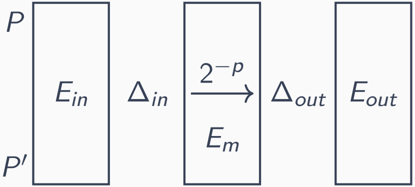
Key Recovery in Differential Attacks



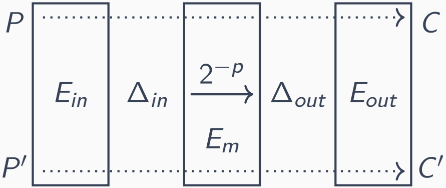
Key Recovery in Differential Attacks



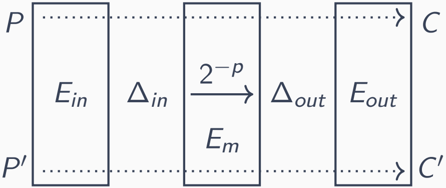
Key Recovery in Differential Attacks



Key Recovery in Differential Attacks

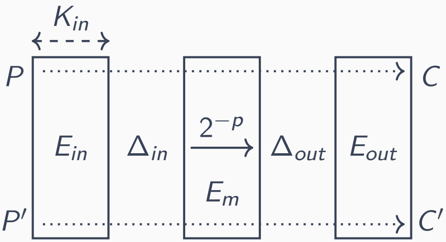


Key Recovery in Differential Attacks



Problem: efficiently generate tuples $(P, P', C, C', K_{in}, K_{out})$ with

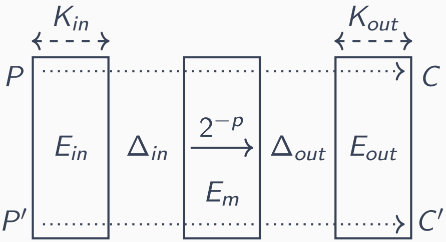
Key Recovery in Differential Attacks



Problem: efficiently generate tuples $(P, P', C, C', K_{in}, K_{out})$ with

- $E_{in}(P) + E_{in}(P') = \Delta_{in}$ using K_{in}

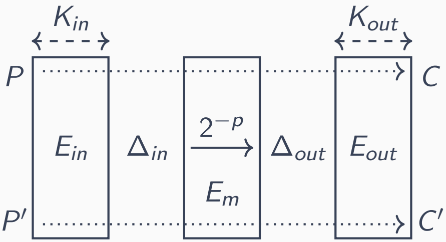
Key Recovery in Differential Attacks



Problem: efficiently generate tuples $(P, P', C, C', K_{in}, K_{out})$ with

- $E_{in}(P) + E_{in}(P') = \Delta_{in}$ using K_{in}
- $E_{out}^{-1}(C) + E_{out}^{-1}(C') = \Delta_{out}$ using K_{out}

Key Recovery in Differential Attacks

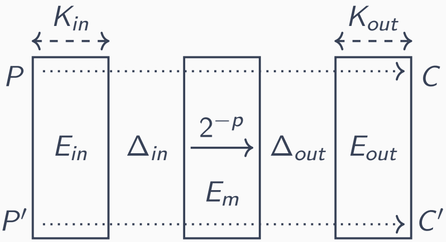


(Classical) Differential Attacks

Problem: efficiently generate tuples $(P, P', C, C', K_{in}, K_{out})$ with

- $E_{in}(P) + E_{in}(P') = \Delta_{in}$ using K_{in}
- $E_{out}^{-1}(C) + E_{out}^{-1}(C') = \Delta_{out}$ using K_{out}

Key Recovery in Differential Attacks



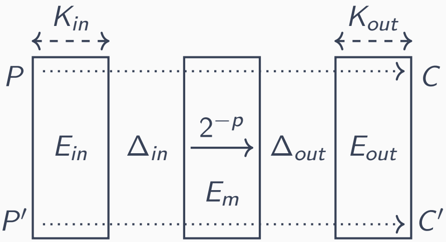
Problem: efficiently generate tuples $(P, P', C, C', K_{in}, K_{out})$ with

- $E_{in}(P) + E_{in}(P') = \Delta_{in}$ using K_{in}
- $E_{out}^{-1}(C) + E_{out}^{-1}(C') = \Delta_{out}$ using K_{out}

(Classical) Differential Attacks

1. Generate (P, P') that can lead to Δ_{in}

Key Recovery in Differential Attacks



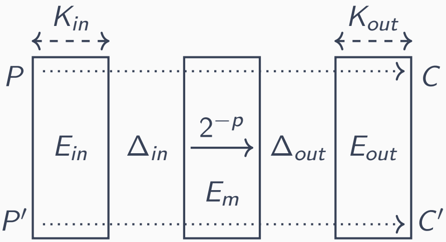
Problem: efficiently generate tuples $(P, P', C, C', K_{in}, K_{out})$ with

- $E_{in}(P) + E_{in}(P') = \Delta_{in}$ using K_{in}
- $E_{out}^{-1}(C) + E_{out}^{-1}(C') = \Delta_{out}$ using K_{out}

(Classical) Differential Attacks

1. Generate (P, P') that can lead to Δ_{in}
2. Encrypt and filter if (C, C') cannot lead to Δ_{out}

Key Recovery in Differential Attacks



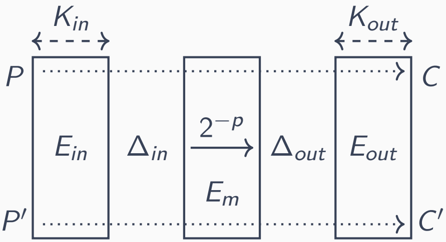
Problem: efficiently generate tuples $(P, P', C, C', K_{in}, K_{out})$ with

- $E_{in}(P) + E_{in}(P') = \Delta_{in}$ using K_{in}
- $E_{out}^{-1}(C) + E_{out}^{-1}(C') = \Delta_{out}$ using K_{out}

(Classical) Differential Attacks

1. Generate (P, P') that can lead to Δ_{in}
2. Encrypt and filter if (C, C') cannot lead to Δ_{out}
3. Verify Δ_{in} and Δ_{out} by guessing K_{in} and K_{out}

Key Recovery in Differential Attacks



Problem: efficiently generate tuples $(P, P', C, C', K_{in}, K_{out})$ with

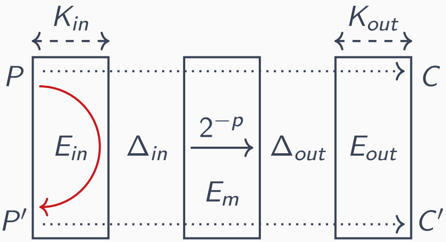
- $E_{in}(P) + E_{in}(P') = \Delta_{in}$ using K_{in}
- $E_{out}^{-1}(C) + E_{out}^{-1}(C') = \Delta_{out}$ using K_{out}

(Classical) Differential Attacks

1. Generate (P, P') that can lead to Δ_{in}
2. Encrypt and filter if (C, C') cannot lead to Δ_{out}
3. Verify Δ_{in} and Δ_{out} by guessing K_{in} and K_{out}

Differential MitM Attacks [BDDLNP23]

Key Recovery in Differential Attacks



Problem: efficiently generate tuples $(P, P', C, C', K_{in}, K_{out})$ with

- $E_{in}(P) + E_{in}(P') = \Delta_{in}$ using K_{in}
- $E_{out}^{-1}(C) + E_{out}^{-1}(C') = \Delta_{out}$ using K_{out}

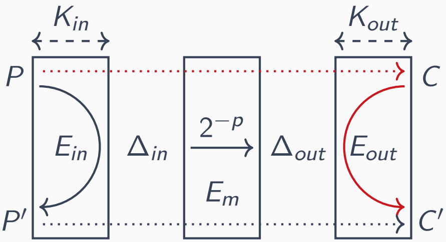
(Classical) Differential Attacks

1. Generate (P, P') that can lead to Δ_{in}
2. Encrypt and filter if (C, C') cannot lead to Δ_{out}
3. Verify Δ_{in} and Δ_{out} by guessing K_{in} and K_{out}

Differential MitM Attacks [BDDLNP23]

1.1. Guess K_{in} and compute P' based on P

Key Recovery in Differential Attacks



Problem: efficiently generate tuples $(P, P', C, C', K_{in}, K_{out})$ with

- $E_{in}(P) + E_{in}(P') = \Delta_{in}$ using K_{in}
- $E_{out}^{-1}(C) + E_{out}^{-1}(C') = \Delta_{out}$ using K_{out}

(Classical) Differential Attacks

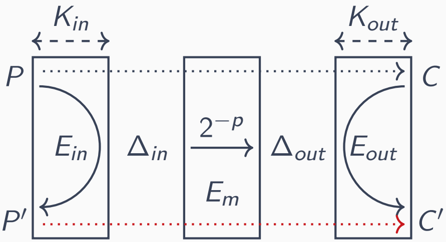
1. Generate (P, P') that can lead to Δ_{in}
2. Encrypt and filter if (C, C') cannot lead to Δ_{out}
3. Verify Δ_{in} and Δ_{out} by guessing K_{in} and K_{out}

Differential MitM Attacks [BDDLNP23]

1.1. Guess K_{in} and compute P' based on P

1.2. Guess K_{out} and compute C' based on $C = E(P)$

Key Recovery in Differential Attacks



Problem: efficiently generate tuples $(P, P', C, C', K_{in}, K_{out})$ with

- $E_{in}(P) + E_{in}(P') = \Delta_{in}$ using K_{in}
- $E^{-1}_{out}(C) + E^{-1}_{out}(C') = \Delta_{out}$ using K_{out}

(Classical) Differential Attacks

1. Generate (P, P') that can lead to Δ_{in}
2. Encrypt and filter if (C, C') cannot lead to Δ_{out}
3. Verify Δ_{in} and Δ_{out} by guessing K_{in} and K_{out}

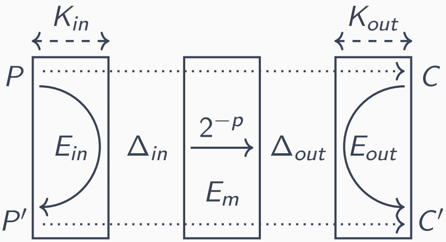
Differential MitM Attacks [BDDLNP23]

1.1. Guess K_{in} and compute P' based on P

1.2. Guess K_{out} and compute C' based on $C = E(P)$

2. Match on $E(P') \stackrel{?}{=} C'$

Key Recovery in Differential Attacks



Problem: efficiently generate tuples $(P, P', C, C', K_{in}, K_{out})$ with

- $E_{in}(P) + E_{in}(P') = \Delta_{in}$ using K_{in}
- $E_{out}^{-1}(C) + E_{out}^{-1}(C') = \Delta_{out}$ using K_{out}

(Classical) Differential Attacks

1. Generate (P, P') that can lead to Δ_{in}
2. Encrypt and filter if (C, C') cannot lead to Δ_{out}
3. Verify Δ_{in} and Δ_{out} by guessing K_{in} and K_{out}

Differential MitM Attacks [BDDLNP23]

1.1. Guess K_{in} and compute P' based on P

1.2. Guess K_{out} and compute C' based on $C = E(P)$

2. Match on $E(P') \stackrel{?}{=} C'$

State-Test Technique [BNPS14]

- Reducing key guesses improves complexity of attack

State-Test Technique [BNPS14]

- Reducing key guesses improves complexity of attack
- Idea: guess part of the state instead of the key involved in its computation

State-Test Technique [BNPS14]

- Reducing key guesses improves complexity of attack
- Idea: guess part of the state instead of the key involved in its computation
- Example: $y = S(x_0 + k_0) + S(x_1 + k_1)$

State-Test Technique [BNPS14]

- Reducing key guesses improves complexity of attack
- Idea: guess part of the state instead of the key involved in its computation
- Example: $y = S(x_0 + k_0) + S(x_1 + k_1)$
  - A guess of y is more efficient than guessing k_0 and k_1

State-Test Technique [BNPS14]

- Reducing key guesses improves complexity of attack
- Idea: guess part of the state instead of the key involved in its computation
- Example: $y = S(x_0 + k_0) + S(x_1 + k_1)$
    - A guess of y is more efficient than guessing k_0 and k_1
- Introduced in the context of impossible-differential Attacks [BNPS14]

State-Test Technique [BNPS14]

- Reducing key guesses improves complexity of attack
- Idea: guess part of the state instead of the key involved in its computation
- Example: $y = S(x_0 + k_0) + S(x_1 + k_1)$
  - A guess of y is more efficient than guessing k_0 and k_1
- Introduced in the context of impossible-differential Attacks [BNPS14]
  - Needs state guesses to partition the keys

State-Test Technique [BNPS14]

- Reducing key guesses improves complexity of attack
- Idea: guess part of the state instead of the key involved in its computation
- Example: $y = S(x_0 + k_0) + S(x_1 + k_1)$
  - A guess of y is more efficient than guessing k_0 and k_1
- Introduced in the context of impossible-differential Attacks [BNPS14]
  - Needs state guesses to partition the keys
- Also used in the context of differential-MitM attacks [AKMMNP24]

State-Test Technique [BNPS14]

- Reducing key guesses improves complexity of attack
- Idea: guess part of the state instead of the key involved in its computation
- Example: $y = S(x_0 + k_0) + S(x_1 + k_1)$
  - A guess of y is more efficient than guessing k_0 and k_1
- Introduced in the context of impossible-differential Attacks [BNPS14]
  - Needs state guesses to partition the keys
- Also used in the context of differential-MitM attacks [AKMMNP24]
  - State guesses define non-linear equations in the key

State-Test Technique [BNPS14]

- Reducing key guesses improves complexity of attack
- Idea: guess part of the state instead of the key involved in its computation
- Example: $y = S(x_0 + k_0) + S(x_1 + k_1)$
  - A guess of y is more efficient than guessing k_0 and k_1
- Introduced in the context of impossible-differential Attacks [BNPS14]
  - Needs state guesses to partition the keys
- Also used in the context of differential-MitM attacks [AKMMNP24]
  - State guesses define non-linear equations in the key
  - Solving them allows to recover more key material

Our Contribution

State-Test Technique in Differential & Differential-Linear Attacks

Our Contribution

State-Test Technique in Differential & Differential-Linear Attacks

- With counter

Our Contribution

State-Test Technique in Differential & Differential-Linear Attacks

- With counter
  - If part of counter: need to partition the key (*cf.* impossible differential)

Our Contribution

State-Test Technique in Differential & Differential-Linear Attacks

- With counter
  - If part of counter: need to partition the key (*cf.* impossible differential)
  - If *not* part of counter: yield over-define system of equations

Our Contribution

State-Test Technique in Differential & Differential-Linear Attacks

- With counter
  - If part of counter: need to partition the key (*cf.* impossible differential)
  - If *not* part of counter: yield over-define system of equations
  - Additional filtering & more key material recovered (*cf.* differential MitM)

Our Contribution

State-Test Technique in Differential & Differential-Linear Attacks

- With counter
  - If part of counter: need to partition the key (*cf.* impossible differential)
  - If *not* part of counter: yield over-define system of equations
  - Additional filtering & more key material recovered (*cf.* differential MitM)
- Without counter: similar to differential MitM attacks

Our Contribution

State-Test Technique in Differential & Differential-Linear Attacks

- With counter
  - If part of counter: need to partition the key (*cf.* impossible differential)
  - If *not* part of counter: yield over-define system of equations
  - Additional filtering & more key material recovered (*cf.* differential MitM)
- Without counter: similar to differential MitM attacks
- Improve best known attack on Pride

Our Contribution

State-Test Technique in Differential & Differential-Linear Attacks

- With counter
  - If part of counter: need to partition the key (*cf.* impossible differential)
  - If *not* part of counter: yield over-define system of equations
  - Additional filtering & more key material recovered (*cf.* differential MitM)
- Without counter: similar to differential MitM attacks
- Improve best known attack on Pride

Insights into the Applicability of the State-Test Technique

Our Contribution

State-Test Technique in Differential & Differential-Linear Attacks

- With counter
  - If part of counter: need to partition the key (*cf.* impossible differential)
  - If *not* part of counter: yield over-define system of equations
  - Additional filtering & more key material recovered (*cf.* differential MitM)
- Without counter: similar to differential MitM attacks
- Improve best known attack on Pride

Insights into the Applicability of the State-Test Technique

First Attacks on 24 to 26 Rounds of CRAFT

Our Contribution

State-Test Technique in Differential & Differential-Linear Attacks

- With counter
  - If part of counter: need to partition the key (*cf.* impossible differential)
  - If *not* part of counter: yield over-define system of equations
  - Additional filtering & more key material recovered (*cf.* differential MitM)
- Without counter: similar to differential MitM attacks
- Improve best known attack on Pride

Insights into the Applicability of the State-Test Technique

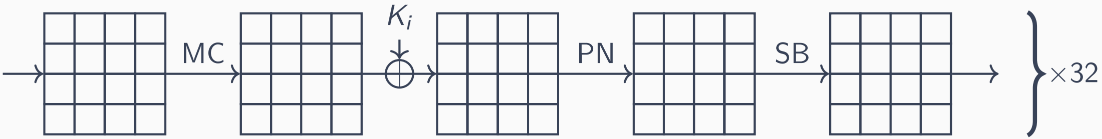
First Attacks on 24 to 26 Rounds of CRAFT

The Tweakable Block Cipher Craft [BLMR19]

- 64 bit state, represented as a 4×4 matrix, and 128 bit key (K_0, K_1)

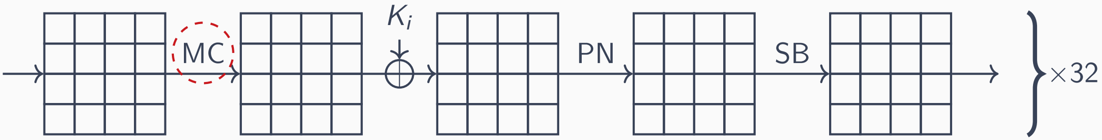
The Tweakable Block Cipher Craft [BLMR19]

- 64 bit state, represented as a 4×4 matrix, and 128 bit key (K_0, K_1)



The Tweakable Block Cipher Craft [BLMR19]

- 64 bit state, represented as a 4×4 matrix, and 128 bit key (K_0, K_1)

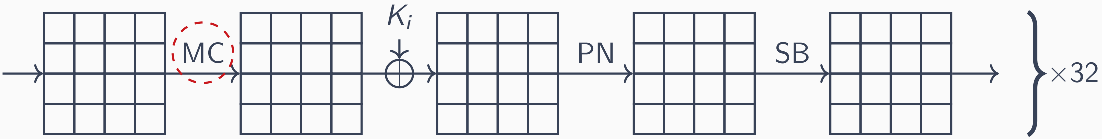


Mix Columns

$$\begin{bmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

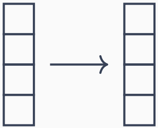
The Tweakable Block Cipher Craft [BLMR19]

- 64 bit state, represented as a 4×4 matrix, and 128 bit key (K_0, K_1)



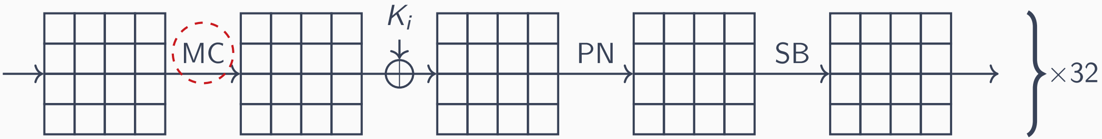
Mix Columns

$$\begin{bmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



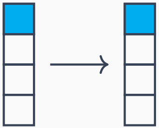
The Tweakable Block Cipher Craft [BLMR19]

- 64 bit state, represented as a 4×4 matrix, and 128 bit key (K_0, K_1)



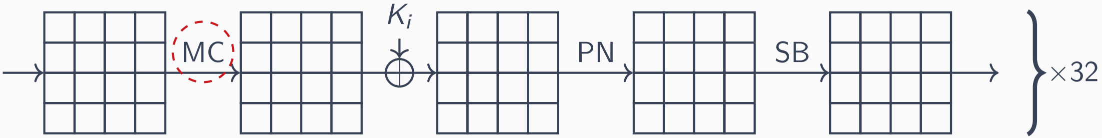
Mix Columns

$$\begin{bmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



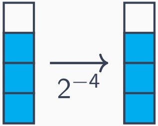
The Tweakable Block Cipher Craft [BLMR19]

- 64 bit state, represented as a 4×4 matrix, and 128 bit key (K_0, K_1)



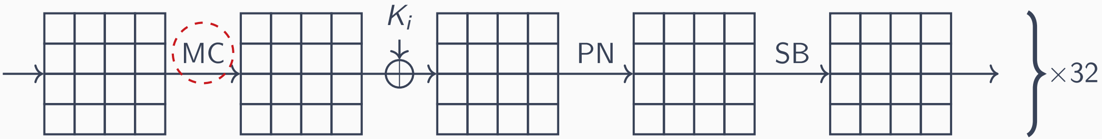
Mix Columns

$$\begin{bmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



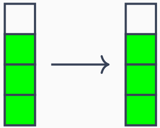
The Tweakable Block Cipher Craft [BLMR19]

- 64 bit state, represented as a 4×4 matrix, and 128 bit key (K_0, K_1)



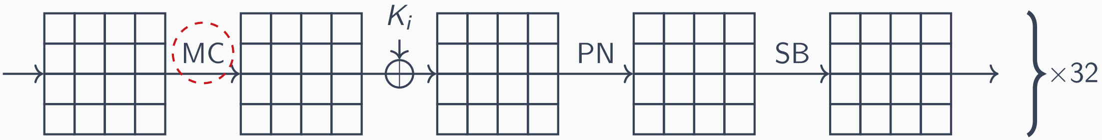
Mix Columns

$$\begin{bmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



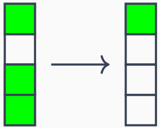
The Tweakable Block Cipher Craft [BLMR19]

- 64 bit state, represented as a 4×4 matrix, and 128 bit key (K_0, K_1)



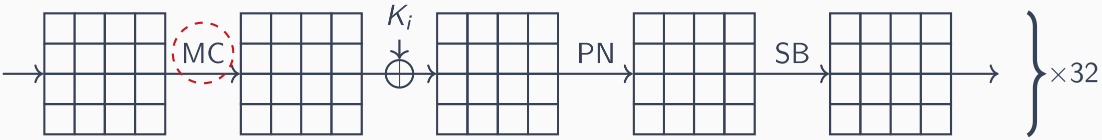
Mix Columns

$$\begin{bmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



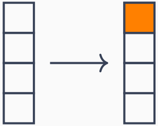
The Tweakable Block Cipher Craft [BLMR19]

- 64 bit state, represented as a 4×4 matrix, and 128 bit key (K_0, K_1)



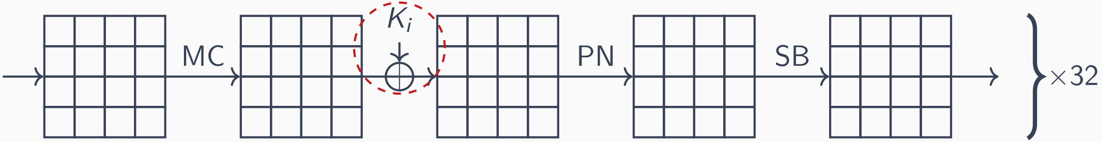
Mix Columns

$$\begin{bmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



The Tweakable Block Cipher Craft [BLMR19]

- 64 bit state, represented as a 4×4 matrix, and 128 bit key (K_0, K_1)

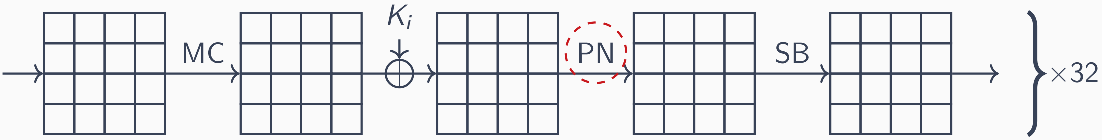


Key Addition

Add $K_{i \bmod 2}$ in round i

The Tweakable Block Cipher Craft [BLMR19]

- 64 bit state, represented as a 4×4 matrix, and 128 bit key (K_0, K_1)



Permute Nibbles

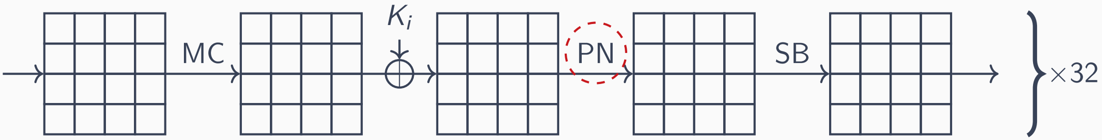
0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15

→

15	12	13	14
10	9	8	11
6	5	4	7
1	2	3	0

The Tweakable Block Cipher Craft [BLMR19]

- 64 bit state, represented as a 4×4 matrix, and 128 bit key (K_0, K_1)



Permute Nibbles

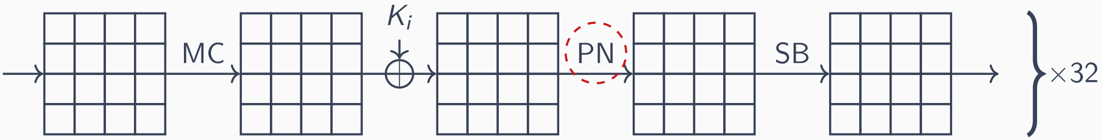
0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15

→

15	12	13	14
10	9	8	11
6	5	4	7
1	2	3	0

The Tweakable Block Cipher Craft [BLMR19]

- 64 bit state, represented as a 4×4 matrix, and 128 bit key (K_0, K_1)



Permute Nibbles

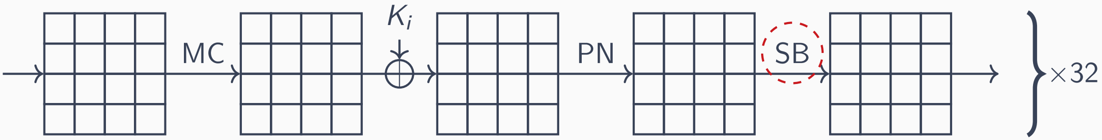
0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15

$\longrightarrow$

15	12	13	14
10	9	8	11
6	5	4	7
1	2	3	0

The Tweakable Block Cipher Craft [BLMR19]

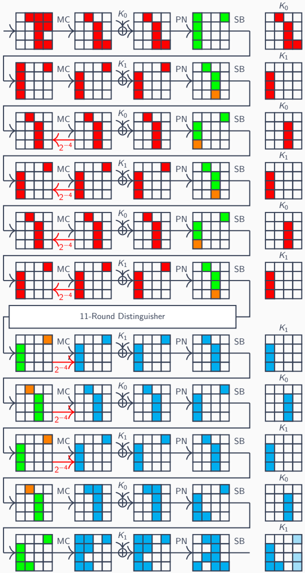
- 64 bit state, represented as a 4×4 matrix, and 128 bit key (K_0, K_1)



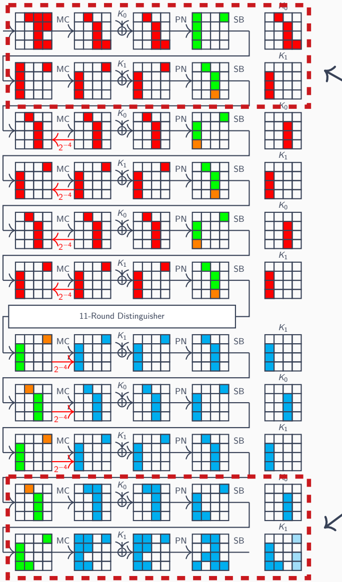
S-Box Layer

Apply s-box to all cells

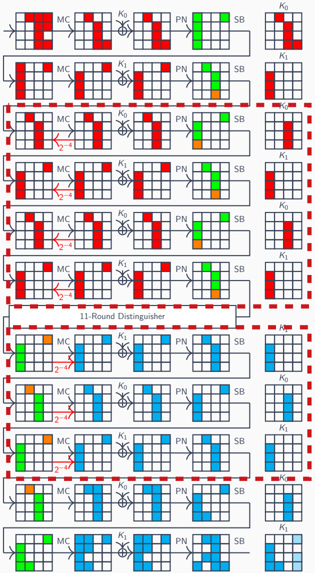
Differential Meet-in-the-Middle Attack on 23-Rounds [AKMMNP24]



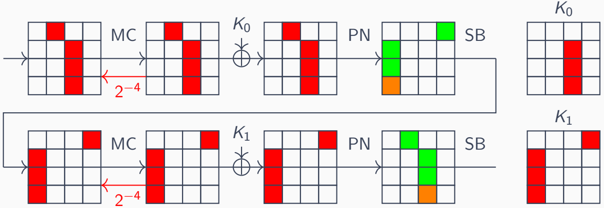
Differential Meet-in-the-Middle Attack on 23-Rounds [AKMMNP24]



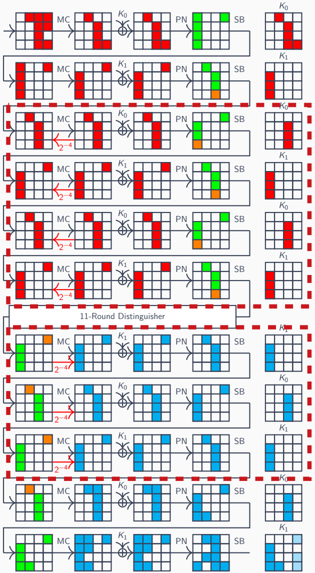
Deterministic Extension



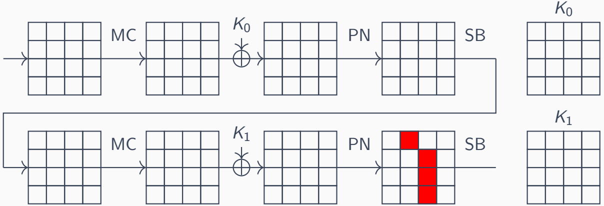
Probabilistic Key Recovery



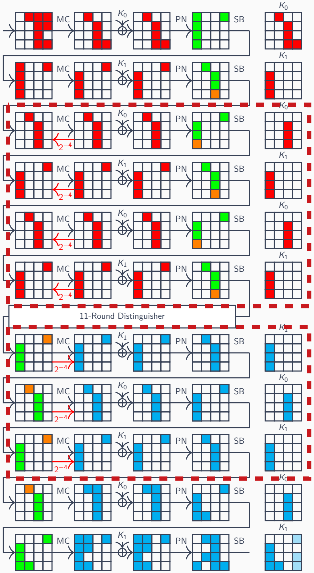
Differential Meet-in-the-Middle Attack on 23-Rounds [AKMMNP24]



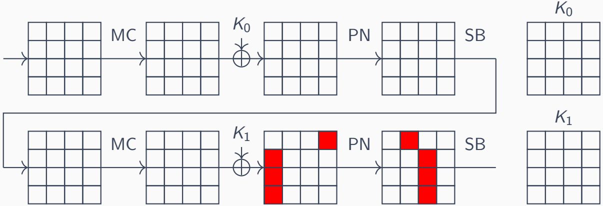
Probabilistic Key Recovery



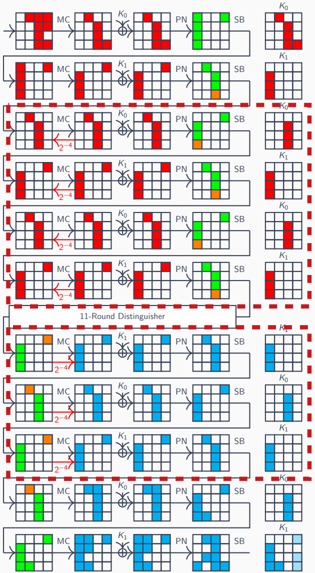
Differential Meet-in-the-Middle Attack on 23-Rounds [AKMMNP24]



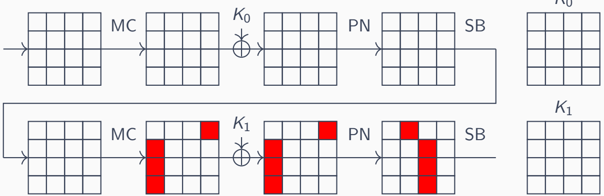
Probabilistic Key Recovery



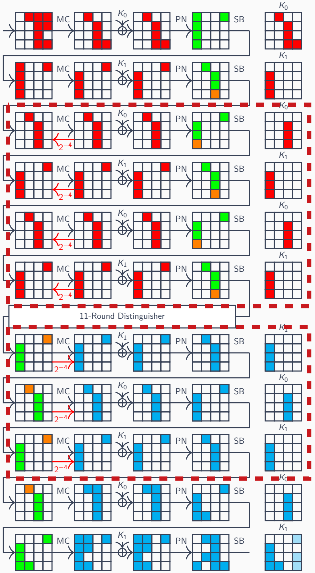
Differential Meet-in-the-Middle Attack on 23-Rounds [AKMMNP24]



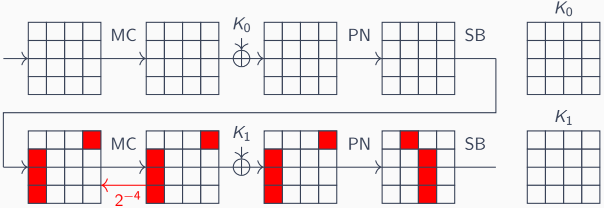
Probabilistic Key Recovery

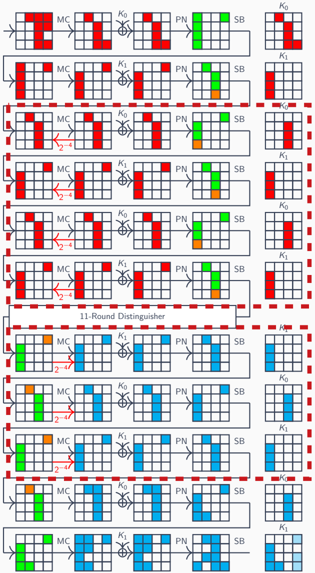


Differential Meet-in-the-Middle Attack on 23-Rounds [AKMMNP24]

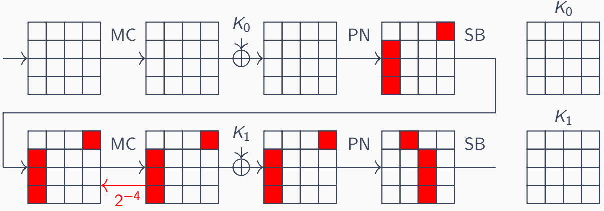


Probabilistic Key Recovery

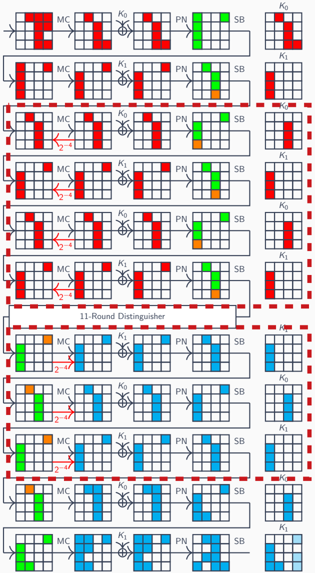




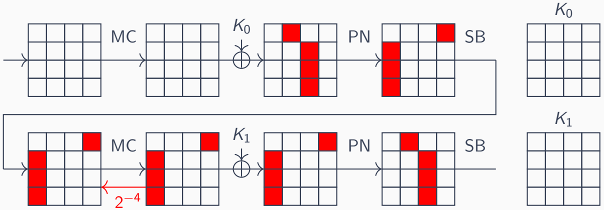
Probabilistic Key Recovery

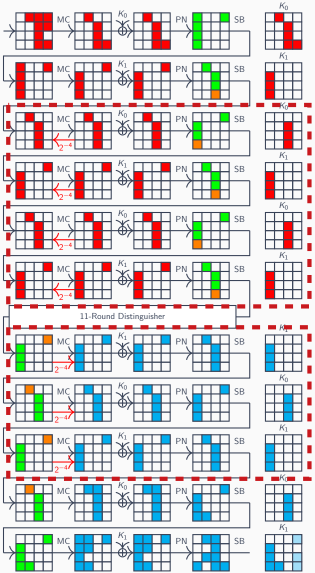


Differential Meet-in-the-Middle Attack on 23-Rounds [AKMMNP24]

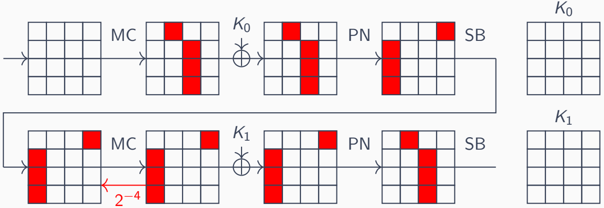


Probabilistic Key Recovery

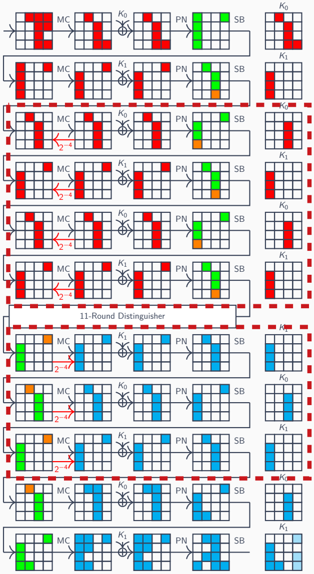




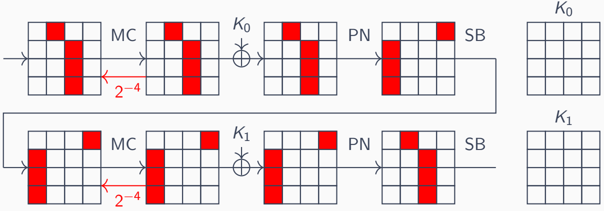
Probabilistic Key Recovery



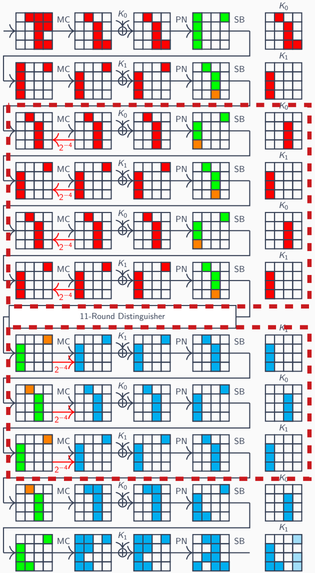
Differential Meet-in-the-Middle Attack on 23-Rounds [AKMMNP24]



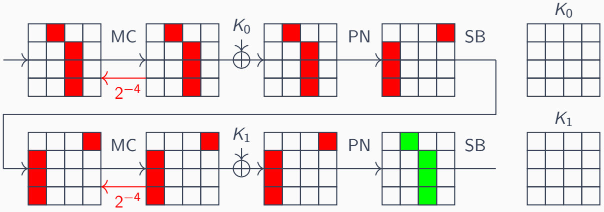
Probabilistic Key Recovery



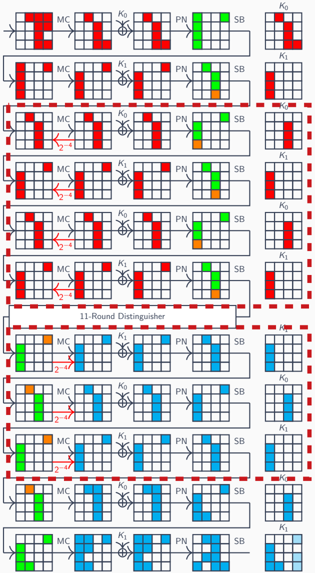
Differential Meet-in-the-Middle Attack on 23-Rounds [AKMMNP24]



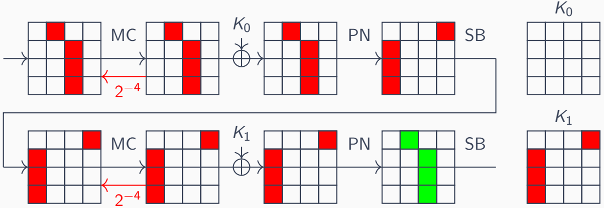
Probabilistic Key Recovery



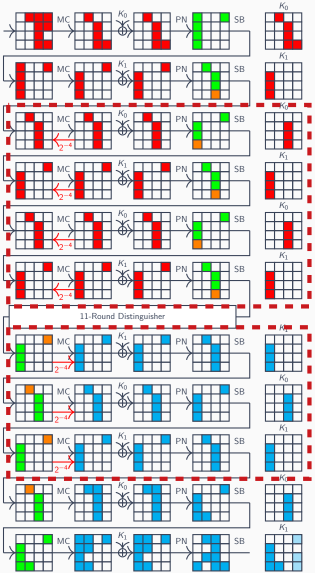
Differential Meet-in-the-Middle Attack on 23-Rounds [AKMMNP24]



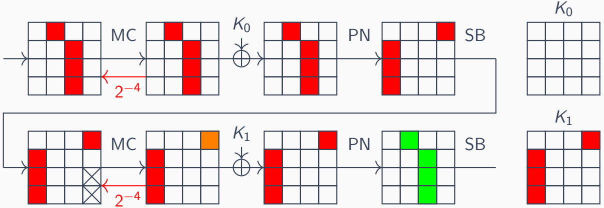
Probabilistic Key Recovery



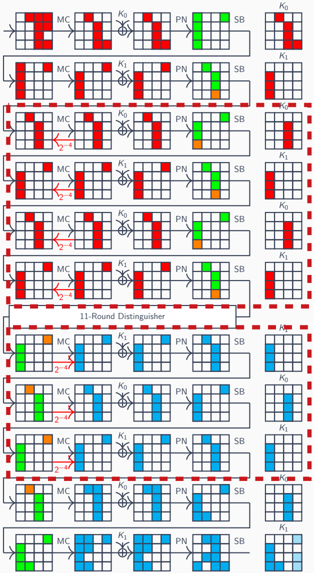
Differential Meet-in-the-Middle Attack on 23-Rounds [AKMMNP24]



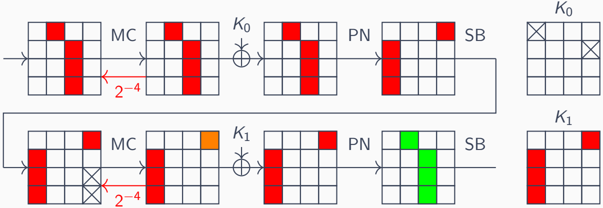
Probabilistic Key Recovery



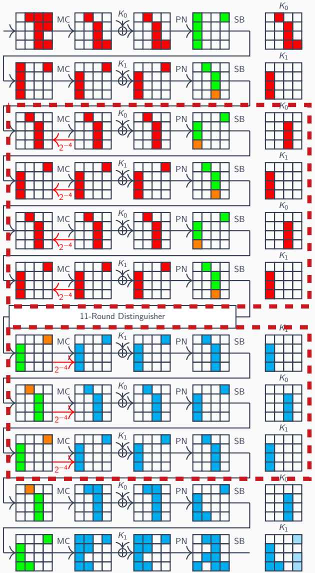
Differential Meet-in-the-Middle Attack on 23-Rounds [AKMMNP24]



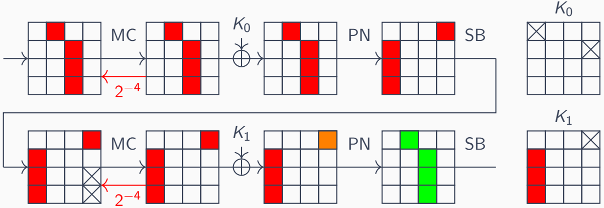
Probabilistic Key Recovery



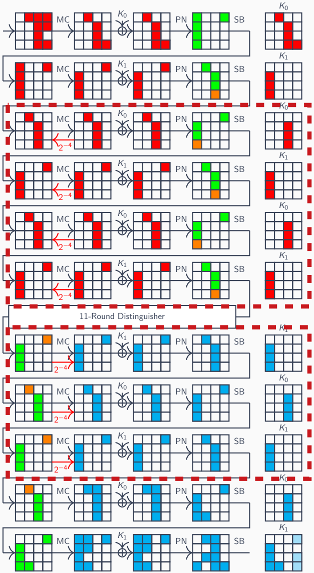
Differential Meet-in-the-Middle Attack on 23-Rounds [AKMMNP24]



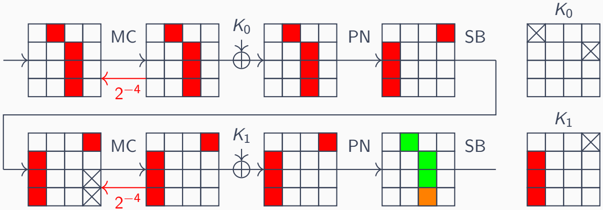
Probabilistic Key Recovery

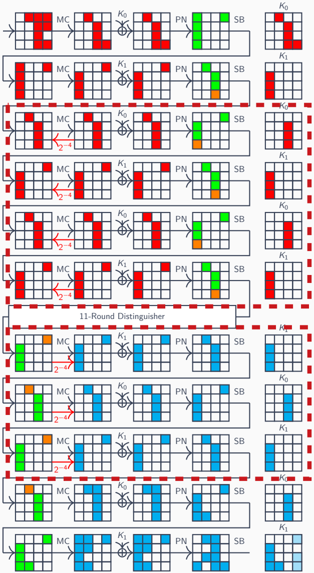


Differential Meet-in-the-Middle Attack on 23-Rounds [AKMMNP24]

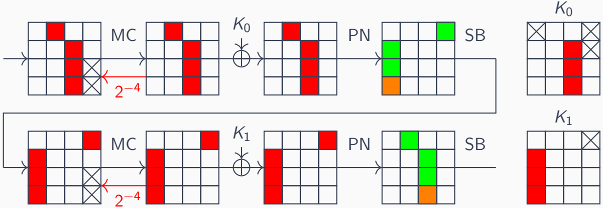


Probabilistic Key Recovery

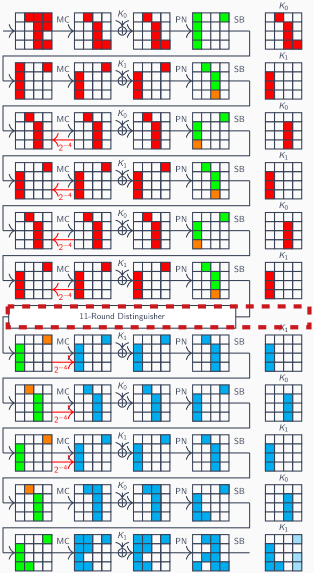




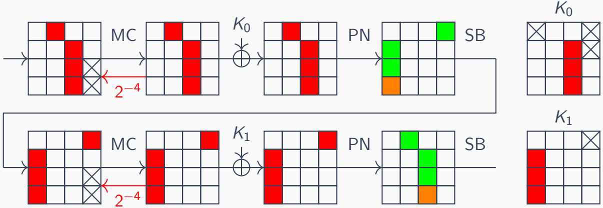
Probabilistic Key Recovery



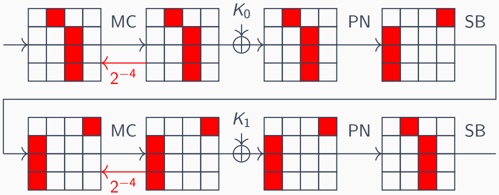
Differential Meet-in-the-Middle Attack on 23-Rounds [AKMMNP24]



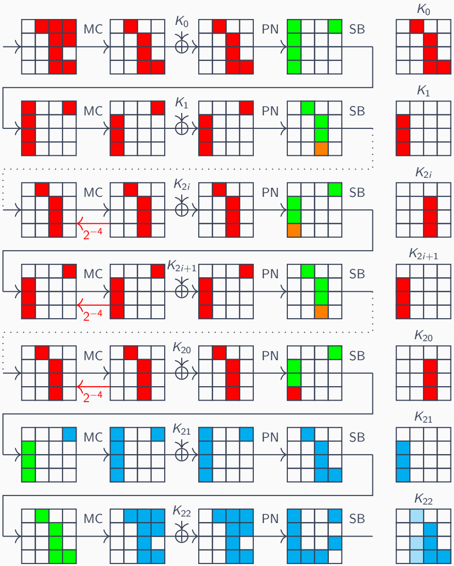
Probabilistic Key Recovery



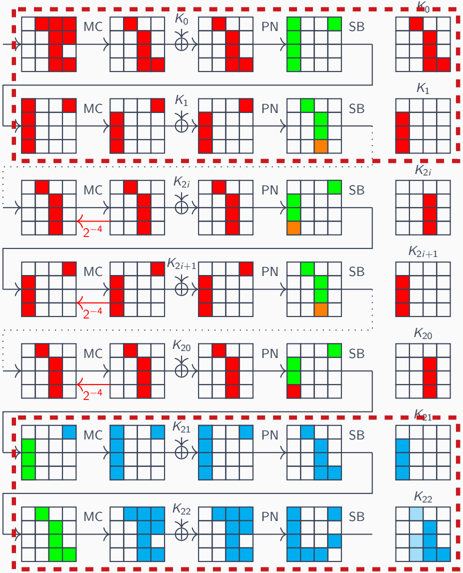
Truncated-Differential Characteristic



Our Attack

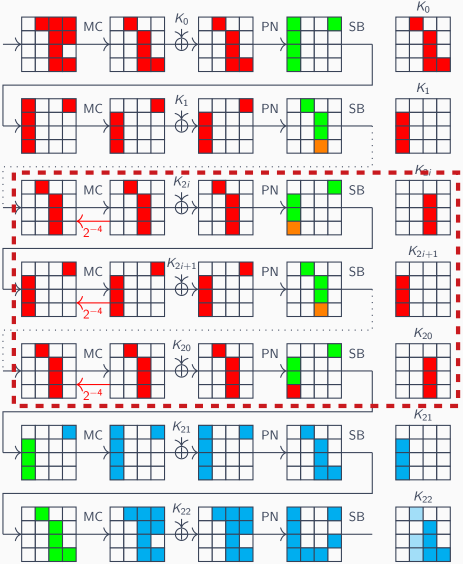


Our Attack



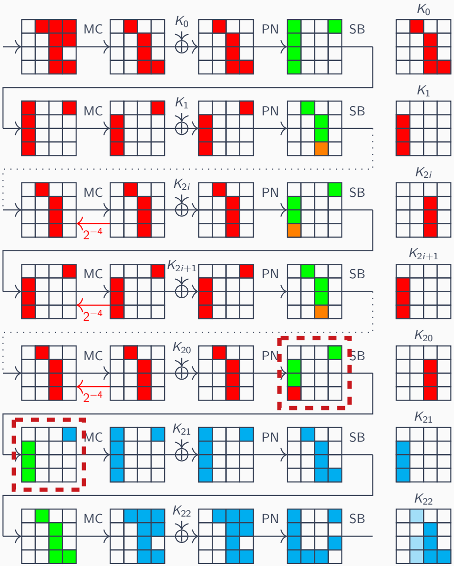
Deterministic Extension

Our Attack



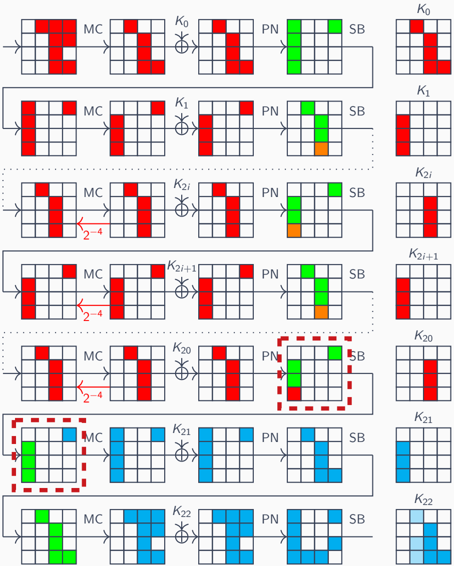
Probabilistic Key Recovery

Our Attack



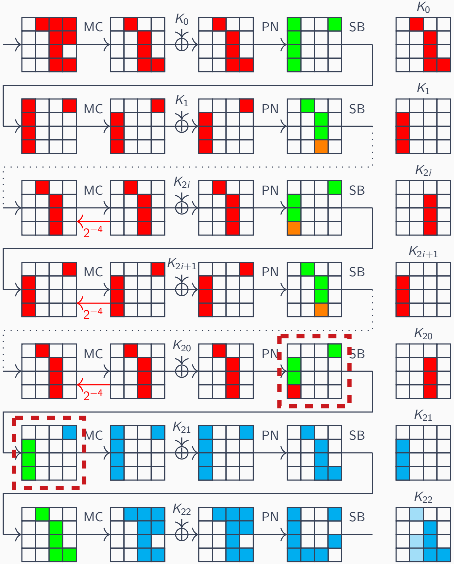
Zero-Round Distinguisher

Our Attack



Zero-Round Distinguisher

Compute same cells from both sides

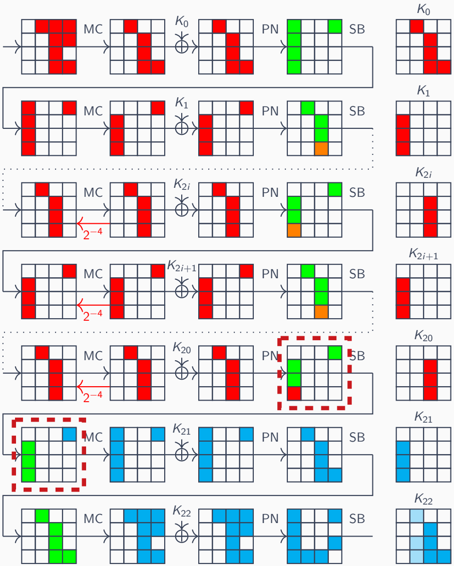


Zero-Round Distinguisher

Compute same cells from both sides

→ Use as additional filter

Our Attack



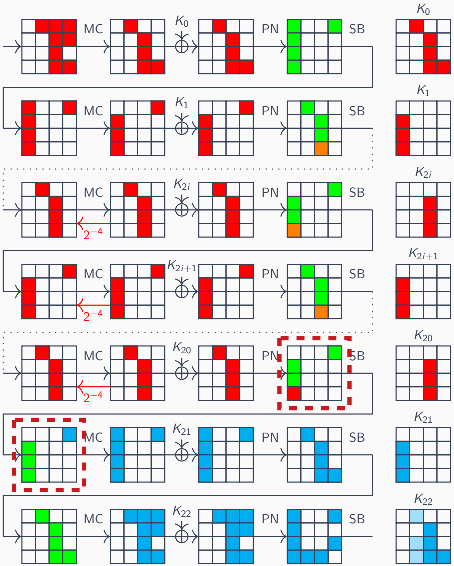
Zero-Round Distinguisher

Compute same cells from both sides

→ Use as additional filter

Two-Round Structure

Our Attack



Zero-Round Distinguisher

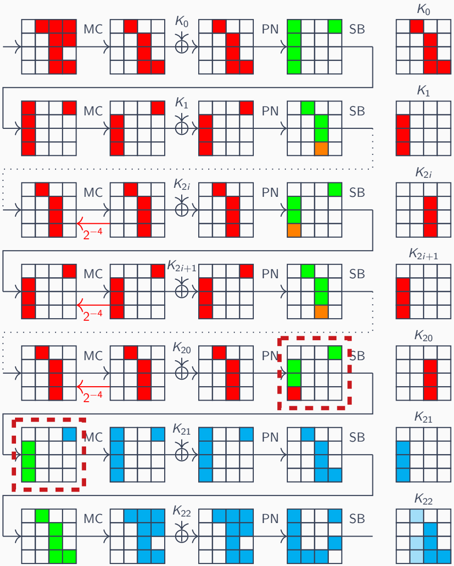
Compute same cells from both sides

→ Use as additional filter

Two-Round Structure

One-round structure in [AKMMNP24]

Our Attack



Zero-Round Distinguisher

Compute same cells from both sides

→ Use as additional filter

Two-Round Structure

One-round structure in [AKMMNP24]

Details in the paper

Comparison with Prior Work

Cipher	Rounds	Time	Memory	Data	Attack	Ref.
CRAFT	23	2^{125}	2^{101}	2^{60}	TD-MitM	[AKMMNP24]
		$2^{111.46}$	2^{120}	$2^{60.99}$	D	[SYCHW24]
		2^{109}	2^{36}	2^{58}	TD-MitM	This Work
	24	2^{110}	2^{34}	2^{60}	TD-MitM	This Work
	25	$2^{117.58}$	2^{48}	2^{60}	TD-MitM	This Work
	26	2^{118}	2^{34}	2^{64}	TD-MitM	This Work

D: Differential TD-MitM: Truncated Differential MitM

Comparison with Prior Work

Cipher	Rounds	Time	Memory	Data	Attack	Ref.
CRAFT	23	2^{125}	2^{101}	2^{60}	TD-MitM	[AKMMNP24]
		$2^{111.46}$	2^{120}	$2^{60.99}$	D	[SYCHW24]
		2^{109}	2^{36}	2^{58}	TD-MitM	This Work
	24	2^{110}	2^{34}	2^{60}	TD-MitM	This Work
	25	$2^{117.58}$	2^{48}	2^{60}	TD-MitM	This Work
	26	2^{118}	2^{34}	2^{64}	TD-MitM	This Work

D: Differential TD-MitM: Truncated Differential MitM

Thank you for your attention!