Speeding Up the Search Algorithm For the Best Differential and Best Linear Trail

Bao Zhenzhen Zhang Wentao Lin Dongdai

State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences

Outline

Outline

Why Searching For the Best Differential and Best Linear Trail?

- Differential cryptanalysis (DC) and linear cryptanalysis (LC)
  - Exploit good differentials or good linear approximations
- Judging the resistance of a given primitive to DC and LC
  - Establish an upper bound on the probability of the best differential or the bias of the best linear approximations
- Searching for the best trails to estimate the highest probability or highest bias

Why Searching For the Best Differential and Best Linear Trail?

- Differential cryptanalysis (DC) and linear cryptanalysis (LC)
  - Exploit good differentials or good linear approximations
- Judging the resistance of a given primitive to DC and LC
  - Establish an upper bound on the probability of the best differential or the bias of the best linear approximations
- Searching for the best trails to estimate the highest probability or highest bias

Why Searching For the Best Differential and Best Linear Trail?

- Differential cryptanalysis (DC) and linear cryptanalysis (LC)
  - Exploit good differentials or good linear approximations
- Judging the resistance of a given primitive to DC and LC
  - Establish an upper bound on the probability of the best differential or the bias of the best linear approximations
- Searching for the best trails to estimate the highest probability or highest bias

Why Searching For the Best Differential and Best Linear Trail?

- Searching for the optimal is an interesting, challenging and universal work
  - Running through all of the possibilities in the combinatorial universe: Great cardinality of the set of candidates

What Do We Already Have?

- Counting the minimal number of active S-Boxes to get the upper bound of the probabilities or bias of the best trails
  - Without the instantiated actual differences or without the knowldge of exact probabilities
  - Fail to find the best trail which does not have the minimal number of active S-Boxes: remains a gap
- Dijkstra's algorithm to find all best truncated differential trails and instantiate them with actual differences
  - Fail to find the best trail which does not have the minimal number of active S-Boxes
  - Breath-first approach, for ciphers using large number of small S-Boxes and have weak alignment , storage starvation

What Do We Already Have?

- Counting the minimal number of active S-Boxes to get the upper bound of the probabilities or bias of the best trails
    - Without the instantiated actual differences or without the knowldge of exact probabilities
    - Fail to find the best trail which does not have the minimal number of active S-Boxes: remains a gap
- Dijkstra's algorithm to find all best truncated differential trails and instantiate them with actual differences
    - Fail to find the best trail which does not have the minimal number of active S-Boxes
    - Breath-first approach, for ciphers using large number of small S-Boxes and have weak alignment , storage starvation

What Do We Already Have for Complete Search for the Best Trail?

- Original: Mitsuru Matsui. *On Correlation Between the Order of S-boxes and the Strength of DES*
  - A branch-and-bound depth-first search algorithm
  - Effectively find the best trails of DES, not fast enough for some other cryptosystems (eg. FEAL)
- Improved: Kazuo Ohta, Shiho Moriai, and Kazumaro Aoki. *Improving the Search Algorithm for the Best Linear Expression*
  - Search patterns: reduce unnecessary search candidates
  - New results on best linear approximations for FEAL
- Further improved: Kazumaro Aoki, Kunio Kobayashi, and Shiho Moriai . *Best Differential Characteristic Search of FEAL*
  - Using a pre-search strategy
  - Good results of the search for the best differential trails of FEAL

Why Do We Have to Speed up the Search?

- For designers:
  - Choose optimal components
  - Reduce the number of round to be sufficient and necessary and keep a good balance between security and efficiency
- For attackers:
  - To improve the success probabilities
  - To attack as long as possible

Why Do We Have to Speed up the Search?

- For designers:
  - Choose optimal components
  - Reduce the number of round to be sufficient and necessary and keep a good balance between security and efficiency
- For attackers:
  - To improve the success probabilities
  - To attack as long as possible

Outline

Iterated Block Ciphers

Attributes: Simple round $\underset{\hookrightarrow}{\textit{iterate}}$ Complex cryptosystem

Simple-Components XOR subkey, sboxes, linear permutations

$\Downarrow$ *composite*

Round-Function cryptographically weaker

$\Downarrow$ *iterating n times*

Iterated-Block-Cipher cryptographically strong

Iterated Block Ciphers

Attributes: Simple round $\underset{\hookrightarrow}{iterate}$ Complex cryptosystem

Simple-Components XOR subkey, sboxes, linear permutations

$\Downarrow$ *composite*

Round-Function cryptographically weaker

$\Downarrow$ *iterating n times*

Iterated-Block-Cipher cryptographically strong

Iterated Block Ciphers

Attributes: Simple round $\underset{\hookrightarrow}{\textit{iterate}}$ Complex cryptosystem

Simple-Components XOR subkey, sboxes, linear permutations

⇓ *composite*

Round-Function cryptographically weaker

⇓ *iterating n times*

Iterated-Block-Cipher cryptographically strong

Iterated Block Ciphers

Attributes: Simple round $\underset{\hookrightarrow}{\textit{iterate}}$ Complex cryptosystem

Simple-Components XOR subkey, sboxes, linear permutations

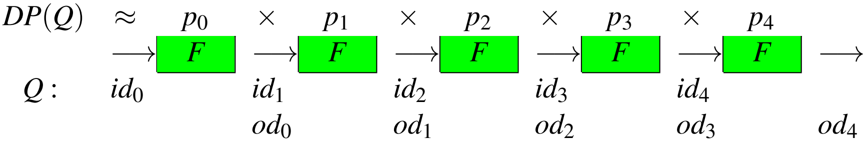
⇓ *composite*

Round-Function cryptographically weaker

⇓ *iterating n times*

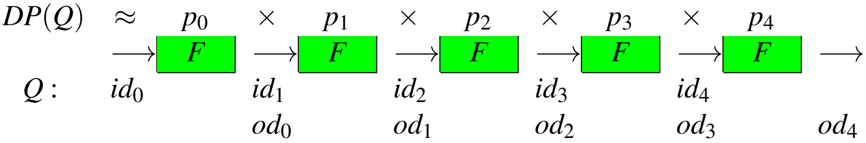
Iterated-Block-Cipher cryptographically strong

Differential Trails in Iterated Block Ciphers



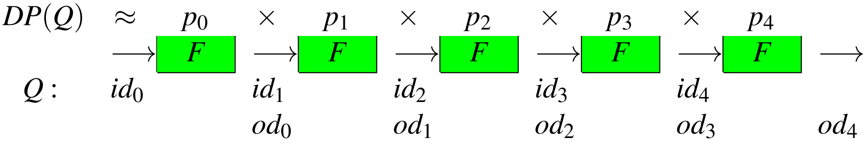
- Trail: sequence of differences
- $DP(Q) \approx \prod_{r=1}^{n} p_r$: fraction of pairs that exhibit differences in Q
- $w^n = \sum_{r=1}^{n} w_r$: a more natural way to characterize the power of trails
  - $w_r = -\log_2 p_r$

Differential Trails in Iterated Block Ciphers



- Trail: sequence of differences
- $DP(Q) \approx \prod_{r=1}^{n} p_r$: fraction of pairs that exhibit differences in Q
- $w^n = \sum_{r=1}^{n} w_r$: a more natural way to characterize the power of trails
  - $w_r = -\log_2 p_r$

Differential Trails in Iterated Block Ciphers



- Trail: sequence of differences
- $DP(Q) \approx \prod_{r=1}^{n} p_r$: fraction of pairs that exhibit differences in Q
- $w^n = \sum_{r=1}^{n} w_r$: a more natural way to characterize the power of trails
  - $w_r = -\log_2 p_r$

Differential Trails in Iterated Block Ciphers

Noekeon Round Founction

```
Round(Key, State ,Constant1 ,Constant2)
{
   State[0] ^= Constant1;
   Theta(Key, State);
   State[0] ^= Constant2;
   Pi1(State);
   Gamma(State);
   Pi2(State);
}
```

Differential Trails in Iterated Block Ciphers

Looking at Noekeon in a SP-Structure Way

$Theta(\cdot)$ $Pi_1(\cdot)$	$Pi_1(Theta(Pi_2(\cdot)))$	λ
$Gamma(\cdot)$	$Gamma(\cdot)$	γ
$Pi_2(\cdot)$ $Theta(\cdot)$ $Pi_1(\cdot)$	$Pi_1(Theta(Pi_2(\cdot)))$	λ
$Gamma(\cdot)$	$Gamma(\cdot)$	γ
$Pi_2(\cdot)$	$Pi_1(Theta(Pi_2(\cdot)))$	λ

Differential Trails in Iterated Block Ciphers

Differential Trails in Noekeon

r	p_r	w_r		$Trails(0x)$
0	2^{-17}	17	*pod*	0000000000000001000020210c280001
			sod	000000000000000c0000408e0182000c
1	2^{-8}	8	*pod*	000000000000000c0000008e0000000c
			sod	00000000000000010000002100000001
2	2^{-12}	12	*pod*	00000000004000000040080000400000
			sod	00000000002000000020040000300000
3	2^{-14}	14	*pod*	00000000002000400021040000300000
			sod	0000000000800020008c020000800000

000000000000000100002021 0c280001

$2^{-17} \downarrow \gamma$

000000000000000c0000408e0182000c

$\downarrow \lambda$

000000000000000c0000008e0000000c

$2^{-8} \downarrow \gamma$

00000000000000010000002100000001

$\downarrow \lambda$

00000000004000000040080000400000

$2^{-12} \downarrow \gamma$

00000000002000000020040000300000

Outline

Matsui's Algorithm

Algorithm property: Recursive & Search algorithm

Recursive Induction on the number of rounds n. i.e. derives the best n-round weight Bw^n from knowledge of the best r-round weight $Bw^r(1 \leq r \leq n-1)$

Search Depth-first & branch and bound search algorithm

Matsui's Algorithm

Algorithm property: Recursive & Search algorithm

Recursive Induction on the number of rounds n. i.e. derives the best n-round weight Bw^n from knowledge of the best r-round weight $Bw^r (1 \leq r \leq n-1)$

Search Depth-first & branch and bound search algorithm

Matsui's Algorithm

- Goal: Finding Bw^n, where:
  $w_1 + w_2 + \cdots + w_i + \cdots + w_n = Bw^n$
- An important fact:
  $w_{1_c} + w_{2_c} + \cdots + w_{i_c} + Bw^{n-i} \leq Bwc^n$, $\forall 1 \leq i \leq n-1$
  - $w_{1_c} + Bw^{n-1} \leq Bwc^n$
  - $w_{1_c} + w_{2_c} + Bw^{n-2} \leq Bwc^n$
  - $\vdots$
  - $w_{1_c} + w_{2_c} + \cdots + w_{i_c} + Bw^{n-i} \leq Bwc^n$
  - $\vdots$
  - $w_{1_c} + w_{2_c} + \cdots + w_{i_c} \cdots + w_{n-1_c} + Bw^1 \leq Bwc^n$
  - $w_{1_c} + w_{2_c} + \cdots + w_{i_c} \cdots + w_{n-1_c} + w_n \leq$

Matsui's Algorithm

- Goal: Finding Bw^n, where:
  $w_1 + w_2 + \cdots + w_i + \cdots + w_n = Bw^n$
- An important fact:
  $w_{1_c} + w_{2_c} + \cdots + w_{i_c} + Bw^{n-i} \leq Bwc^n, \forall 1 \leq i \leq n-1$
  - $w_{1_c} + Bw^{n-1} \leq Bwc^n$
  - $w_{1_c} + w_{2_c} + Bw^{n-2} \leq Bwc^n$
  - $\vdots$
  - $w_{1_c} + w_{2_c} + \cdots + w_{i_c} + Bw^{n-i} \leq Bwc^n$
  - $\vdots$
  - $w_{1_c} + w_{2_c} + \cdots + w_{i_c} \cdots + w_{n-1_c} + Bw^1 \leq Bwc^n$
  - $w_{1_c} + w_{2_c} + \cdots + w_{i_c} \cdots + w_{n-1_c} + w_n \leq$

Matsui's Algorithm

- Goal: Finding Bw^n, where:
  $w_1 + w_2 + \cdots + w_i + \cdots + w_n = Bw^n$
- An important fact:
  $w_{1_c} + w_{2_c} + \cdots + w_{i_c} + Bw^{n-i} \leq Bwc^n, \forall 1 \leq i \leq n-1$
  - $w_{1_c} + Bw^{n-1} \leq Bwc^n$
  - $w_{1_c} + w_{2_c} + Bw^{n-2} \leq Bwc^n$
  - $\vdots$
  - $w_{1_c} + w_{2_c} + \cdots + w_{i_c} + Bw^{n-i} \leq Bwc^n$
  - $\vdots$
  - $w_{1_c} + w_{2_c} + \cdots + w_{i_c} \cdots + w_{n-1_c} + Bw^1 \leq Bwc^n$
  - $w_{1_c} + w_{2_c} + \cdots + w_{i_c} \cdots + w_{n-1_c} + w_n \leq$

Matsui's Algorithm

- Goal: Finding Bw^n, where:
  $w_1 + w_2 + \cdots + w_i + \cdots + w_n = Bw^n$
- An important fact:
  $w_{1_c} + w_{2_c} + \cdots + w_{i_c} + Bw^{n-i} \leq Bwc^n, \forall 1 \leq i \leq n-1$
  - $w_{1_c} + Bw^{n-1} \leq Bwc^n$
  - $w_{1_c} + w_{2_c} + Bw^{n-2} \leq Bwc^n$
  - $\vdots$
  - $w_{1_c} + w_{2_c} + \cdots + w_{i_c} + Bw^{n-i} \leq Bwc^n$
  - $\vdots$
  - $w_{1_c} + w_{2_c} + \cdots + w_{i_c} \cdots + w_{n-1_c} + Bw^1 \leq Bwc^n$
  - $w_{1_c} + w_{2_c} + \cdots + w_{i_c} \cdots + w_{n-1_c} + w_n \leq$

Matsui's Algorithm

- Goal: Finding Bw^n, where:
  $w_1 + w_2 + \cdots + w_i + \cdots + w_n = Bw^n$
- An important fact:
  $w_{1_c} + w_{2_c} + \cdots + w_{i_c} + Bw^{n-i} \leq Bwc^n, \forall 1 \leq i \leq n-1$
  - $w_{1_c} + Bw^{n-1} \leq Bwc^n$
  - $w_{1_c} + w_{2_c} + Bw^{n-2} \leq Bwc^n$
  - $\vdots$
  - $w_{1_c} + w_{2_c} + \cdots + w_{i_c} + Bw^{n-i} \leq Bwc^n$
  - $\vdots$
  - $w_{1_c} + w_{2_c} + \cdots + w_{i_c} \cdots + w_{n-1_c} + Bw^1 \leq Bwc^n$
  - $w_{1_c} + w_{2_c} + \cdots + w_{i_c} \cdots + w_{n-1_c} + w_n \leq$

Matsui's Algorithm

- Goal: Finding Bw^n, where:
  $w_1 + w_2 + \cdots + w_i + \cdots + w_n = Bw^n$
- An important fact:
  $w_{1_c} + w_{2_c} + \cdots + w_{i_c} + Bw^{n-i} \leq Bwc^n, \forall 1 \leq i \leq n-1$
  - $w_{1_c} + Bw^{n-1} \leq Bwc^n$
  - $w_{1_c} + w_{2_c} + Bw^{n-2} \leq Bwc^n$
  - $\vdots$
  - $w_{1_c} + w_{2_c} + \cdots + w_{i_c} + Bw^{n-i} \leq Bwc^n$
  - $\vdots$
  - $w_{1_c} + w_{2_c} + \cdots + w_{i_c} \cdots + w_{n-1_c} + Bw^1 \leq Bwc^n$
  - $w_{1_c} + w_{2_c} + \cdots + w_{i_c} \cdots + w_{n-1_c} + w_n \leq$

Matsui's Algorithm

- Goal: Finding Bw^n, where:
  $w_1 + w_2 + \cdots + w_i + \cdots + w_n = Bw^n$
- An important fact:
  $w_{1_c} + w_{2_c} + \cdots + w_{i_c} + Bw^{n-i} \leq Bwc^n, \ \forall 1 \leq i \leq n-1$
  - $w_{1_c} + Bw^{n-1} \leq Bwc^n$
  - $w_{1_c} + w_{2_c} + Bw^{n-2} \leq Bwc^n$
  - $\vdots$
  - $w_{1_c} + w_{2_c} + \cdots + w_{i_c} + Bw^{n-i} \leq Bwc^n$
  - $\vdots$
  - $w_{1_c} + w_{2_c} + \cdots + w_{i_c} \cdots + w_{n-1_c} + Bw^1 \leq Bwc^n$
  - $w_{1_c} + w_{2_c} + \cdots + w_{i_c} \cdots + w_{n-1_c} + w_n \leq$

Matsui's Algorithm

- Goal: Finding Bw^n, where:
  $w_1 + w_2 + \cdots + w_i + \cdots + w_n = Bw^n$
- An important fact:
  $w_{1_c} + w_{2_c} + \cdots + w_{i_c} + Bw^{n-i} \leq Bwc^n, \forall 1 \leq i \leq n-1$
  - $w_{1_c} + Bw^{n-1} \leq Bwc^n$
  - $w_{1_c} + w_{2_c} + Bw^{n-2} \leq Bwc^n$
  - $\vdots$
  - $w_{1_c} + w_{2_c} + \cdots + w_{i_c} + Bw^{n-i} \leq Bwc^n$
  - $\vdots$
  - $w_{1_c} + w_{2_c} + \cdots + w_{i_c} \cdots + w_{n-1_c} + Bw^1 \leq Bwc^n$
  - $w_{1_c} + w_{2_c} + \cdots + w_{i_c} \cdots + w_{n-1_c} + w_n \leq$

Matsui's Algorithm

- Goal: Finding Bw^n, where:
  $w_1 + w_2 + \cdots + w_i + \cdots + w_n = Bw^n$
- An important fact:
  $w_{1_c} + w_{2_c} + \cdots + w_{i_c} + Bw^{n-i} \leq Bwc^n, \forall 1 \leq i \leq n-1$
  - $w_{1_c} + Bw^{n-1} \leq Bwc^n$
  - $w_{1_c} + w_{2_c} + Bw^{n-2} \leq Bwc^n$
  - $\vdots$
  - $w_{1_c} + w_{2_c} + \cdots + w_{i_c} + Bw^{n-i} \leq Bwc^n$
  - $\vdots$
  - $w_{1_c} + w_{2_c} + \cdots + w_{i_c} \cdots + w_{n-1_c} + Bw^1 \leq Bwc^n$
  - $w_{1_c} + w_{2_c} + \cdots + w_{i_c} \cdots + w_{n-1_c} + w_n \leq Bwc^n$

Matsui's Algorithm

- Goal: Finding Bw^n, where:
  $w_1 + w_2 + \cdots + w_i + \cdots + w_n = Bw^n$
- An important fact:
  $w_{1_c} + w_{2_c} + \cdots + w_{i_c} + Bw^{n-i} \leq Bwc^n, \forall 1 \leq i \leq n-1$
  - $w_{1_c} + Bw^{n-1} \leq Bwc^n$
  - $w_{1_c} + w_{2_c} + Bw^{n-2} \leq Bwc^n$
  - $\vdots$
  - $w_{1_c} + w_{2_c} + \cdots + w_{i_c} + Bw^{n-i} \leq Bwc^n$
  - $\vdots$
  - $w_{1_c} + w_{2_c} + \cdots + w_{i_c} \cdots + w_{n-1_c} + Bw^1 \leq Bwc^n$
  - $w_{1_c} + w_{2_c} + \cdots + w_{i_c} \cdots + w_{n-1_c} + w_n \leq Bwc^n$

Matsui's Algorithm

An Important Fact:

$$w_{1_c} + w_{2_c} + \cdots + w_{i_c} + Bw^{n-i} \leq Bwc^n, \ \forall 1 \leq i \leq n-1$$

$\leq Bwc^n$

w_{1_c}	w_{1_c}	$\cdots$	w_{1_c}	$\cdots$	w_{1_c}	w_{1_c}
Bw^{n-1}	w_{2_c}	$\cdots$	w_{2_c}	$\cdots$	w_{2_c}	w_{2_c}
	Bw^{n-2}	$\ddots$	$\vdots$	$\cdots$	$\vdots$	$\vdots$
		$\ddots$	w_{i_c}	$\vdots$	w_{i_c}	w_{i_c}
			Bw^{n-i}	$\ddots$	$\vdots$	$\vdots$
				$\ddots$	w_{n-1_c}	w_{n-1_c}
					Bw^{1}	w_{n_c}

Matsui's Algorithm

An Important Fact:

$$w_{1_c} + w_{2_c} + \cdots + w_{i_c} + Bw^{n-i} \leq Bwc^n, \ \forall 1 \leq i \leq n-1$$

<table>
<tr><td colspan="7">$\leq Bwc^n$</td></tr>
<tr><td>w_{1_c}</td><td>w_{1_c}</td><td>$\cdots$</td><td>w_{1_c}</td><td>$\cdots$</td><td>w_{1_c}</td><td>w_{1_c}</td></tr>
<tr><td rowspan="6">Bw^{n-1}</td><td>w_{2_c}</td><td>$\cdots$</td><td>w_{2_c}</td><td>$\cdots$</td><td>w_{2_c}</td><td>w_{2_c}</td></tr>
<tr><td rowspan="5">Bw^{n-2}</td><td>$\ddots$</td><td>$\vdots$</td><td>$\cdots$</td><td>$\vdots$</td><td>$\vdots$</td></tr>
<tr><td rowspan="4">$\ddots$</td><td>w_{i_c}</td><td>$\vdots$</td><td>w_{i_c}</td><td>w_{i_c}</td></tr>
<tr><td rowspan="3">Bw^{n-i}</td><td>$\ddots$</td><td>$\vdots$</td><td>$\vdots$</td></tr>
<tr><td rowspan="2">$\ddots$</td><td>w_{n-1_c}</td><td>w_{n-1_c}</td></tr>
<tr><td>Bw^1</td><td>w_{n_c}</td></tr>
</table>

Matsui's Algorithm

An Important Fact:

$$w_{1_c} + w_{2_c} + \cdots + w_{i_c} + Bw^{n-i} \leq Bwc^n, \ \forall 1 \leq i \leq n-1$$

$\leq Bwc^n$						
w_{1_c}	w_{1_c}	$\cdots$	w_{1_c}	$\cdots$	w_{1_c}	w_{1_c}
Bw^{n-1}	w_{2_c}	$\cdots$	w_{2_c}	$\cdots$	w_{2_c}	w_{2_c}
	Bw^{n-2}	$\ddots$	$\vdots$	$\cdots$	$\vdots$	$\vdots$
		$\ddots$	w_{i_c}	$\vdots$	w_{i_c}	w_{i_c}
			Bw^{n-i}	$\ddots$	$\vdots$	$\vdots$
				$\ddots$	w_{n-1_c}	w_{n-1_c}
					Bw^1	w_{n_c}

Matsui's Algorithm

An Important Fact:

$$w_{1_c} + w_{2_c} + \cdots + w_{i_c} + Bw^{n-i} \leq Bwc^n, \ \forall 1 \leq i \leq n-1$$

$\leq Bwc^n$						
w_{1_c}	w_{1_c}	$\cdots$	w_{1_c}	$\cdots$	w_{1_c}	w_{1_c}
Bw^{n-1}	w_{2_c}	$\cdots$	w_{2_c}	$\cdots$	w_{2_c}	w_{2_c}
	Bw^{n-2}	$\ddots$	$\vdots$	$\cdots$	$\vdots$	$\vdots$
		$\ddots$	w_{i_c}	$\vdots$	w_{i_c}	w_{i_c}
			Bw^{n-i}	$\ddots$	$\vdots$	$\vdots$
				$\ddots$	w_{n-1_c}	w_{n-1_c}
					Bw^1	w_{n_c}

Matsui's Algorithm

An Important Fact:

$$w_{1_c} + w_{2_c} + \cdots + w_{i_c} + Bw^{n-i} \leq Bwc^n, \ \forall 1 \leq i \leq n-1$$

$\leq Bwc^n$						
w_{1_c}	w_{1_c}	$\cdots$	w_{1_c}	$\cdots$	w_{1_c}	w_{1_c}
Bw^{n-1}	w_{2_c}	$\cdots$	w_{2_c}	$\cdots$	w_{2_c}	w_{2_c}
	Bw^{n-2}	$\ddots$	$\vdots$	$\cdots$	$\vdots$	$\vdots$
		$\ddots$	w_{i_c}	$\vdots$	w_{i_c}	w_{i_c}
			Bw^{n-i}	$\ddots$	$\vdots$	$\vdots$
				$\ddots$	w_{n-1_c}	w_{n-1_c}
					Bw^1	w_{n_c}

Outline

Moriai et al.'s Algorithm

Algorithm property: Use Search Pattern to Delete

- **Based-on** Matsui's branch-and-bound depth-first search
- **Introduced** Search patterns
  - **Delete** Duplicate candidates and nonexistent candidates

Moriai et al.'s Algorithm

Algorithm property: Use Search Pattern to Delete

- **Based-on** Matsui's branch-and-bound depth-first search
- **Introduced** Search patterns
- **Delete** Duplicate candidates and nonexistent candidates

Moriai et al.'s Algorithm

Algorithm property: Use Search Pattern to Delete

Based-on Matsui's branch-and-bound depth-first search

Introduced Search patterns

Delete Duplicate candidates and nonexistent candidates

Moriai et al.'s Algorithm

Definition 1. (Search Pattern) An n-round search pattern used in the search for the best differential trail is a vector of n values of weights, which is denoted as $\mathbb{W}^n = (w_1, w_2, \ldots, w_n)$, where w_i is the weight of the i-th round differential $(1 \leq i \leq n)$. Let $|\mathbb{W}^n| \equiv \sum_{i=1}^{n} w_i$.

Moriai et al.'s Algorithm

Search Pattern Examples: 3-round search patterns

⋯	Patterns for 28	Patterns for 29	Patterns for 30	⋯
⋮	⋮	⋮	⋮	⋮
⋮	(2, 14, 12)	(2, 14, 13)	(2, 14, 14)	⋮
⋮	(5, 14, 9)	(5, 14, 10)	(5, 14, 11)	⋮
⋮	(6, 11, 11)	(6, 11, 12)	(6, 11, 13)	⋮
⋮	(11, 6, 11)	(12, 6, 11)	(13, 6, 11)	⋮
⋮	(13, 7, 8)	(13, 7, 9)	(13, 7, 10)	⋮
⋮	⋮	⋮	⋮	⋮

Moriai et al.'s Algorithm

Problem: Duplicate Candidates

n-round differential whose difference of the i-th round F function (id_i, od_i) is exchanged for that of the $(n-i-1)$-th round F function (id_{n-i+1}, od_{n-i+1}) for all i $(1 \leq i \leq n)$ has the same meaning as the original one.

Solution: Deletion of Duplication Candidates

$C(w_1, w_2) \leq C(w_{n-1}, w_n)$

Moriai et al.'s Algorithm

Problem: Duplicate Candidates

n-round differential whose difference of the i-th round F function (id_i, od_i) is exchanged for that of the $(n-i-1)$-th round F function (id_{n-i+1}, od_{n-i+1}) for all i $(1 \leq i \leq n)$ has the same meaning as the original one.

Solution: Deletion of Duplication Candidates

$C(w_1, w_2) \leq C(w_{n-1}, w_n)$

Moriai et al.'s Algorithm

$$w(Q_1) \approx w_{0,1} + w_{1,2} + w_{2,3} + w_{3,4} + w_{4,5}$$

$$Q_1: \quad d_0 \xrightarrow{F} d_1 \xrightarrow{F} d_2 \xrightarrow{F} d_3 \xrightarrow{F} d_4 \xrightarrow{F} d_5$$

$$w(Q_1) \approx w_{5,4} + w_{4,3} + w_{3,2} + w_{2,1} + w_{1,0}$$

$$Q_1: \quad d_5 \xrightarrow{F^{-1}} d_4 \xrightarrow{F^{-1}} d_3 \xrightarrow{F^{-1}} d_2 \xrightarrow{F^{-1}} d_1 \xrightarrow{F^{-1}} d_0$$

$$w(Q_2) \approx w_{4,5} + w_{3,4} + w_{2,3} + w_{1,2} + w_{0,1}$$

$$Q_2: \quad d_5 \xrightarrow{F} d_4 \xrightarrow{F} d_3 \xrightarrow{F} d_2 \xrightarrow{F} d_1 \xrightarrow{F} d_0$$

Moriai et al.'s Algorithm

$$w(Q_1) \approx w_{0,1} + w_{1,2} + w_{2,3} + w_{3,4} + w_{4,5}$$

$$Q_1: \; d_0 \xrightarrow{} \boxed{F} \xrightarrow{} d_1 \xrightarrow{} \boxed{F} \xrightarrow{} d_2 \xrightarrow{} \boxed{F} \xrightarrow{} d_3 \xrightarrow{} \boxed{F} \xrightarrow{} d_4 \xrightarrow{} \boxed{F} \xrightarrow{} d_5$$

$$w(Q_2) \approx w_{4,5} + w_{3,4} + w_{2,3} + w_{1,2} + w_{0,1}$$

$$Q_2: \; d_5 \xrightarrow{} \boxed{F} \xrightarrow{} d_4 \xrightarrow{} \boxed{F} \xrightarrow{} d_3 \xrightarrow{} \boxed{F} \xrightarrow{} d_2 \xrightarrow{} \boxed{F} \xrightarrow{} d_1 \xrightarrow{} \boxed{F} \xrightarrow{} d_0$$

Solution: Deletion of Duplication Candidates

$$C(w_1, w_2) \leq C(w_{n-1}, w_n)$$

Moriai et al.'s Algorithm

$$w(Q_1) \approx w_{0,1} + w_{1,2} + w_{2,3} + w_{3,4} + w_{4,5}$$

$$Q_1 : \; d_0 \xrightarrow{F} d_1 \xrightarrow{F} d_2 \xrightarrow{F} d_3 \xrightarrow{F} d_4 \xrightarrow{F} d_5$$

$$w(Q_2) \approx w_{4,5} + w_{3,4} + w_{2,3} + w_{1,2} + w_{0,1}$$

$$Q_2 : \; d_5 \xrightarrow{F} d_4 \xrightarrow{F} d_3 \xrightarrow{F} d_2 \xrightarrow{F} d_1 \xrightarrow{F} d_0$$

Solution: Deletion of Duplication Candidates

$$C(w_1, w_2) \leq C(w_{n-1}, w_n)$$

Moriai et al.'s Algorithm

An Original Important Fact:

$$w_{1_c} + w_{2_c} + \cdots + w_{i_c} + Bw^{n-i} \leq Bwc^n, \ \forall 1 \leq i \leq n-1$$

- $w_{1_c} + Bw^{n-1} \leq Bwc^n$
- $w_{1_c} + w_{2_c} + Bw^{n-2} \leq Bwc^n$
- $\vdots$
- $w_{1_c} + w_{2_c} + \cdots + w_{i_c} + Bw^{n-i} \leq Bwc^n$
- $\vdots$
- $w_{1_c} + w_{2_c} + \cdots + w_{i_c} \cdots + w_{n-1_c} + Bw^1 \leq Bwc^n$
- $w_{1_c} + w_{2_c} + \cdots + w_{i_c} \cdots + w_{n-1_c} + w_n \leq Bwc^n$

Moriai et al.'s Algorithm

Another Important Fact:

$$w_{i-r+1_c} + w_{i-r+2_c} + \cdots + w_{i-1_c} + w_{i_c} \geq Bw^r, \; \forall 1 \leq r \leq i$$

- $w_{i_c} \geq Bw^1$
- $w_{i-1_c} + w_{i_c} \geq Bw^2$
- $\vdots$
- $w_{i-r+1_c} + w_{i-r+2_c} + \cdots + w_{i-1_c} + w_{i_c} \geq Bw^r$
- $\vdots$
- $w_{1_c} + w_{2_c} + \cdots + w_{i-j_c} + \cdots + w_{i-1_c} + w_{i_c} \geq Bw^i$

Moriai et al.'s Algorithm

An Original Important Fact:

$$w_{1_c} + w_{2_c} + \cdots + w_{i_c} + Bw^{n-i} \leq Bwc^n, \ \forall 1 \leq i \leq n-1$$

$\leq Bwc^n$					
$w_{1_c}+$	$w_{1_c}+$	$\cdots$	$w_{1_c}+$	$\cdots$	$w_{1_c}+$
Bw^{n-1}	$w_{2_c}+$	$\cdots$	$w_{2_c}+$	$\cdots$	$w_{2_c}+$
	Bw^{n-2}	$\ddots$	$\vdots$	$\cdots$	$\vdots$
			$w_{i_c}+$	$\vdots$	$w_{i_c}+$
		$\ddots$	Bw^{n-i}	$\ddots$	$\vdots$
				$\ddots$	$w_{n-1_c}+$
					Bw^1

Moriai et al.'s Algorithm

An Original Important Fact:

$$w_{1_c} + w_{2_c} + \cdots + w_{i_c} + Bw^{n-i} \leq Bwc^n, \ \forall 1 \leq i \leq n-1$$

$\leq Bwc^n$					
$w_{1_c}+$	$w_{1_c}+$	$\cdots$	$w_{1_c}+$	$\cdots$	$w_{1_c}+$
Bw^{n-1}	$w_{2_c}+$	$\cdots$	$w_{2_c}+$	$\cdots$	$w_{2_c}+$
	Bw^{n-2}	$\ddots$	$\vdots$	$\cdots$	$\vdots$
			$w_{i_c}+$	$\vdots$	$w_{i_c}+$
		$\ddots$	Bw^{n-i}	$\ddots$	$\vdots$
				$\ddots$	$w_{n-1_c}+$ Bw^1

Moriai et al.'s Algorithm

Another important facts:

$$w_{i-r+1_c} + w_{i-r+2_c} + \cdots + w_{i-1_c} + w_{i_c} \geq Bw^r, \ \forall 1 \leq r \leq i$$

$\geq Bw^1$	$\geq Bw^2$	$\geq Bw^3$	$\cdots$	$\geq Bw^r$	$\cdots$	$\geq Bw^i$
						$w_{1_c}+$
					$\iddots$	$\vdots$
				$w_{i-r+1_c}+$	$\vdots$	$w_{i-r+1_c}+$
			$\iddots$	$\vdots$	$\cdots$	$\vdots$
		$w_{i-2_c}+$	$\iddots$	$w_{i-2_c}+$	$\vdots$	$w_{i-2_c}+$
	$w_{i-1_c}+$	$w_{i-1_c}+$		$w_{i-1_c}+$	$\iddots$	$w_{i-1_c}+$
w_{i_c}	w_{i_c}	w_{i_c}		w_{i_c}	$\iddots$	w_{i_c}

Outline

Aoki et al.'s Algorithm

Algorithm property: Use Pre-Search to Delete

- Based-on: Matsui's and Moriai et al.'s
- Using: Pre-Search Procedure
- Delete: Reduced-Round Nonexistent Candidates

Aoki et al.'s Algorithm

Algorithm property: Use Pre-Search to Delete

Based-on Matsui's and Moriai et al.'s

Using Pre-Search Procedure

Delete Reduced-Round Nonexistent Candidates

Aoki et al.'s Algorithm

Algorithm property: Use Pre-Search to Delete

- **Based-on** Matsui's and Moriai et al.'s
- **Using** Pre-Search Procedure
- **Delete** Reduced-Round Nonexistent Candidates

Aoki et al.'s Algorithm

Algorithm property: Use Pre-Search to Delete

- Knowledge of r-round patterns as well as all r-round Bw^r are used more sufficiently
- Knowledge on search patterns with higher weights as well as with the best weight are used to detecting the impossible

Aoki et al.'s Algorithm

Algorithm property: Use Pre-Search to Delete

- Knowledge of r-round patterns as well as all r-round Bw^r are used more sufficiently
- Knowledge on search patterns with higher weights as well as with the best weight are used to detecting the impossible

Aoki et al.'s Algorithm

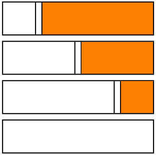
Pre-Search Existent Pattern Examples

1 for 28	1 for 29	2 for 30	3 for 31	5 for 32	8 for 33
$(11,6,11)$	$(12,6,11)$	$(13,6,11)$	$(15,6,10)$	$(14,4,14)$	$(15,3,15)$
		$(12,6,12)$	$(14,6,11)$	$(16,6,10)$	$(4,5,24)$
			$(13,6,12)$	$(15,6,11)$	$(15,4,14)$
				$(14,6,12)$	$(17,6,10)$
				$(13,6,13)$	$(16,6,11)$
					$(15,6,12)$
					$(14,6,13)$
					$(13,9,11)$

Previous Work Summry

- Algorithm 1: $w_{1_c} + w_{2_c} + \cdots + w_{i_c} + Bw^{n-i} \leq Bwc^n$
  - $\Longrightarrow w_{i_c} \leq$ *an upper bound*
  -

Previous Work Summry

- Algorithm 1: $w_{1_c} + w_{2_c} + \cdots + w_{i_c} + Bw^{n-i} \leq Bwc^n$
  - $\Longrightarrow w_{i_c} \leq$ *an upper bound*
  - 

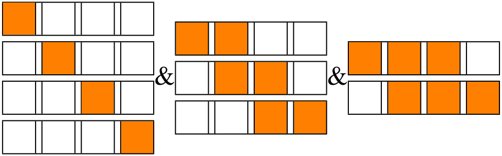
Previous Work Summry

- Algorithm 1: $\Longrightarrow w_{i_c} \leq$ *an upper bound*
- Algorithm 2:
  $w_{i-r+1_c} + w_{i-r+2_c} + \cdots + w_{i-1_c} + w_{i_c} \geq Bw^r, \forall 1 \leq r \leq i$
  - $\Longrightarrow w_{i_c} \geq$ *a lower bound*
  -

Previous Work Summry

- Algorithm 1: $\Longrightarrow w_{i_c} \leq$ *an upper bound*
- Algorithm 2:
  $w_{i-r+1_c} + w_{i-r+2_c} + \cdots + w_{i-1_c} + w_{i_c} \geq Bw^r, \ \forall 1 \leq r \leq i$
  - $\Longrightarrow w_{i_c} \geq$ *a lower bound*

Previous Work Summry

- Algorithm 1: $\Longrightarrow w_{i_c} \leq$ *an upper bound*
- Algorithm 2:
  $w_{i-r+1_c} + w_{i-r+2_c} + \cdots + w_{i-1_c} + w_{i_c} \geq Bw^r, \forall 1 \leq r \leq i$
  - $\Longrightarrow w_{i_c} \geq$ *a lower bound*
  - 

Previous Work Summry

- Algorithm 1: $\Longrightarrow w_{i_c} \leq$ *an upper bound*
- Algorithm 2: $\Longrightarrow w_{i_c} \geq$ *a lower bound*
- Algorithm 3: $(w_{i-r+1_c}, w_{i-r+2_c}, \cdots, w_{i-1_c}, w_{i_c}) \in \{r - \textit{round exist pattern}\}_{\forall 1 \leq r \leq i}$
  - $\Longrightarrow w_{i_c} \in$ *some values in* $[\textit{a lower bound}, \textit{an upper bound}]$

Previous Work Summry

- Algorithm 1: $\Longrightarrow w_{i_c} \leq$ *an upper bound*
- Algorithm 2: $\Longrightarrow w_{i_c} \geq$ *a lower bound*
- Algorithm 3: $(w_{i-r+1_c}, w_{i-r+2_c}, \cdots, w_{i-1_c}, w_{i_c}) \in \{r - \text{round exist pattern}\}_{\forall 1 \leq r \leq i}$
  - $\Longrightarrow w_{i_c} \in$ *some values in* [*a lower bound*, *an upper bound*]

Previous Work Summry

- Algorithm 1: $\Longrightarrow w_{i_c} \leq$ *an upper bound*
- Algorithm 2: $\Longrightarrow w_{i_c} \geq$ *a lower bound*
- Algorithm 3: $(w_{i-r+1_c}, w_{i-r+2_c}, \cdots, w_{i-1_c}, w_{i_c}) \in \{r - \textit{round exist pattern}\}_{\forall 1 \leq r \leq i}$
  - $\Longrightarrow w_{i_c} \in$ *some values in* $[\textit{a lower bound}, \textit{an upper bound}]$

Previous Work Summry

- Algorithm 1: $\Longrightarrow w_{i_c} \leq$ *an upper bound*
- Algorithm 2: $\Longrightarrow w_{i_c} \geq$ *a lower bound*
- Algorithm 3: $\Longrightarrow w_{i_c} \in$ *some values in [a lower bound, an upper bound]*

Previous Work Summry

- Algorithm 1: $\Longrightarrow w_{i_c} \leq$ *an upper bound*
- Algorithm 2: $\Longrightarrow w_{i_c} \geq$ *a lower bound*
- Algorithm 3:
$\Longrightarrow w_{i_c} \in$ *some values in* $[$*a lower bound*, *an upper bound*$]$

Previous Work Summry

- Algorithm 1: $\Longrightarrow w_{i_c} \leq$ *an upper bound*
- Algorithm 2: $\Longrightarrow w_{i_c} \geq$ *a lower bound*
- Algorithm 3:
  $\Longrightarrow w_{i_c} \in$ *some values in* $[$*a lower bound*, *an upper bound*$]$

Basic Strategy and Basic Principle

Speeding Up the Search Algorithm Using Optimized Strategies

Based-on Matsui's, Moriai et al.'s and Aoki et al.'s Algorithm

Using Branch-and-bound depth-first, Search patterns, Pre-Search

Search In an organized way following optimization strategies

Basic Strategy and Basic Principle

Speeding Up the Search Algorithm Using Optimized Strategies

Based-on Matsui's, Moriai et al.'s and Aoki et al.'s Algorithm

Using Branch-and-bound depth-first, Search patterns, Pre-Search

Search In an organized way following optimization strategies

Basic Strategy and Basic Principle

Speeding Up the Search Algorithm Using Optimized Strategies

Based-on Matsui's, Moriai et al.'s and Aoki et al.'s Algorithm

Using Branch-and-bound depth-first, Search patterns, Pre-Search

Search In an organized way following optimization strategies

Basic Strategy and Basic Principle

For the short round cipher: eg. n-round

$w^n = w\_u^n$	Collect patterns supported by an n-round trail	
$w^n = w^n + 1$	Collect patterns supported by an n-round trail	
$\vdots$	Collect patterns supported by an n-round trail	
$w^n = w^n + 1$	$\leftarrow$find out an n-round trail?	Yes, $Bw^n = w^n$
$\vdots$	$\leftarrow$find out an n-round trail?	No
$w^n = w^n + 1$	$\leftarrow$find out an n-round trail?	No
$w^n = w\_l^n$	$\leftarrow$find out an n-round trail?	No

Basic Strategy and Basic Principle

For the short round cipher: eg. n-round

w^n =w_u^n	Collect patterns supported by an n-round trail	
$w^n = w^n + 1$	Collect patterns supported by an n-round trail	
⋮	Collect patterns supported by an n-round trail	
$w^n = w^n + 1$	←find out an n-round trail?	Yes, $Bw^n = w^n$
⋮	←find out an n-round trail?	No
$w^n = w^n + 1$	←find out an n-round trail?	No
w^n =w_l^n	←find out an n-round trail?	No

Basic Strategy and Basic Principle

For the short round cipher: eg. n-round

$w^n = w\_u^n$	Collect patterns supported by an n-round trail	
$w^n = w^n + 1$	Collect patterns supported by an n-round trail	
$\vdots$	Collect patterns supported by an n-round trail	
$w^n = w^n + 1$	$\leftarrow$find out an n-round trail?	Yes, $Bw^n = w^n$
$\vdots$	$\leftarrow$find out an n-round trail?	No
$w^n = w^n + 1$	$\leftarrow$find out an n-round trail?	No
$w^n = w\_l^n$	$\leftarrow$find out an n-round trail?	No

Basic Strategy and Basic Principle

For the short round cipher: eg. n-round

w^n =w_u^n	Collect patterns supported by an n-round trail	
$w^n = w^n + 1$	Collect patterns supported by an n-round trail	
⋮	Collect patterns supported by an n-round trail	
$w^n = w^n + 1$	←find out an n-round trail?	Yes, $Bw^n = w^n$
⋮	←find out an n-round trail?	No
$w^n = w^n + 1$	←find out an n-round trail?	No
w^n =w_l^n	←find out an n-round trail?	No

Basic Strategy and Basic Principle

For the short round cipher: eg. n-round

w^n =w_u^n	Collect patterns supported by an n-round trail	
$w^n = w^n + 1$	Collect patterns supported by an n-round trail	
$\vdots$	Collect patterns supported by an n-round trail	
$w^n = w^n + 1$	←find out an n-round trail?	Yes, $Bw^n = w^n$
$\vdots$	←find out an n-round trail?	No
$w^n = w^n + 1$	←find out an n-round trail?	No
w^n =w_l^n	←find out an n-round trail?	No

Basic Strategy and Basic Principle

For the short round cipher: eg. n-round

$w^n = w\_u^n$	Collect patterns supported by an n-round trail	
$w^n = w^n + 1$	Collect patterns supported by an n-round trail	
$\vdots$	Collect patterns supported by an n-round trail	
$w^n = w^n + 1$	$\leftarrow$find out an n-round trail?	Yes, $Bw^n = w^n$
$\vdots$	$\leftarrow$find out an n-round trail?	No
$w^n = w^n + 1$	$\leftarrow$find out an n-round trail?	No
$w^n = w\_l^n$	$\leftarrow$find out an n-round trail?	No

Basic Strategy and Basic Principle

For the short round cipher: eg. n-round

$w^n = w\_u^n$	Collect patterns supported by an n-round trail	
$w^n = w^n + 1$	Collect patterns supported by an n-round trail	
$\vdots$	Collect patterns supported by an n-round trail	
$w^n = w^n + 1$	$\leftarrow$find out an n-round trail?	Yes, $Bw^n = w^n$
$\vdots$	$\leftarrow$find out an n-round trail?	No
$w^n = w^n + 1$	$\leftarrow$find out an n-round trail?	No
$w^n = w\_l^n$	$\leftarrow$find out an n-round trail?	No

Basic Strategy and Basic Principle

For the short round cipher: eg. n-round

$w^n = w\_u^n$	Collect patterns supported by an n-round trail	
$w^n = w^n + 1$	Collect patterns supported by an n-round trail	
$\vdots$	Collect patterns supported by an n-round trail	
$w^n = w^n + 1$	$\leftarrow$find out an n-round trail?	Yes, $Bw^n = w^n$
$\vdots$	$\leftarrow$find out an n-round trail?	No
$w^n = w^n + 1$	$\leftarrow$find out an n-round trail?	No
$w^n = w\_l^n$	$\leftarrow$find out an n-round trail?	No

Basic Strategy and Basic Principle

For the short round cipher: eg. n-round

$w^n = w\_u^n$	Collect patterns supported by an n-round trail	
$w^n = w^n + 1$	Collect patterns supported by an n-round trail	
$\vdots$	Collect patterns supported by an n-round trail	
$w^n = w^n + 1$	$\leftarrow$find out an n-round trail?	Yes, $Bw^n = w^n$
$\vdots$	$\leftarrow$find out an n-round trail?	No
$w^n = w^n + 1$	$\leftarrow$find out an n-round trail?	No
$w^n = w\_l^n$	$\leftarrow$find out an n-round trail?	No

Basic Strategy and Basic Principle

For the short round cipher: eg. n-round

$w^n = w\_u^n$	Collect patterns supported by an n-round trail	
$w^n = w^n + 1$	Collect patterns supported by an n-round trail	
$\vdots$	Collect patterns supported by an n-round trail	
$w^n = w^n + 1$	←find out an n-round trail?	Yes, $Bw^n = w^n$
$\vdots$	←find out an n-round trail?	No
$w^n = w^n + 1$	←find out an n-round trail?	No
$w^n = w\_l^n$	←find out an n-round trail?	No

Basic Strategy and Basic Principle

For the short round cipher: eg. n-round

$w^n = w\_u^n$	Collect patterns supported by an n-round trail	
$w^n = w^n + 1$	Collect patterns supported by an n-round trail	
$\vdots$	Collect patterns supported by an n-round trail	
$w^n = w^n + 1$	$\leftarrow$find out an n-round trail?	Yes, $Bw^n = w^n$
$\vdots$	$\leftarrow$find out an n-round trail?	No
$w^n = w^n + 1$	$\leftarrow$find out an n-round trail?	No
$w^n = w\_l^n$	$\leftarrow$find out an n-round trail?	No

Strategies to Make Optimization

- Starting from the narrowest point
- Concretizing and grouping search patterns
- Trialling in minimal changes order

Strategies to Make Optimization

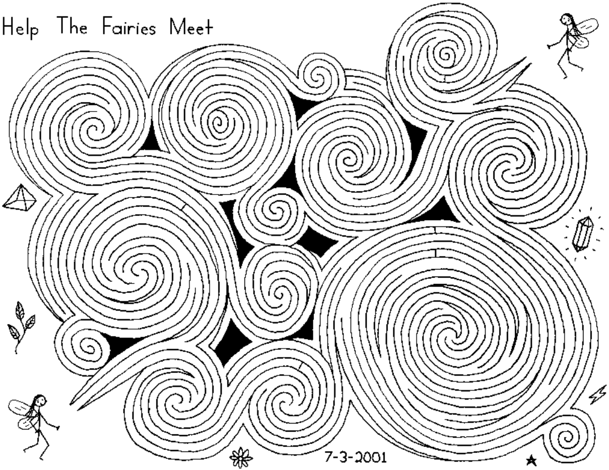
- Starting from the narrowest point
- Concretizing and grouping search patterns
- Trialling in minimal changes order

Strategies to Make Optimization

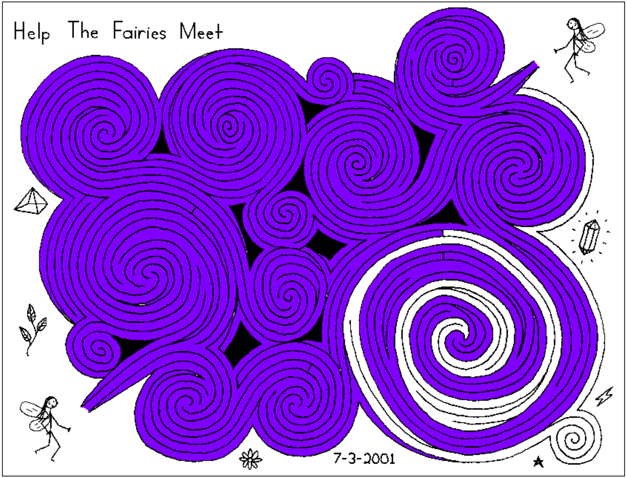
- Starting from the narrowest point
- Concretizing and grouping search patterns
- Trialling in minimal changes order

Outline

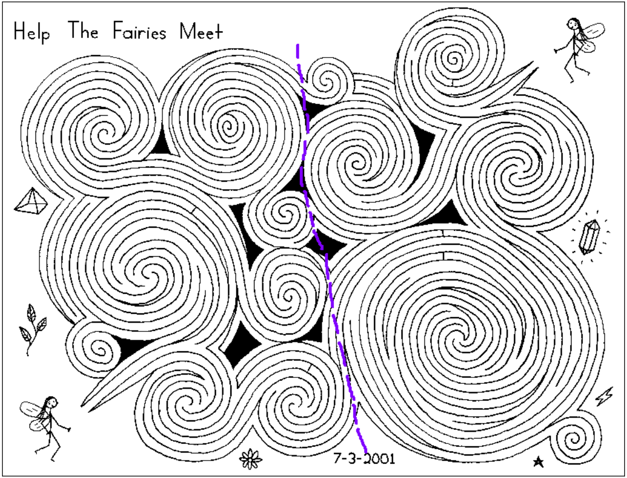
Start From the Narrowest Point - A Metaphor



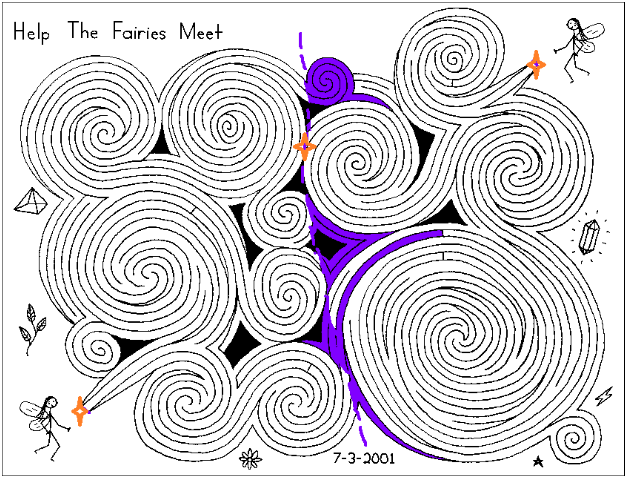
Start From the Narrowest Point - A Metaphor



Start From the Narrowest Point - A Metaphor



Start From the Narrowest Point - A Metaphor



Start From the Narrowest Point - Proposal

Definition 2. (Narrowest Point and Relative-Index Form)

Given $\mathbb{W}^n = (w_1, w_2, \ldots, w_n)$

Suppose $w_{x_i} = w_{min} \equiv \min(w_1, w_2, \ldots, w_n)$ *for* $1 \leq i \leq k$

Let $\mathrm{nxt}(x) \equiv \begin{cases} x+1 & 1 \leq x \leq n-1 \\ n-1 & x = n \end{cases}$,

$v_{min} \equiv \min(w_{\mathrm{nxt}(x_1)}, w_{\mathrm{nxt}(x_2)}, \ldots, w_{\mathrm{nxt}(x_k)})$

Suppose $w_m = w_{min}$, $w_{\mathrm{nxt}(m)} = v_{min}$, we call m the **narrowest point**.

Rewrite $\mathbb{W}^n$ as $\breve{\mathbb{W}}^n = (\breve{w}_{-m+1}, \ldots, \breve{w}_{-1}, \breve{w}_0, \breve{w}_1, \ldots, \breve{w}_{n-m})$, where $\breve{w}_{x-m} = w_x$. We call $\breve{\mathbb{W}}^n$ the **relative-index form** of $\mathbb{W}^n$ and define the relative index of w_x as $\mathrm{rix}(w_x) = \mathrm{rix}(\breve{w}_{x-m}) = x - m$, $1 \leq x \leq n$.

Start From the Narrowest Point - Proposal

A search pattern $\mathbb{W}^n$ is placed at the search tree in its relative-index form $\breve{\mathbb{W}}^n$ as

$$\breve{w}_0$$
$$|$$
$$\breve{w}_1$$
$$\vdots$$
$$\breve{w}_{n-m}$$
$$|$$
$$\breve{w}_{-1}$$
$$\vdots$$
$$\breve{w}_{-m+1} \quad .$$

Figure: Placing a search pattern at the search tree in its relative-index form

Start From the Narrowest Point - Proposal

Example

Considering a set of search patterns with $|\mathbb{W}^3| = 30$ of 3-round NOEKEON:
$\mathbb{S} = \{$ (2, 14, 14), (14, 2, 14), (4, 13, 13), (13, 4, 13), (6, 11, 13), (13, 6, 11), (6, 12, 12), (12, 6, 12) $\}$.

Start From the Narrowest Point - Proposal

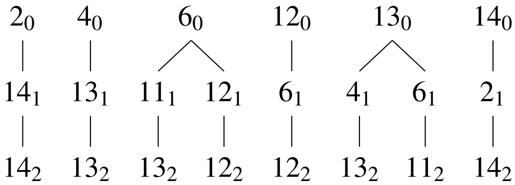


Figure: Organizing from the first points (index components relative to the round-index of the first component)

Start From the Narrowest Point - Proposal

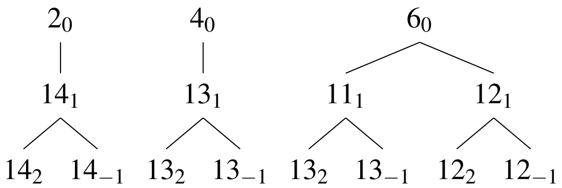


Figure: Organizing from the narrowest points (index components relative to the round-index of the narrowest point)

Start From the Narrowest Point - Justification

- More nodes and longer prefix-paths can be shared $\Rightarrow$ avoid repeated work
- The smaller the weight, the less the number of candidate round differentials $\Rightarrow$ avoid working in vain
- Restriction are more stringent, backtracking in invalid path could arise early $\Rightarrow$ avoid working in vain

Start From the Narrowest Point - Justification

- More nodes and longer prefix-paths can be shared $\Rightarrow$ avoid repeated work
- The smaller the weight, the less the number of candidate round differentials $\Rightarrow$ avoid working in vain
- Restriction are more stringent, backtracking in invalid path could arise early $\Rightarrow$ avoid working in vain

Start From the Narrowest Point - Justification

- More nodes and longer prefix-paths can be shared $\Rightarrow$ avoid repeated work
- The smaller the weight, the less the number of candidate round differentials $\Rightarrow$ avoid working in vain
- Restriction are more stringent, backtracking in invalid path could arise early $\Rightarrow$ avoid working in vain

Start From the Narrowest Point - Justification

- More nodes and longer prefix-paths can be shared $\Rightarrow$ avoid repeated work

Example: the range of the minimal value in a search pattern set is narrower than the range of an arbitrary value

- Partition 11 to 4 positive numbers x_1, x_2, x_3, x_4:
  - the smallest number must equal to 1 or 2
  - x_1 can be any number between 1 and 8.
- For the search pattern set with $|\mathbb{W}^3| = 30$ of 3-round NOEKEON
  - Set of values at the narrowest point is {2, 4, 6},
  - Set of values at the first point is {2, 4, 6, 12, 13, 14}.

Start From the Narrowest Point - Justification

- More nodes and longer prefix-paths can be shared ⇒ avoid repeated work
  - There are only 7 nodes at the first two layers of the latter structure, while 14 nodes at that of the former.
  - More patterns can share search prefix-pathes in the latter structure than in the former.

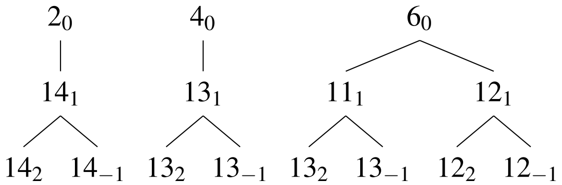


Figure: Organizing from the narrowest points

Start From the Narrowest Point - Justification

- More nodes and longer prefix-paths can be shared $\Rightarrow$ avoid repeated work
  - There are only 7 nodes at the first two layers of the latter structure, while 14 nodes at that of the former.
  - More patterns can share search prefix-pathes in the latter structure than in the former.

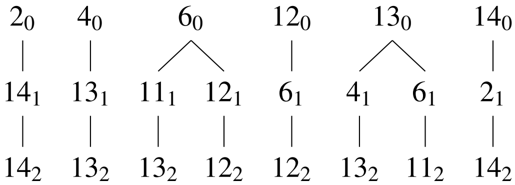


Figure: Organizing from the first points

Start From the Narrowest Point - Justification

- The smaller the weight, the less the number of candidate round differentials ⇒ avoid working in vain

Table: Numbers of candidates for one-round differential under various weight

w^1	2	3	4	5	6	7	8	9
$CN \approx$	$2^{4.58}$	$2^{6.17}$	$2^{18.12}$	$2^{20.71}$	$2^{26.08}$	$2^{29.20}$	$2^{33.68}$	$2^{37.08}$

Start From the Narrowest Point - Justification

- Restriction are more stringent, backtracking in invalid path could arise early $\Rightarrow$ avoid working in vain
    - If the preceding round is with one active S-Box, there are 12 trails propagating through the succeeding round, if the preceding round is with two active S-Boxes, there are 981 trails.
    - Number of differential pairs with weight 3 is much more than that with weight 2 for active S-Boxes.

Start From the Narrowest Point - Experiment

Table: Experimental results comparison between search *starting from the first point* (abbr. as "First" or "F"), search *starting from the narrowest point* (abbr. as "Narrowest" or "N")

w^3	Time(mins)		Ratio
	First	Narrowest	F/N
28	7.43	0.11	67.55
29	8.01	0.98	8.17
30	375.60	1.10	341.45
31	375.88	1.11	338.63
32	2398.50	15.99	150.00

Outline

Concretizing and Grouping Search Patterns - Proposal

Concretizing: For a search pattern

$\mathbb{W}^n = (w_1, \ldots, w_m, \ldots, w_n)$ and its relative-index form $\breve{\mathbb{W}}^n = (\breve{w}_{-m+1}, \ldots, \breve{w}_0, \ldots, \breve{w}_{n-m})$, its **concretized search patterns** are

$$\{\breve{W}^n\} = \{(\breve{w}_{-m+1}, \ldots, \binom{[asn]}{\breve{w}_0}, \ldots, \breve{w}_{n-m}) | asn \in [asn\_min, asn\_max]\}$$

where $[asn\_min, asn\_max]$ is the range of possible ASN of round-differential at the narrowest round with round-weight $\breve{w}_0$.

Concretizing and Grouping Search Patterns - Proposal

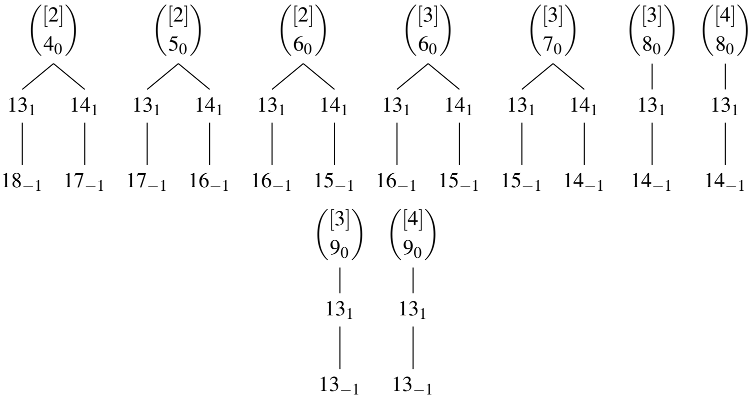


Figure: Concretizing search patterns by appending information of possible number of active S-Boxes at the narrowest point

Concretizing and Grouping Search Patterns - Proposal

Grouping: For two search patterns having same possible ASN at the narrowest point:

- $\mathbb{W}_1^n = (w_{1,1}, \ldots, w_{1,m_1}, \ldots, w_{1,n})$ and its relative-index form $\breve{\mathbb{W}}_1^n = (\breve{w}_{1,-m_1+1}, \ldots, \breve{w}_{1,0}, \ldots, \breve{w}_{1,n-m_1})$, and one of its concretized pattern
$$\breve{W}_1^n = \left(\breve{w}_{1,-m_1+1}, \ldots, \begin{pmatrix} [asn] \\ \breve{w}_{1,0} \end{pmatrix}, \ldots, \breve{w}_{1,n-m_1}\right)$$
- $\mathbb{W}_2^n = (w_{2,1}, \ldots, w_{2,m_2}, \ldots, w_{2,n})$ and its relative-index form $\breve{\mathbb{W}}_2^n = (\breve{w}_{2,-m_2+1}, \ldots, \breve{w}_{2,0}, \ldots, \breve{w}_{2,n-m_2})$, and one of its concretized pattern
$$\breve{W}_2^n = \left(\breve{w}_{2,-m_2+1}, \ldots, \begin{pmatrix} [asn] \\ \breve{w}_{2,0} \end{pmatrix}, \ldots, \breve{w}_{2,n-m_2}\right)$$

Concretizing and Grouping Search Patterns - Proposal

Grouping: For two search patterns having same possible ASN at the narrowest point:

$$\begin{pmatrix} [asn] \\ \{\breve{w}_{1,0}, \breve{w}_{2,0}\}_0 \end{pmatrix}$$

$$v_i^{\{\breve{w}_{1,0}, \breve{w}_{2,0}\}}$$

$$\vdots \qquad\qquad \vdots$$

We group them as $\breve{w}_{1,-m_1+1}^{\{\breve{w}_{1,0}\}} \qquad \breve{w}_{2,-m_2+1}^{\{\breve{w}_{2,0}\}}$.

Concretizing and Grouping Search Patterns - Proposal

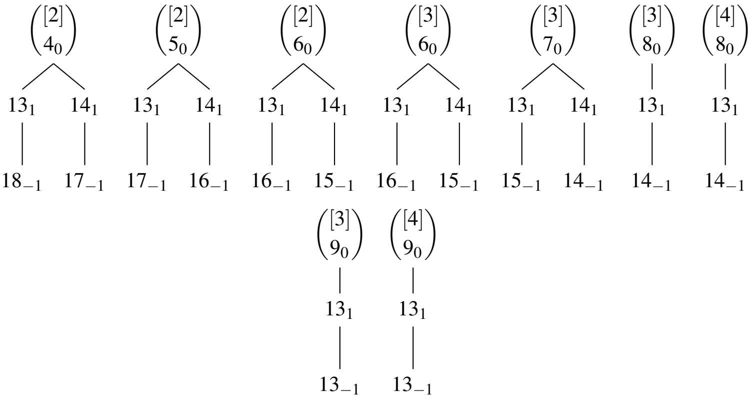


Figure: Concretizing search patterns by appending information of possible number of active S-Boxes at the narrowest point

Concretizing and Grouping Search Patterns - Proposal

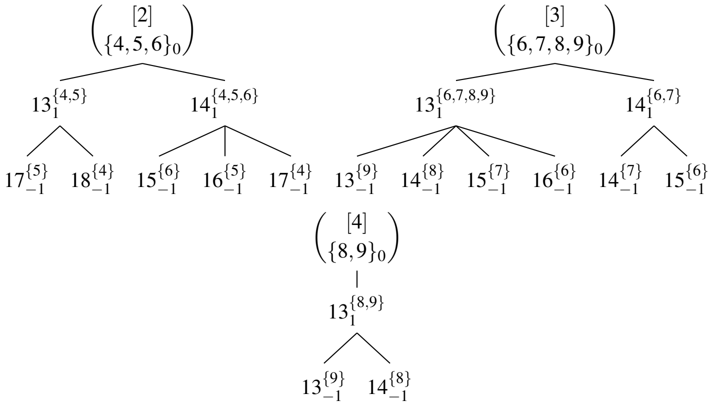


Figure: Grouping search patterns by number of active S-Boxes at the narrowest point

Concretizing and Grouping Search Patterns - Justification

- More specified knowledge of the search patterns will be learned during the pre-search phase.
- Searches can share the forward propagation prefixes among different search patterns with various narrowest point weight

Concretizing and Grouping Search Patterns - Justification

- More specified knowledge of the search patterns will be learned during the pre-search phase.
- Searches can share the forward propagation prefixes among different search patterns with various narrowest point weight

Concretizing and Grouping Search Patterns - Justification

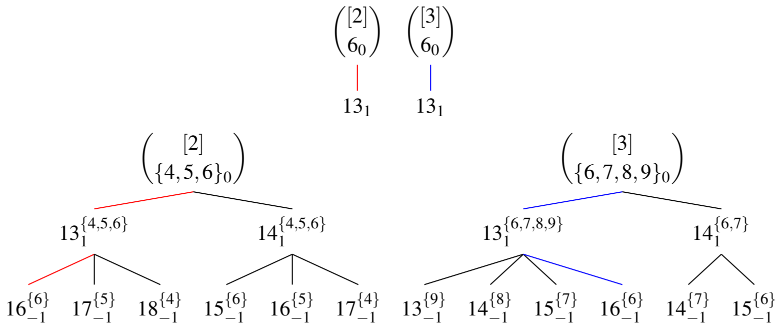


Figure: Knowledge of ASN at the narrowest point will be learned during the pre-search phase

Concretizing and Grouping Search Patterns - Justification

- Searches can share the forward propagation prefixes among different search patterns with various narrowest point weight

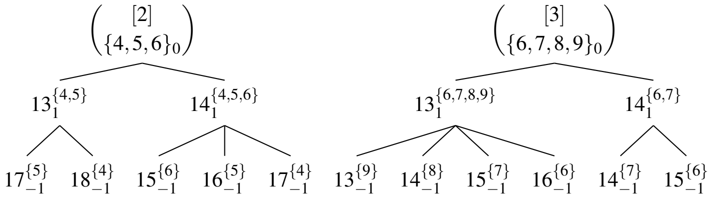


Figure: Grouping search patterns by number of active S-Boxes at the narrowest point

Concretizing and Grouping Search Patterns - Experiment

w^3	Time(mins)			Ratio		
	First	Narrowest	Concretize	F/N	N/C	F/C
28	7.43	0.11	0.01	67.55	11.00	743.00
29	8.01	0.98	0.01	8.17	98.00	801.00
30	375.60	1.10	0.44	341.45	2.50	853.64
31	375.88	1.11	0.45	338.63	2.47	835.29
32	2398.50	15.99	0.85	150.00	18.81	2821.76
33	-	16.65	0.91	-		-
34	-	16.77	1.08	-		-
35	-	165.54	1.56	-		-
36	-	172.82	30.97	-		-
37	-	177.73	33.70	-		-

Outline

Trailling In Minimal Changes Order

- Avoid the full execution of the P-Layer considering that there is locality of individual S-Box within S-Layer and linearity of P-Layer, which makes local calculation feasible
- Trialling in minimal changes order to minimize the number of local calculation, thus to minimize the cost of generating round differentials

Trailling In Minimal Changes Order

- Avoid the full execution of the P-Layer considering that there is locality of individual S-Box within S-Layer and linearity of P-Layer, which makes local calculation feasible
- Trialling in minimal changes order to minimize the number of local calculation, thus to minimize the cost of generating round differentials

Trailling In Minimal Changes Order

Original round difference pair

$0000\cdots4\cdots00004008000040000 0$

$\downarrow\gamma$

$0000\cdots2\cdots000020040000300000$

$\downarrow\lambda$

$0000\cdots2\cdots400021040000300000$

In SP-Table $nibble(id,od)=(4,3)$:

$0000\cdots0\cdots000000000000400000$

$\downarrow\gamma$

$0000\cdots0\cdots000000000000300000$

$\downarrow\lambda$

$8201\cdots0\cdots408201004082310040$

Generate new round difference pair

$0000\cdots4\cdots000040080000400000$

$\downarrow\gamma$

$0000\cdots2\cdots000020040000200000$

$\downarrow\lambda$

?

In SP-Table $nibble(id,od)=(4,2)$:

$0000\cdots0\cdots000000000000400000$

$\downarrow\gamma$

$0000\cdots0\cdots000000000000200000$

$\downarrow\lambda$

$0001\cdots0\cdots400001004000210040$

Trailling In Minimal Changes Order

Original round difference pair

$0000\cdots\mathbf{4}\cdots0000\mathbf{4}00\mathbf{8}0000\mathbf{4}00000$

$\downarrow\gamma$

$0000\cdots\mathbf{2}\cdots0000\mathbf{2}00\mathbf{4}0000\mathbf{3}00000$

$\downarrow\lambda$

$0000\cdots\mathbf{2}\cdots\mathbf{4}000\mathbf{21}0\mathbf{4}0000\mathbf{3}00000$

In SP-Table $nibble(id, od) = (4,3)$:

$0000\cdots0\cdots000000000000\mathbf{4}00000$

$\downarrow\gamma$

$0000\cdots0\cdots000000000000\mathbf{3}00000$

$\downarrow\lambda$

$\mathbf{82}0\mathbf{1}\cdots0\cdots\mathbf{4}0\mathbf{821}00\mathbf{4}0\mathbf{8231}00\mathbf{4}0$

Generate new round difference pair

$0000\cdots\mathbf{4}\cdots0000\mathbf{4}00\mathbf{8}0000\mathbf{4}00000$

$\downarrow\gamma$

$0000\cdots\mathbf{2}\cdots0000\mathbf{2}00\mathbf{4}0000\mathbf{2}00000$

$\downarrow\lambda$

?

In SP-Table $nibble(id, od) = (4,2)$:

$0000\cdots0\cdots000000000000\mathbf{4}00000$

$\downarrow\gamma$

$0000\cdots0\cdots000000000000\mathbf{2}00000$

$\downarrow\lambda$

$000\mathbf{1}\cdots0\cdots\mathbf{4}0000\mathbf{1}00\mathbf{4}000\mathbf{21}00\mathbf{4}0$

Trailling In Minimal Changes Order

Original one-round difference pair⇒Generate new one-round difference pair

$0000\cdots\mathbf{4}\cdots0000\mathbf{4}00\mathbf{8}0000\mathbf{4}00000$

$\downarrow\gamma$

$0000\cdots\mathbf{2}\cdots0000\mathbf{2}00\mathbf{4}0000\mathbf{3}00000$

$\downarrow\lambda$

$0000\cdots\mathbf{2}\cdots\mathbf{4}000\mathbf{21}0\mathbf{4}0000\mathbf{3}00000$

Trailling In Minimal Changes Order

Original one-round difference pair⇒Generate new one-round difference pair

$0000\cdots4\cdots00004008000400000$

$\downarrow\gamma$

$0000\cdots2\cdots00002004000300000\oplus$

$0000\cdots0\cdots00000000000300000\oplus$

$0000\cdots0\cdots00000000000200000$

$\downarrow\lambda$

$0000\cdots2\cdots40002104000300000\oplus$

$8201\cdots0\cdots40820100408231004 0\oplus$

$0001\cdots0\cdots40000100400021004 0$

Trailling In Minimal Changes Order

Original one-round difference pair⇒Generate new one-round difference pair

0000···**4**···0000**4**00**8**0000**4**00000	0000···**4**···0000**4**00**8**0000**4**00000
$\downarrow \gamma$	$\downarrow \gamma$
0000···**2**···0000**2**00**4**0000**3**00000	0000···**2**···0000**2**00**4**0000**2**00000
$\downarrow \lambda$	$\downarrow \lambda$
0000···**2**···**4**000**21**0**4**0000**3**00000	**82**00···**2**···**4**0**8221**0**4**00**82**200000

Trailling In Minimal Changes Order

- Achieve the minimal changes: Candidate round differentials are characterized by
  - The weight patterns of active S-Boxes
  - Indices of their 4-bit candidate differences within each active S-Box

Trailling In Minimal Changes Order

- Achieve the minimal changes: Candidate round differentials are characterized by
  - The weight patterns of active S-Boxes
  - Indices of their 4-bit candidate differences within each active S-Box

Trailling In Minimal Changes Order

- Enumerate the weight patterns of active S-Boxes

Enumerate by the Elements exchange code

- Take one-round differential weight as 10 and number of active S-Boxes as 4 for example.
- Weight patterns of active S-Boxes are (3322), (2323), (2332), (3223), (3232) and (2233).

Trailling In Minimal Changes Order

- Achieve the minimal changes by enumerate the weight patterns by the elements exchange code order

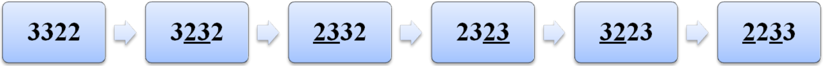


Figure 5.5 Example of elements exchange code order

G. Ehrlich, "Loopless algorithms for generating permutations, combinations, and other combinatorial configurations," *J. ACM* **20**, pp. 500–513, July 1973.

Trailling In Minimal Changes Order - Preliminaries

- Enumerate the indices of 4-bit candidate differences within each active S-Box

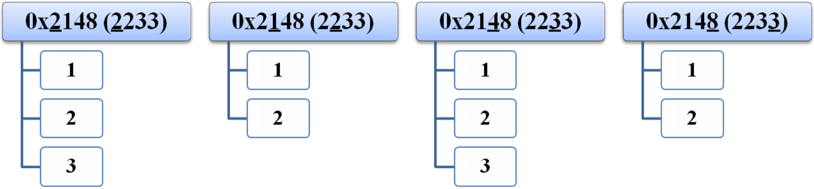


Figure 5.6 An example to be try in Gray code order

Trailling In Minimal Changes Order - Preliminaries

- Achieve the minimal changes by enumerate the indices by the Gray code order

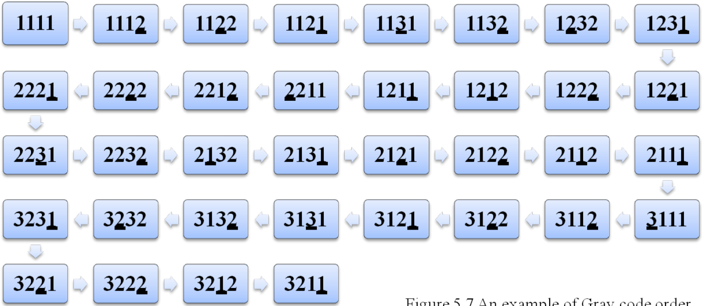


Figure 5.7 An example of Gray code order

D. E. Knuth, *The Art of Computer Programming. Volume 4, Fascicle 0. Introduction to Combinatorial Algo-*

Trailling In Minimal Changes Order - Some Metaphors

- Small Change Effect
  - Generating the new from the old might bemuch cheaper than generate fromnothing if the changes are subtle
- Large Scale Effect
  - Doing things in large scale can be more economical and efficient
  - Exhaustive searches in real life may be more powerful than they seem

Trailling In Minimal Changes Order - Some Metaphors

- Small Change Effect
  - Generating the new from the old might bemuch cheaper than generate fromnothing if the changes are subtle
- Large Scale Effect
  - Doing things in large scale can be more economical and efficient
  - Exhaustive searches in real life may be more powerful than they seem

Trailling In Minimal Changes Order - Some Metaphors

- Small Change Effect
  - Generating the new from the old might bemuch cheaper than generate fromnothing if the changes are subtle
- Large Scale Effect
  - Doing things in large scale can be more economical and efficient
  - Exhaustive searches in real life may be more powerful than they seem

Trailling In Minimal Changes Order - Some Metaphors

- Small Change Effect
  - Generating the new from the old might bemuch cheaper than generate fromnothing if the changes are subtle
- Large Scale Effect
  - Doing things in large scale can be more economical and efficient
  - Exhaustive searches in real life may be more powerful than they seem

Differential Propagation Analysis On Noekeon - Original

By searching the complete space of 4-round trails (both linear and differential) with less than 24 active S-boxes, the designers of NOEKEON can guarantee that

- $\nexists$ 4-round differential trails with a predicted prop ratio $> 2^{-48}$ and
- $\nexists$ 4-round linear trails with a correlation coefficient $> 2^{-24}$.

Differential Propagation Analysis On Noekeon-This work

- Of all 4-round differential trails, the *best* has a probability $= 2^{-51}$
- Of all 4-round linear trails, the *best* has a bias $= 2^{-25}$
- It takes 21 (1.2) hours to systematically investigate whether 4-round differential (linear) trails of weight up to 51 (25) exist on a PC.

Differential Propagation Analysis On Noekeon-This work

#R	NOEKEON-Differential				NOEKEON-Linear			
	Spec.		This		Spec.		This	
	ASN	Prob.	ASN	Prob.	ASN	Bias	ASN	Bias
1	1	2^{-2}	**1**	$\mathbf{2^{-2}}$	1	2^{-2}	**1**	$\mathbf{2^{-2}}$
2	4	2^{-8}	**4**	$\mathbf{2^{-8}}$	4	2^{-5}	**4**	$\mathbf{2^{-5}}$
3	-	-	***13**	$^{*}\mathbf{2^{-28}}$	-	-	***13**	$^{*}\mathbf{2^{-14}}$
4	-	$\leq 2^{-48}$	***22**	$^{*}\mathbf{2^{-51}}$	-	$\leq 2^{-25}$	***21**	$^{*}\mathbf{2^{-25}}$
5	-	-	**-**	$^{*}\leq \mathbf{2^{-65}}$	-	-	**-**	$^{*}\leq \mathbf{2^{-32}}$
6	-	-	**-**	$^{*}\leq \mathbf{2^{-80}}$	-	-	***33**	$^{*}\mathbf{2^{-40}}$

Differential Propagation Analysis On Noekeon-This work

- *Best* 6-round and 9-round linear trails with bias 2^{-40} and 2^{-62} are found out
- $\nexists$ 10-round differential trails with a predicted prop ratio $> 2^{-131}$ and
- $\nexists$ 11-round linear trails with a bias $> 2^{-71}$.

Differential Propagation Analysis On Spongent

#Round	Spec.		This		Spec.		This	
	ASN	Prob	ASN	Prob	ASN	Prob	ASN	Prob
	$b = 88$				$b = 136$			
5	10	2^{-21}	**10**	$*\mathbf{2^{-20}}$	10	2^{-22}	**10**	$*\mathbf{2^{-20}}$
10	20	2^{-47}	**20**	$\mathbf{2^{-47}}$	24	2^{-60}	*__22__	$*\mathbf{2^{-55}}$
15	30	2^{-74}	**30**	$\mathbf{2^{-74}}$	40	2^{-101}	*__43__	$\mathbf{2^{-96}}$
	$b = 176$				$b = 240$			
5	10	2^{-21}	**10**	$*\mathbf{2^{-20}}$	10	2^{-21}	**10**	$*\mathbf{2^{-20}}$
10	20	2^{-50}	**20**	$*\mathbf{2^{-46}}$	20	2^{-43}	**20**	$\mathbf{2^{-43}}$
15	30	2^{-79}	**30**	$\mathbf{2^{-79}}$	30	2^{-66}	**30**	$\mathbf{2^{-66}}$

Differential Propagation Analysis On Spongent

- For variants with $b = 88$, probability of *best* 17-round (18-round) differential trail is 2^{-86} (2^{-94}), which was found out within 1 minute.
- For variants with $b = 240$, probability of *best* 44-round differential trail is 2^{-196}, which was found within 1 minute. By observing the results up to 44-round, we can conclude the following: $Bw^6 = 30$ and for $r \geq 7$, $Bw^r = \begin{cases} Bw^{r-1} + 4 & if\ r\ is\ even \\ Bw^{r-1} + 5 & if\ r\ is\ odd \end{cases}$. An observation is that there is a 2-round iterative trail with weight pattern (4, 5) composing the best trails.
- For variants with $b = 176$, *best* 17-round (18-round) differential trail with weight 91 (99) was found within 50 minutes.

Future Works Overviews

- Start From the Narrowest Point
  - Start from the narrowest place and choose a direction intelligently
  - Start from two narrowest point and meet in a proper point
- Try In Minimal Changes Order - Explore more details on the cipher
  - Classify the search patterns to be pre-searched
  - Classify the candidates: generate the candidates from a generator element with small effort
  - Characterize the candidates: assign a weight (e.g. Hamming) to the candidates to bound the search

Future Works Overviews

- Start From the Narrowest Point
  - Start from the narrowest place and choose a direction intelligently
  - Start from two narrowest point and meet in a proper point
- Try In Minimal Changes Order - Explore more details on the cipher
  - Classify the search patterns to be pre-searched
  - Classify the candidates: generate the candidates from a generator element with small effort
  - Characterize the candidates: assign a weight (e.g. Hamming) to the candidates to bound the search

Future Work

- We trial a search pattern in an order of $(\breve{w}_0, \breve{w}_1, \ldots, \breve{w}_{n-m}, \breve{w}_{-1}, \ldots, \breve{w}_{-m+1})$. As an anonymous reviewers suggested, it might also be interesting to consider the order $(\breve{w}_0, \breve{w}_1, \breve{w}_{-1}, \breve{w}_2, \breve{w}_{-2}, \ldots)$.
- How to use empirical knowledge to add heuristics to the search algorithm remains unclear.
- By avoiding detailed properties of the target ciphers, our algorithm is general to some extent, while remaining space for further improvement by utilizing more special properties of the object ciphers.
- How about the efficiency when adopted them to the case of related-key differential?

Future Work

- We trial a search pattern in an order of $(\breve{w}_0, \breve{w}_1, \ldots, \breve{w}_{n-m}, \breve{w}_{-1}, \ldots, \breve{w}_{-m+1})$. As an anonymous reviewers suggested, it might also be interesting to consider the order $(\breve{w}_0, \breve{w}_1, \breve{w}_{-1}, \breve{w}_2, \breve{w}_{-2}, \ldots)$.
- How to use empirical knowledge to add heuristics to the search algorithm remains unclear.
- By avoiding detailed properties of the target ciphers, our algorithm is general to some extent, while remaining space for further improvement by utilizing more special properties of the object ciphers.
- How about the efficiency when adopted them to the case of related-key differential?

Future Work

- We trial a search pattern in an order of $(\breve{w}_0, \breve{w}_1, \ldots, \breve{w}_{n-m}, \breve{w}_{-1}, \ldots, \breve{w}_{-m+1})$. As an anonymous reviewers suggested, it might also be interesting to consider the order $(\breve{w}_0, \breve{w}_1, \breve{w}_{-1}, \breve{w}_2, \breve{w}_{-2}, \ldots)$.
- How to use empirical knowledge to add heuristics to the search algorithm remains unclear.
- By avoiding detailed properties of the target ciphers, our algorithm is general to some extent, while remaining space for further improvement by utilizing more special properties of the object ciphers.
- How about the efficiency when adopted them to the case of related-key differential?

Future Work

- We trial a search pattern in an order of $(\breve{w}_0, \breve{w}_1, \ldots, \breve{w}_{n-m}, \breve{w}_{-1}, \ldots, \breve{w}_{-m+1})$. As an anonymous reviewers suggested, it might also be interesting to consider the order $(\breve{w}_0, \breve{w}_1, \breve{w}_{-1}, \breve{w}_2, \breve{w}_{-2}, \ldots)$.
- How to use empirical knowledge to add heuristics to the search algorithm remains unclear.
- By avoiding detailed properties of the target ciphers, our algorithm is general to some extent, while remaining space for further improvement by utilizing more special properties of the object ciphers.
- How about the efficiency when adopted them to the case of related-key differential?

Thank You!

Questions?