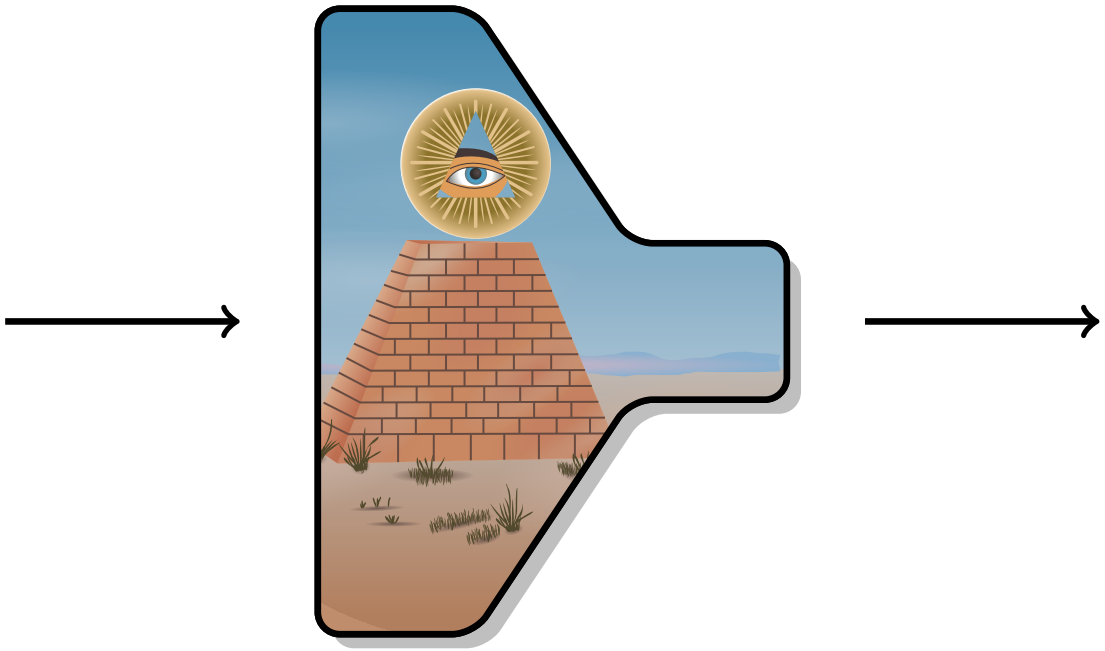
Boomerang Attacks against ARX Hash Functions

Gaëtan Leurent

University of Luxembourg

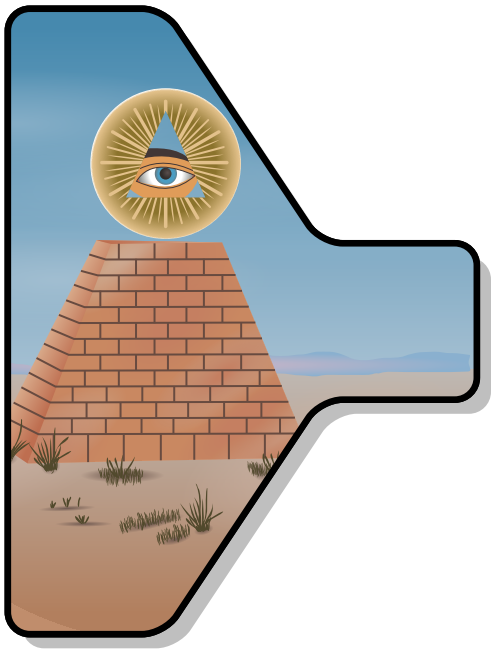
January 24, 2012

An Ideal Hash Function: the Random Oracle



- Public Random Oracle
- The output can be used as a fingerprint of the document

An Ideal Hash Function: the Random Oracle



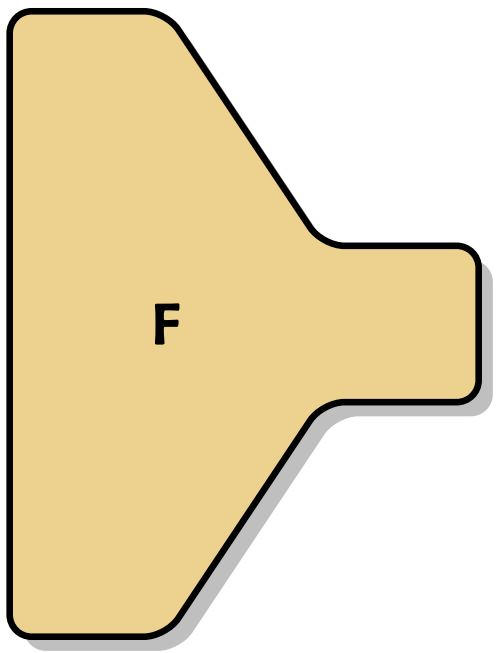
`0x1d66ca77ab361c6f`

- Public Random Oracle
- The output can be used as a fingerprint of the document

A Concrete Hash Function

- A public function with no structural property.
  - Cryptographic strength without any key!

- $F : \{0,1\}^* \longrightarrow \{0,1\}^n$

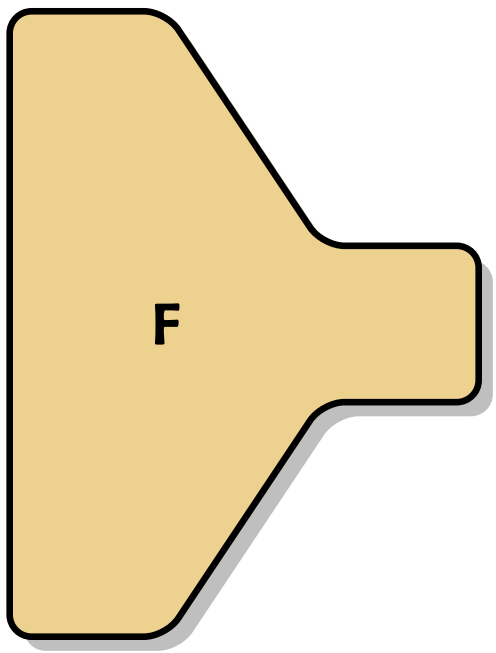


0x1d66ca77ab361c6f

A Concrete Hash Function

- A public function with no structural property.
  - Cryptographic strength without any key!

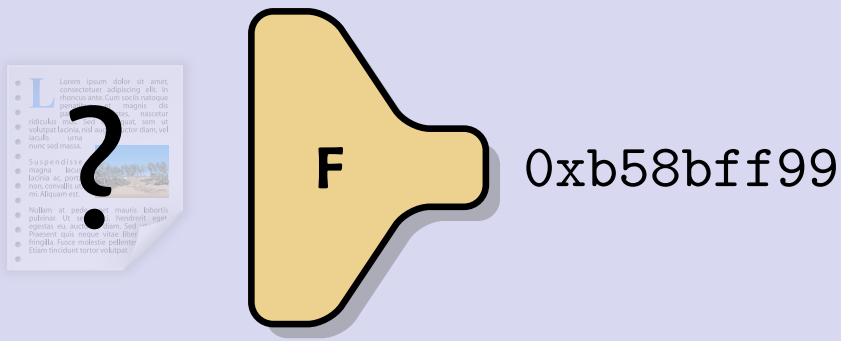
- $F : \{0,1\}^* \longrightarrow \{0,1\}^n$



0x1d66ca77ab361c6f

Security goals

Preimage attack

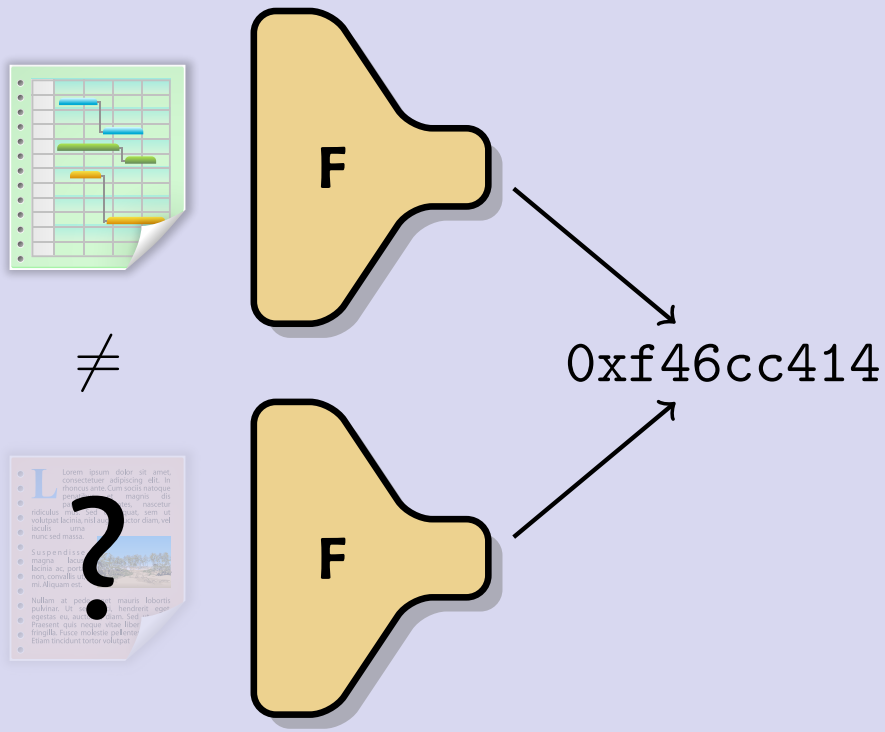


Given F and $\overline{H}$, find M s.t. $F(M) = \overline{H}$.

Ideal security: 2^n.

Security goals

Second-preimage attack

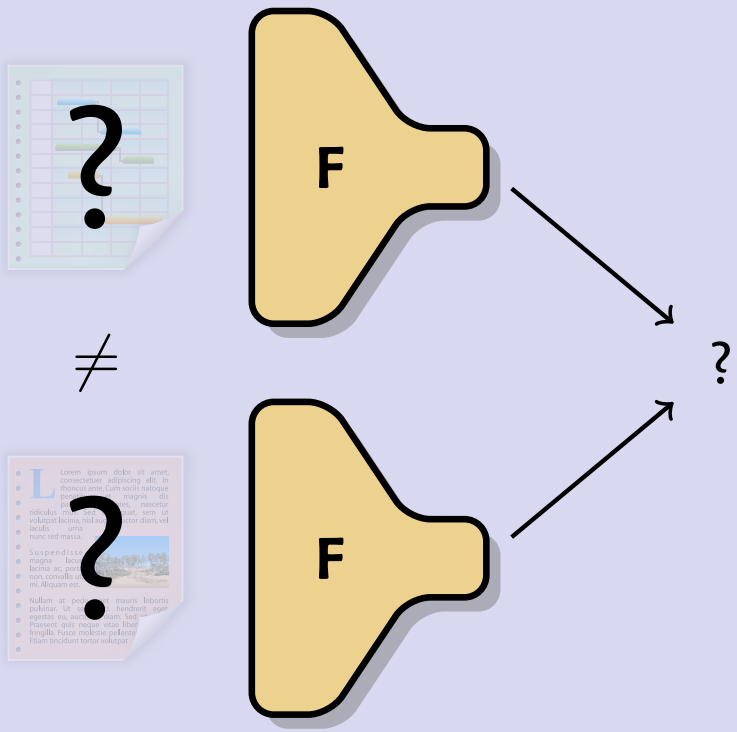


Given F and M_1, find $M_2 \neq M_1$ s.t. $F(M_1) = F(M_2)$.

Ideal security: 2^n.

Security goals

Collision attack



Given F, find $M_1 \neq M_2$ s.t. $F(M_1) = F(M_2)$.

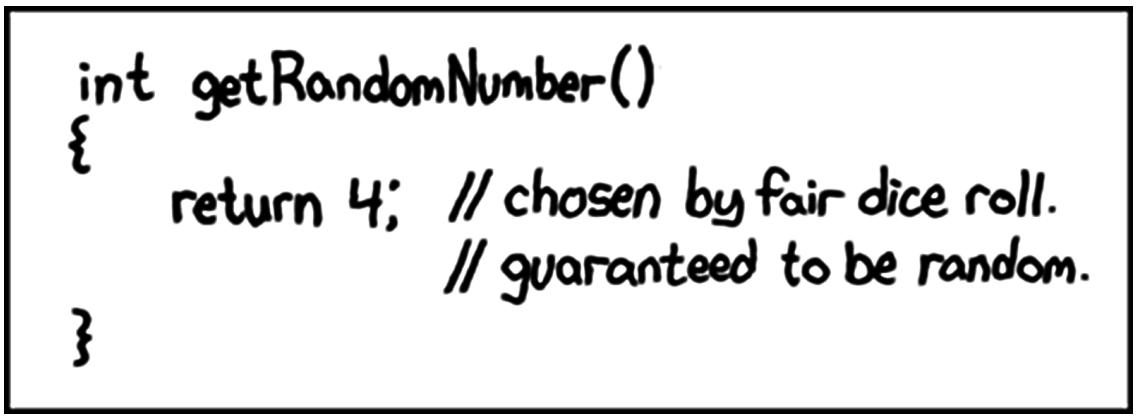
Ideal security: $2^{n/2}$.

Using Hash Functions

Hash functions are used in many different contexts:

- To generate unique identifiers
  - Hash-and-sign signatures
  - Commitment schemes
- As a one-way function
  - One-Time-Passwords
  - Forward security
- To break the structure of the input
  - Entropy extractors
  - Key derivation
  - Pseudo-random number generator
- To build MACs
  - HMAC
  - Challenge/response authentication

Security definitions: difficulties

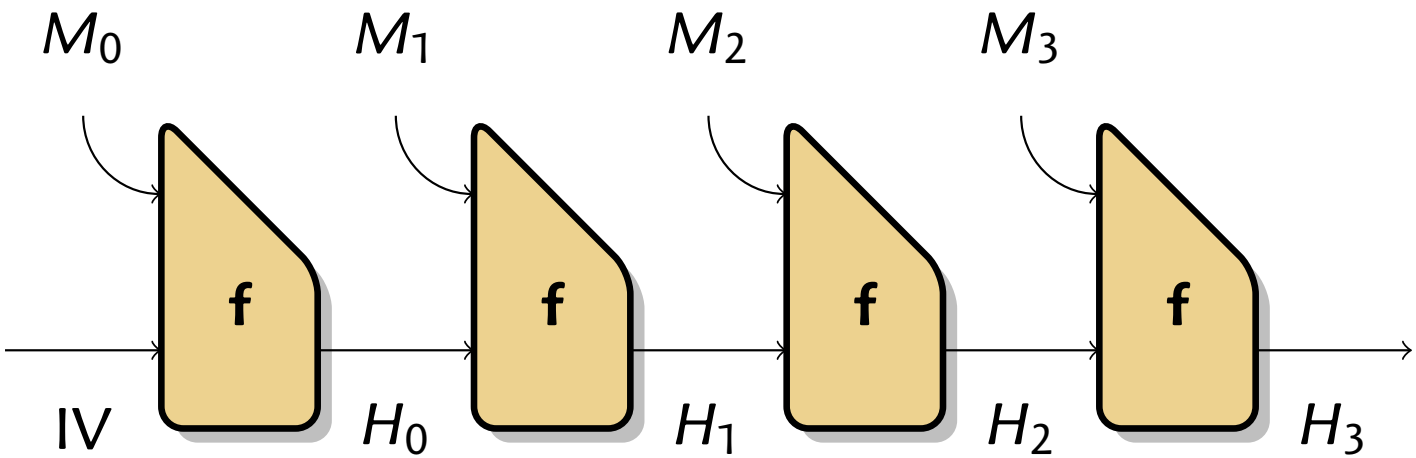


http://xkcd.com/221/

- A single function can not be collision resistant.
  - Precomputation is allowed in standard security definition
  - Define a family of function
- Obvious relations between the security definitions do not hold.
  - Even more mess with families of functions!

Hash function design

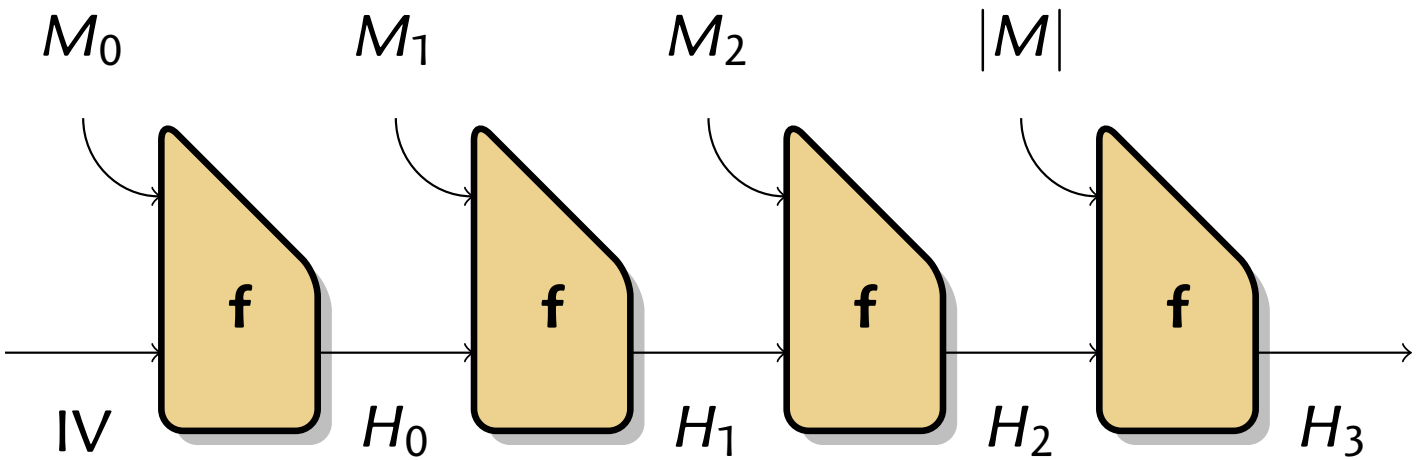
- Build a small compression function, and iterate.
  - Cut the message in chunks $M_0, \ldots M_k$
  - $H_i = f(M_i, H_{-1})$
  - $F(M) = H_k$



Security proof (Merkle, Damgård)

Theorem

If one finds a collision in the hash function,
then one has a collision in the compression function.

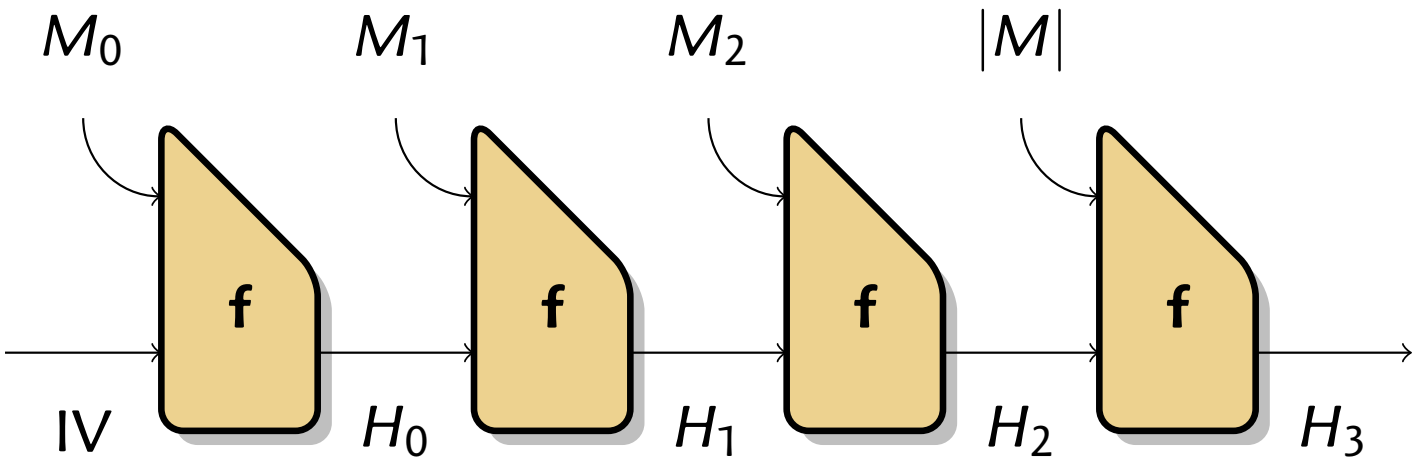


- If $|M| \neq |M'|$, collision in last block.
- Else, look for last block with $H_i = H'_i$.
- The converse is not true

Security proof (Merkle, Damgård)

Theorem

If one finds a collision in the hash function,
then one has a collision in the compression function.



- If $|M| \neq |M'|$, collision in last block.
- Else, look for last block with $H_i = H'_i$.
- The converse is not true

Compression Function Attacks

Fist results usually target the compression function

- Because it's easier: more degrees of freedom
- Because good compression function imply good hash function

Compression Function Attacks

- Hash function collision: $f(\mathrm{IV}, m) = f(\mathrm{IV}, m')$
- Free-start collision: $f(h, m) = f(h, m')$
- Semi-free-start collision: $f(h, m) = f(h', m')$
- Near-collision: $\mathrm{hw}(f(h, m) \oplus f(h', m'))$ small
- Zero-sum: $\sum h_i = 0,\ \sum m_i = 0,\ \sum f(h_i, m_i) = 0$
- Attack on a truncated output, Bias, ...

Compression Function Attacks

Fist results usually target the compression function

- Because it's easier: more degrees of freedom
- Because good compression function imply good hash function

MD5 cryptanalysis

- 1993: Free-start collisions [den Boer and Bosselaers]
- 1996: Semi-free-start collisions [Dobbertin]
- 2005: Collisions [Wang *et. al*]
- 2009: Rogue certificate [Stevens *et. al*]

Wang's and Stevens's attacks are based on the dBB path

Operations used

Two main categories of designs:

ARX designs

- Additions, Rotations, Xors
- Inspired by MD/SHA
- Lots of light rounds
- Bit-level attack
- Known attacks techniques, but finding paths is hard

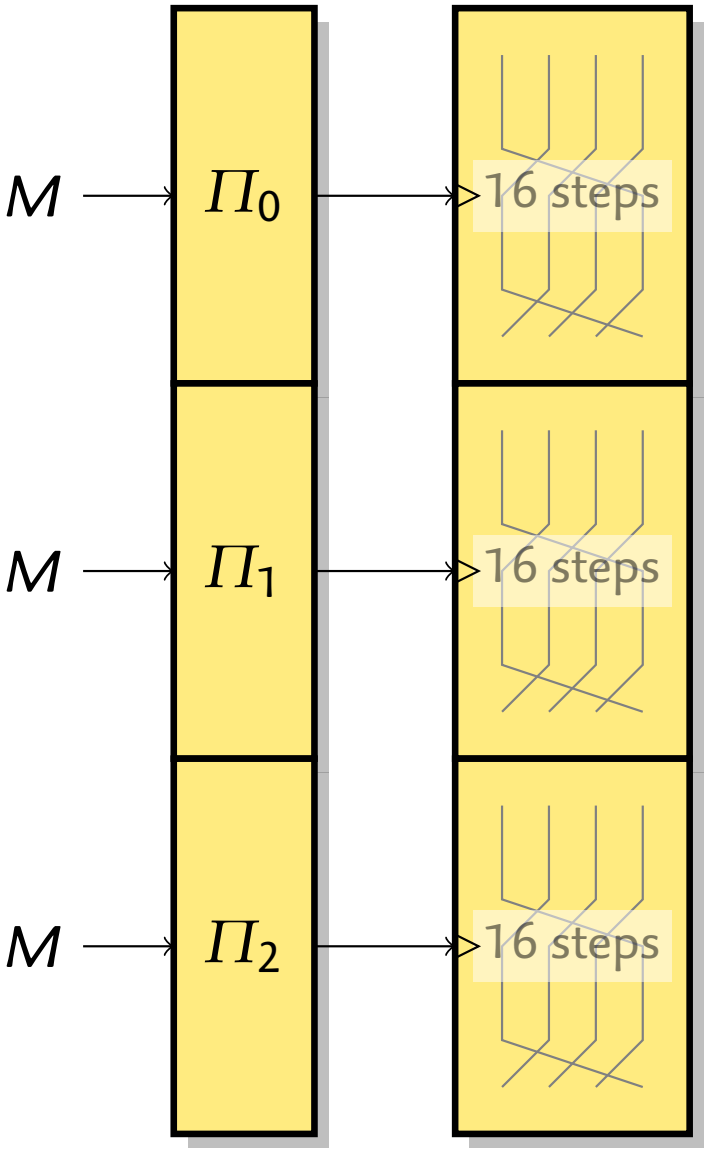
SBox designs

- SBoxes and Linear Layers
- Inspired by the AES
- Few heavy rounds
- SBox-level attacks
- New attacks techniques, *e.g.* rebound attack

Wang et. al's attacks

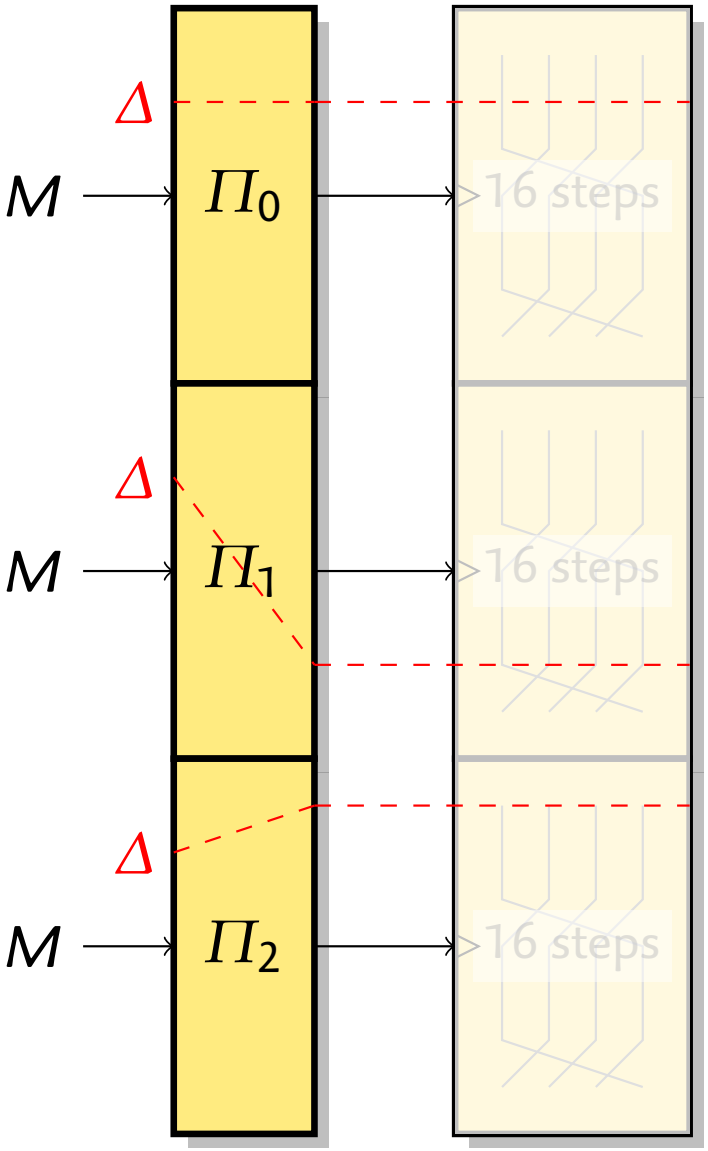
- Based on a differential attack:
  - Consider a pair of message with a small difference
  - Try to control the propagation of the differences
- New ideas:
  - Use a signed difference
  - Use a set of sufficient conditions
  - Some conditions are easy to satisfy: message modification

Wang et al.'s Attack



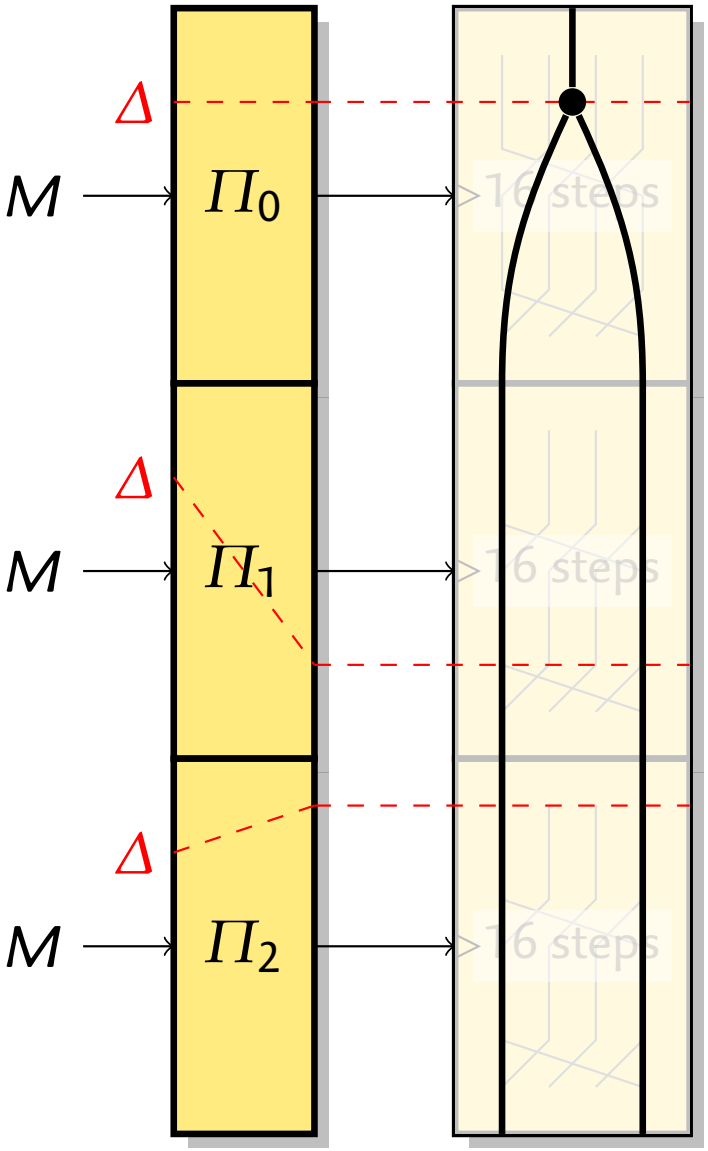
1. Precomputation:
   - Choose a message difference.
   - Build a differential path.
   - Derive a set of sufficient conditions.
2. Collision search:
   - Start with a random message, check the conditions
   - Use message modifications

Wang et al.'s Attack



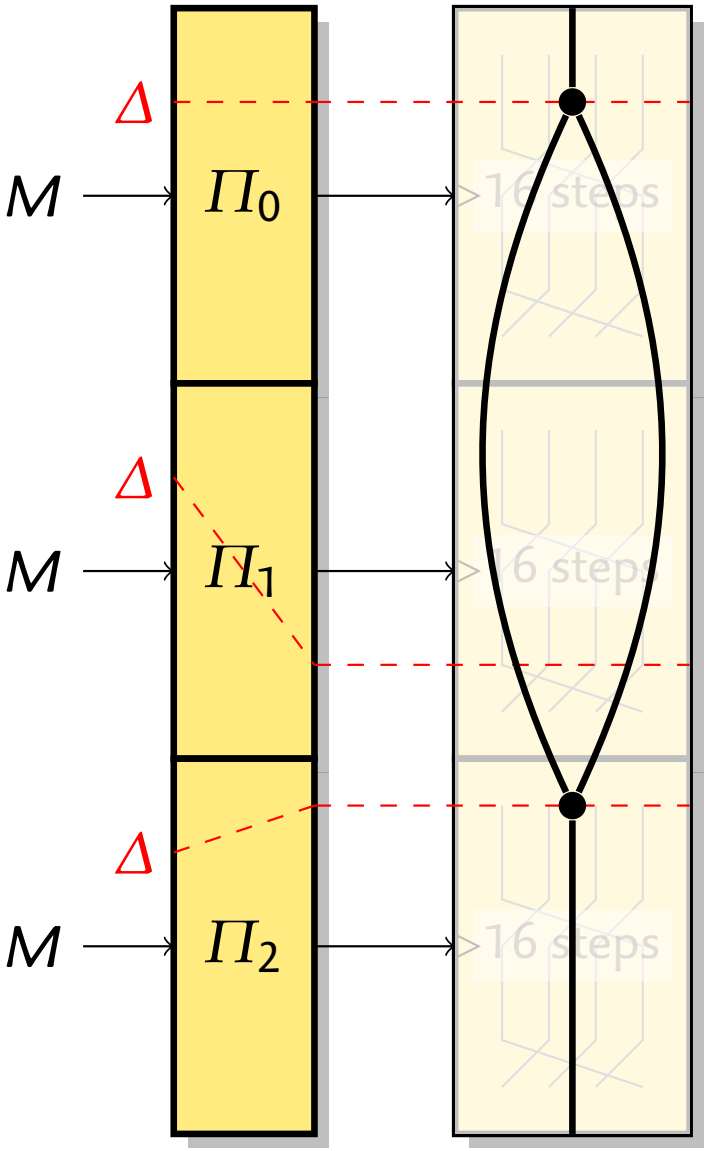
1. Precomputation:
   - Choose a message difference.
   - Build a differential path.
   - Derive a set of sufficient conditions.
2. Collision search:
   - Start with a random message, check the conditions
   - Use message modifications

Wang et al.'s Attack



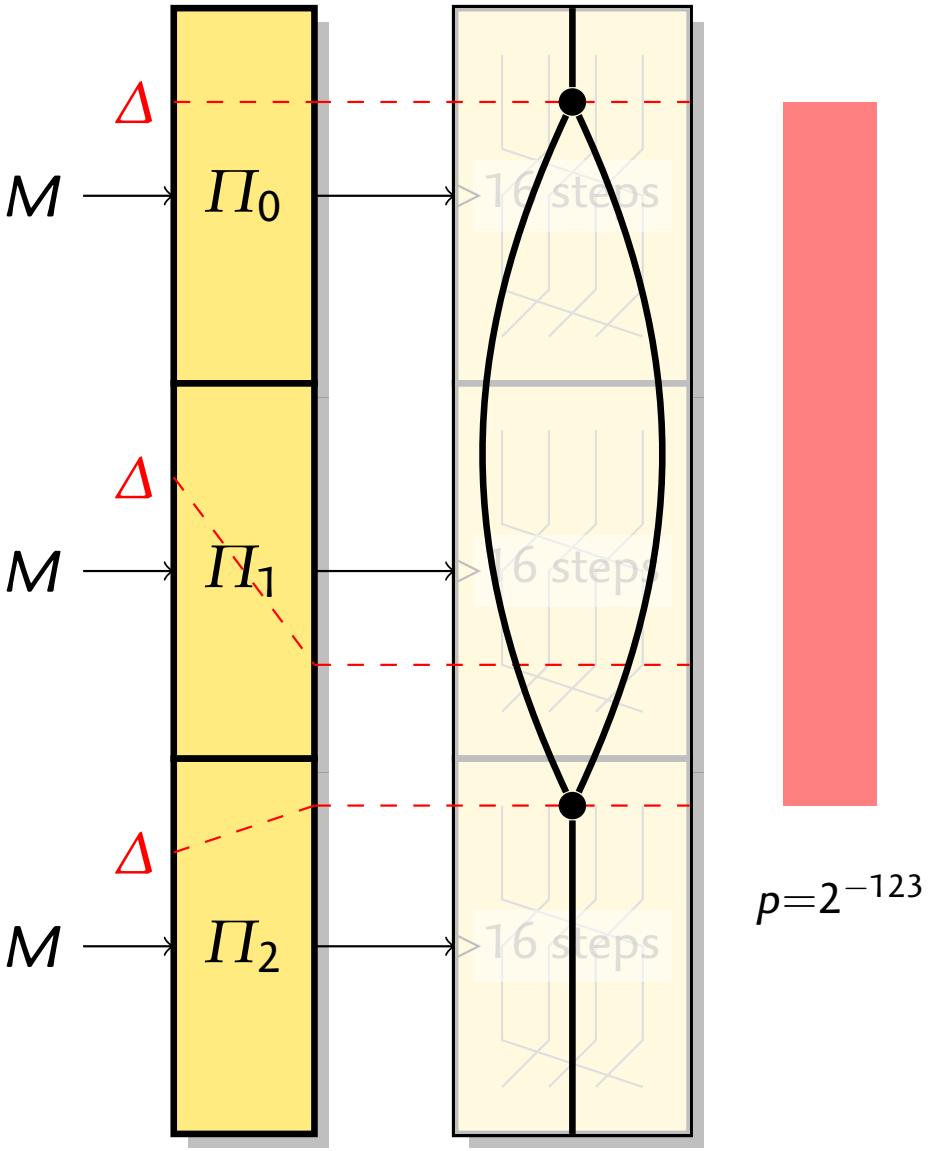
1. Precomputation:
   - Choose a message difference.
   - Build a differential path.
   - Derive a set of sufficient conditions.
2. Collision search:
   - Start with a random message, check the conditions
   - Use message modifications

Wang et al.'s Attack



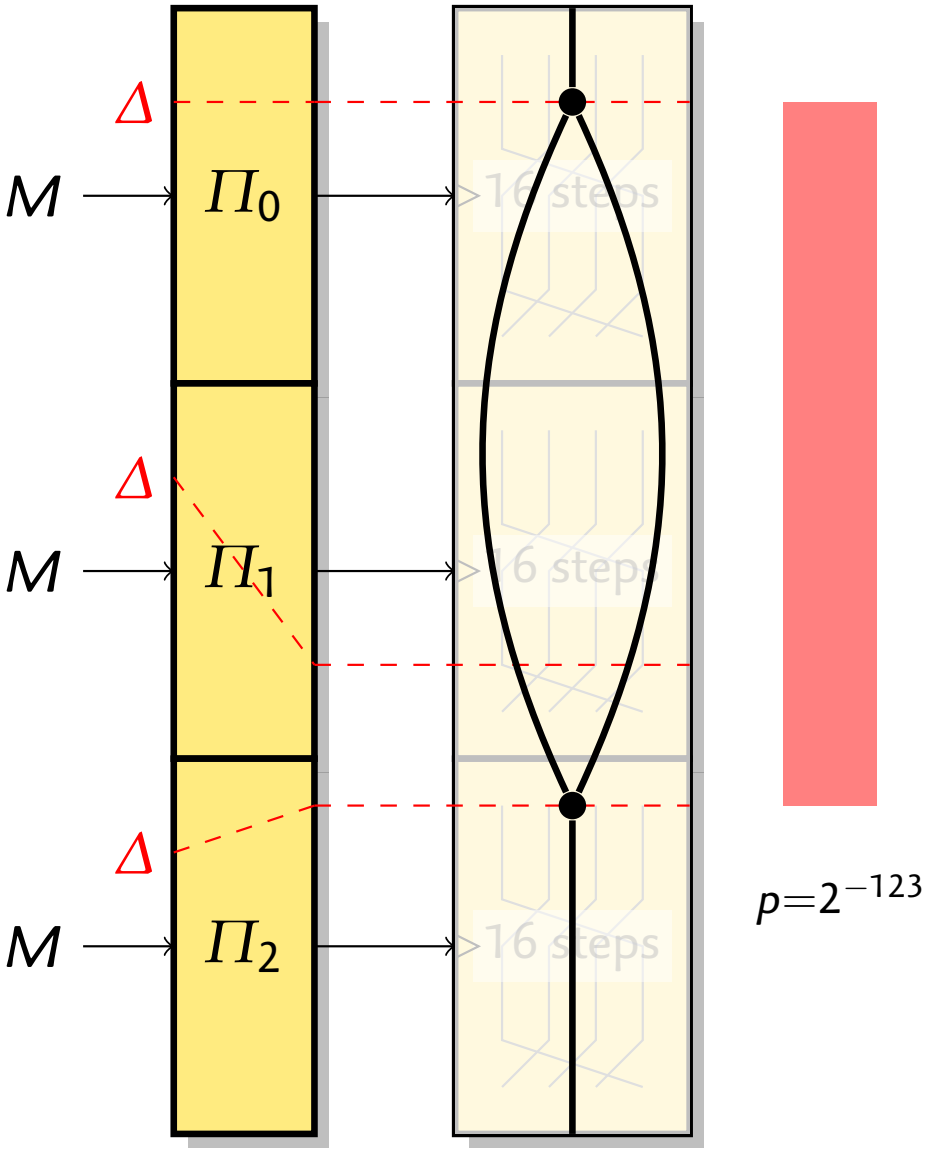
1. Precomputation:
   - Choose a message difference.
   - Build a differential path.
   - Derive a set of sufficient conditions.
2. Collision search:
   - Start with a random message, check the conditions
   - Use message modifications

Wang et al.'s Attack



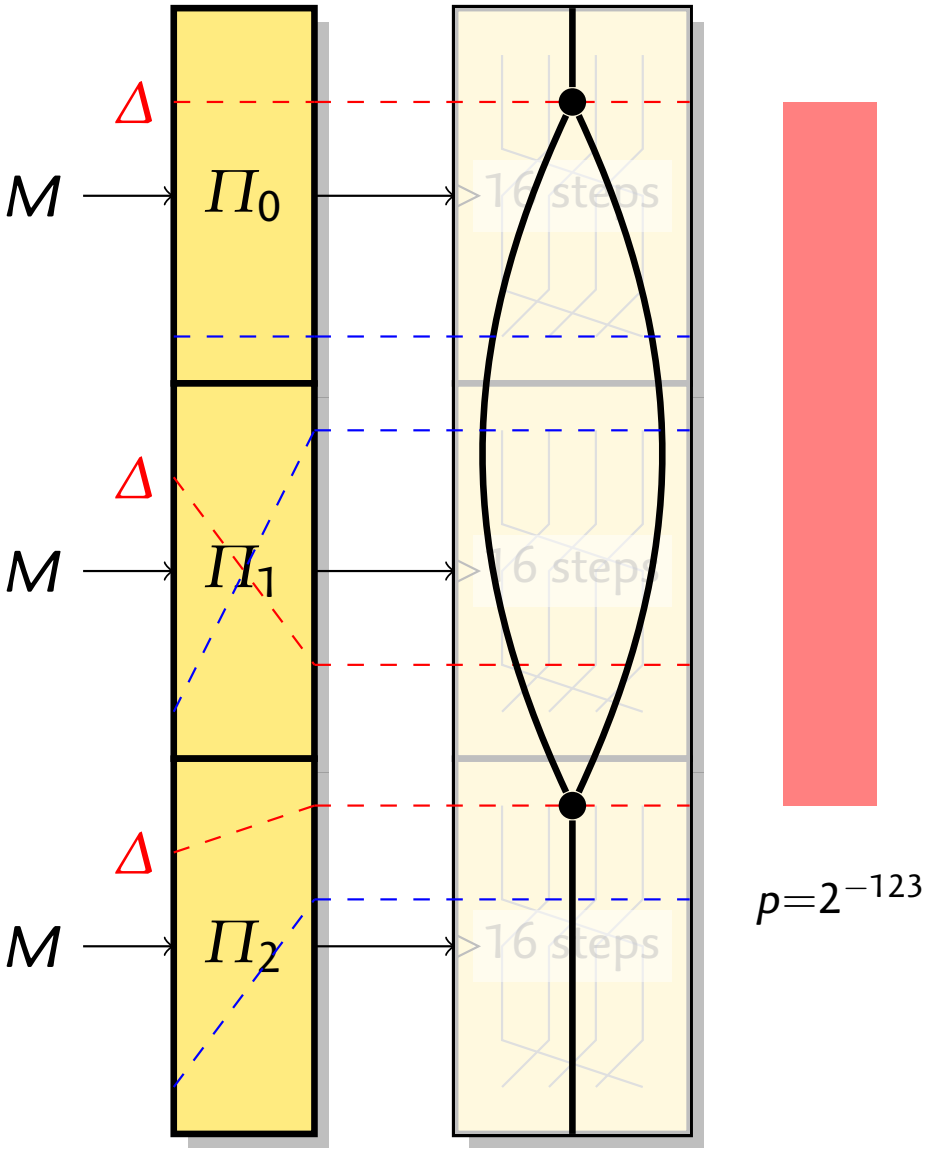
1. Precomputation:
   - Choose a message difference.
   - Build a differential path.
   - Derive a set of sufficient conditions.
2. Collision search:
   - Start with a random message, check the conditions
   - Use message modifications

Wang et al.'s Attack



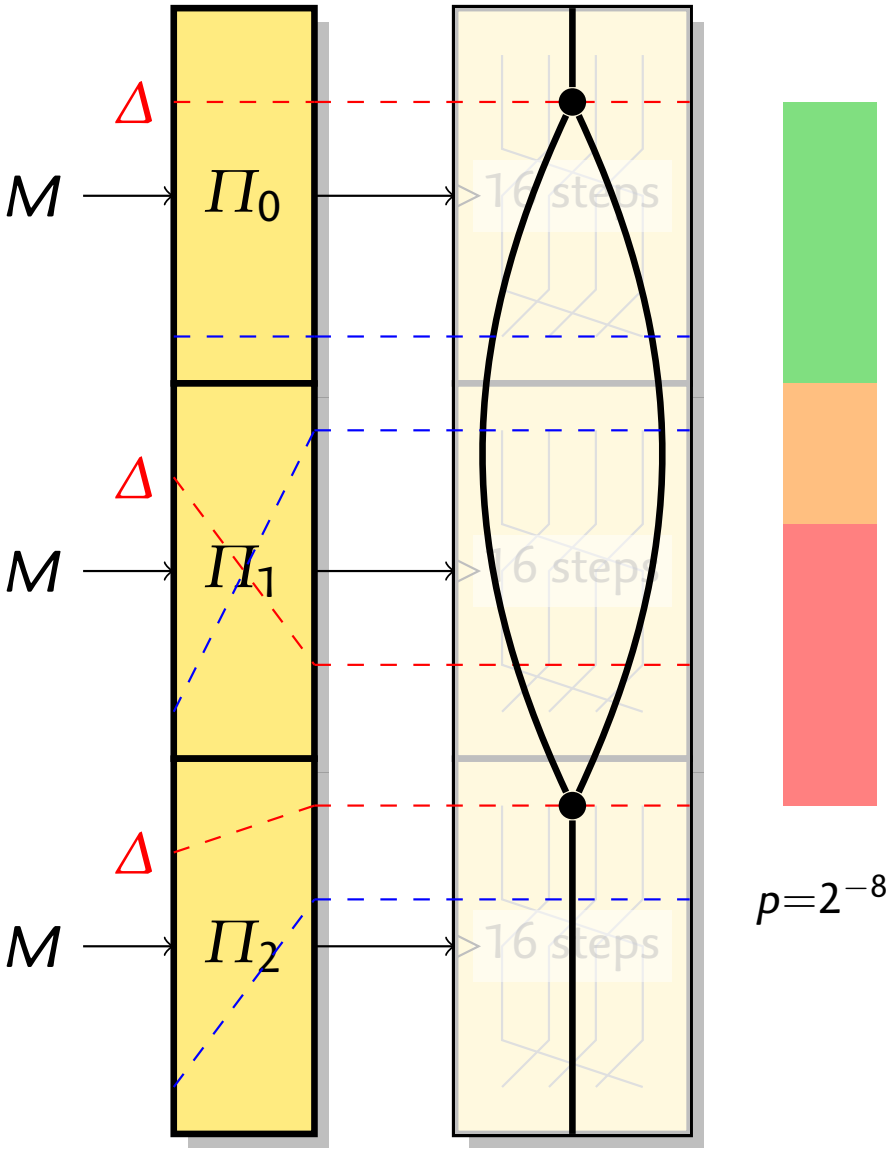
1. Precomputation:
   - Choose a message difference.
   - Build a differential path.
   - Derive a set of sufficient conditions.
2. Collision search:
   - Start with a random message, check the conditions
   - Use message modifications

Wang et al.'s Attack



1. Precomputation:
   - Choose a message difference.
   - Build a differential path.
   - Derive a set of sufficient conditions.
2. Collision search:
   - Start with a random message, check the conditions
   - Use message modifications

Wang et al.'s Attack



1. Precomputation:
   - Choose a message difference.
   - Build a differential path.
   - Derive a set of sufficient conditions.
2. Collision search:
   - Start with a random message, check the conditions
   - Use message modifications

The SHA-3 competition

After the attacks on the MD4 family, we need new hash functions

The SHA-3 competition

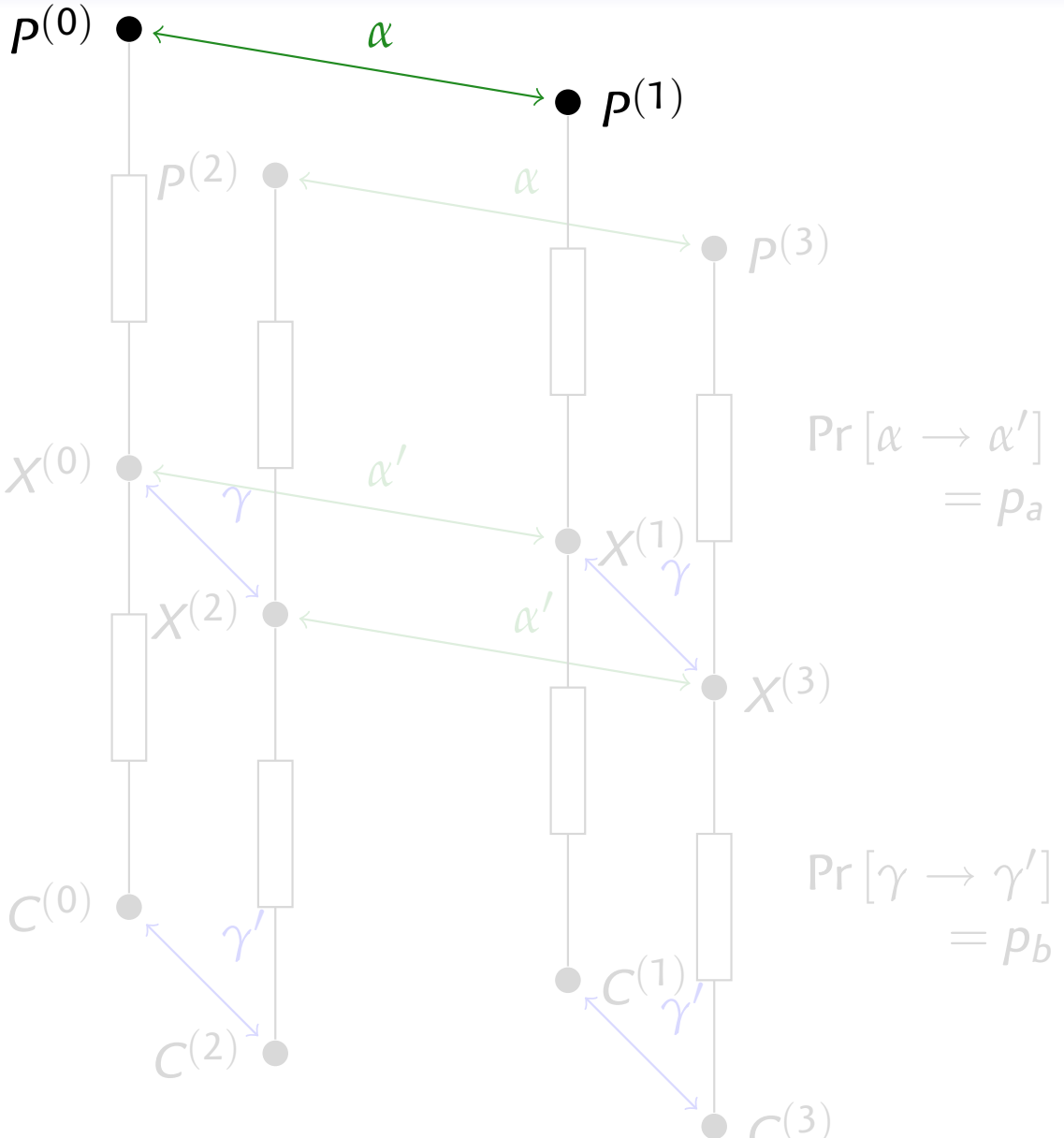
- Organized by NIST
- Similar to the AES competition
- Submission deadline was October 2008: 64 candidiates
- 51 valid submissions
- 14 in the second round (July 2009)
- 5 finalists in December 2010
- Winner in 2012?

Outline

Boomerang Attacks

- Introduced by Wagner, many later improvements
- Combine two short differentials instead of using a long one.
  - $f = f_b \circ f_a$
  - for f_a, $\alpha \longrightarrow \alpha'$ with probability p_a
  - for f_b, $\gamma \longrightarrow \gamma'$ with probability p_b
- Uses an encryprion oracle together with a decryption oracle

Boomerang Attacks



$$\Pr[\alpha \to \alpha'] = p_a$$

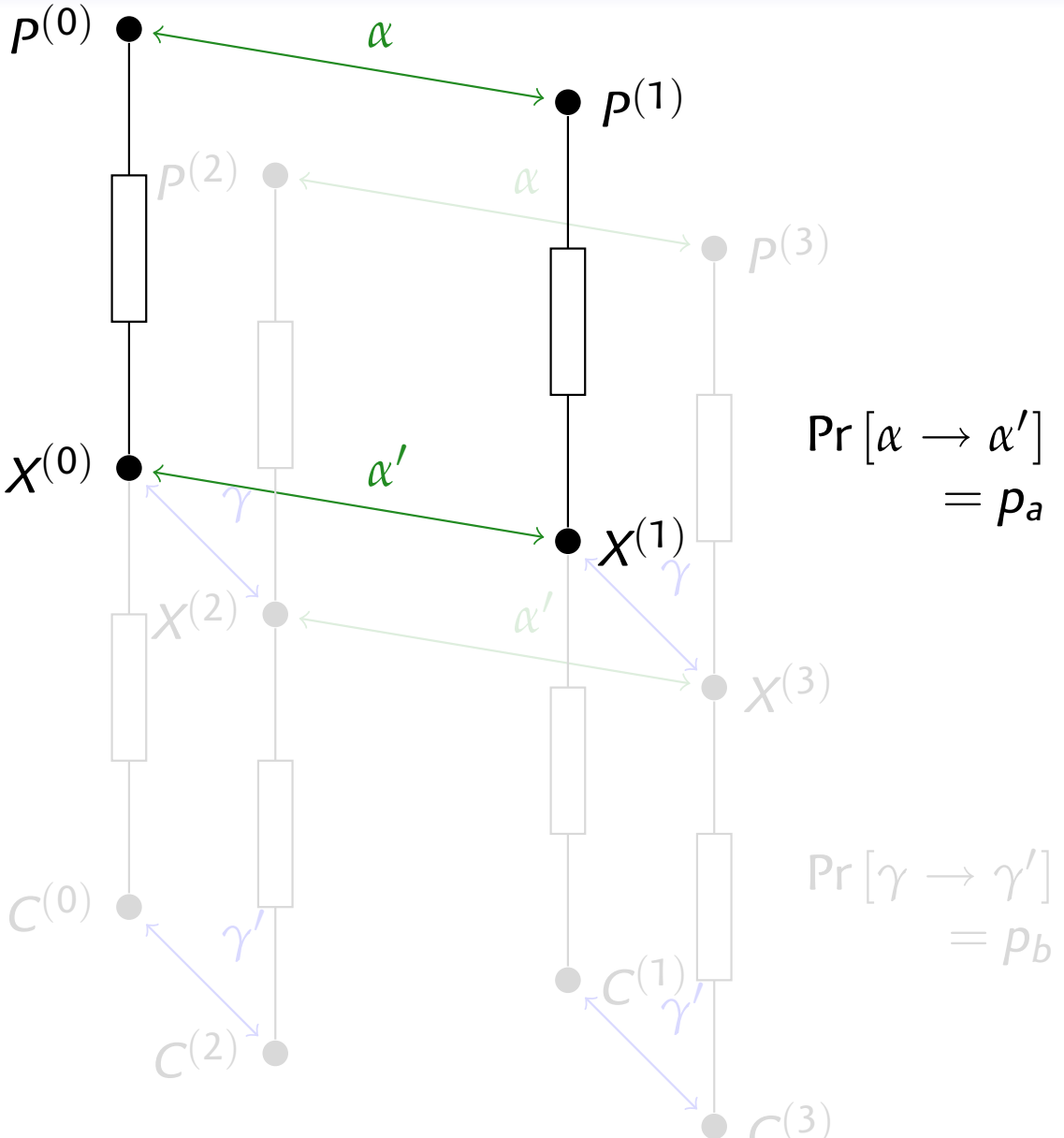
$$\Pr[\gamma \to \gamma'] = p_b$$

1. Start with $P^{(0)}, P^{(1)}$
2. Compute $C^{(0)}, C^{(1)}$
3. Build $C^{(2)}, C^{(3)}$
4. Compute $P^{(2)}, P^{(3)}$

$$C = \frac{1}{p_a} \frac{1}{p_b^2} \frac{1}{p_a}$$

$$P^{(0)} \oplus P^{(1)} = \alpha$$
$$P^{(2)} \oplus P^{(3)} = \alpha$$
$$C^{(0)} \oplus C^{(1)} = \gamma'$$
$$C^{(2)} \oplus C^{(3)} = \gamma'$$

Boomerang Attacks



$$\Pr[\alpha \longrightarrow \alpha'] = p_a$$

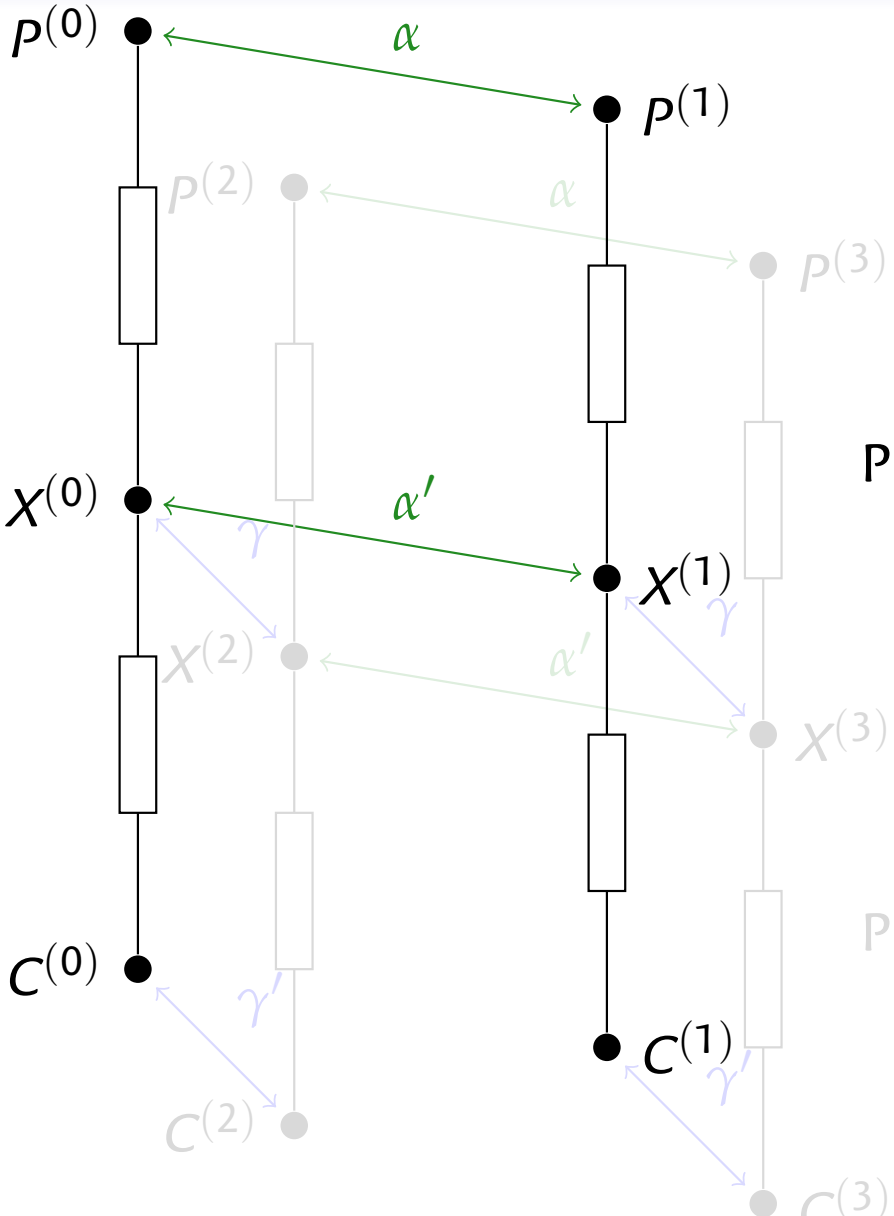
$$\Pr[\gamma \longrightarrow \gamma'] = p_b$$

1. Start with $P^{(0)}, P^{(1)}$
2. Compute $C^{(0)}, C^{(1)}$
3. Build $C^{(2)}, C^{(3)}$
4. Compute $P^{(2)}, P^{(3)}$

$$C = \frac{1}{p_a} \frac{1}{p_b^2} \frac{1}{p_a}$$

$$P^{(0)} \oplus P^{(1)} = \alpha$$
$$P^{(2)} \oplus P^{(3)} = \alpha$$
$$C^{(0)} \oplus C^{(1)} = \gamma'$$
$$C^{(2)} \oplus C^{(3)} = \gamma'$$

Boomerang Attacks



$$\Pr[\alpha \longrightarrow \alpha'] = p_a$$

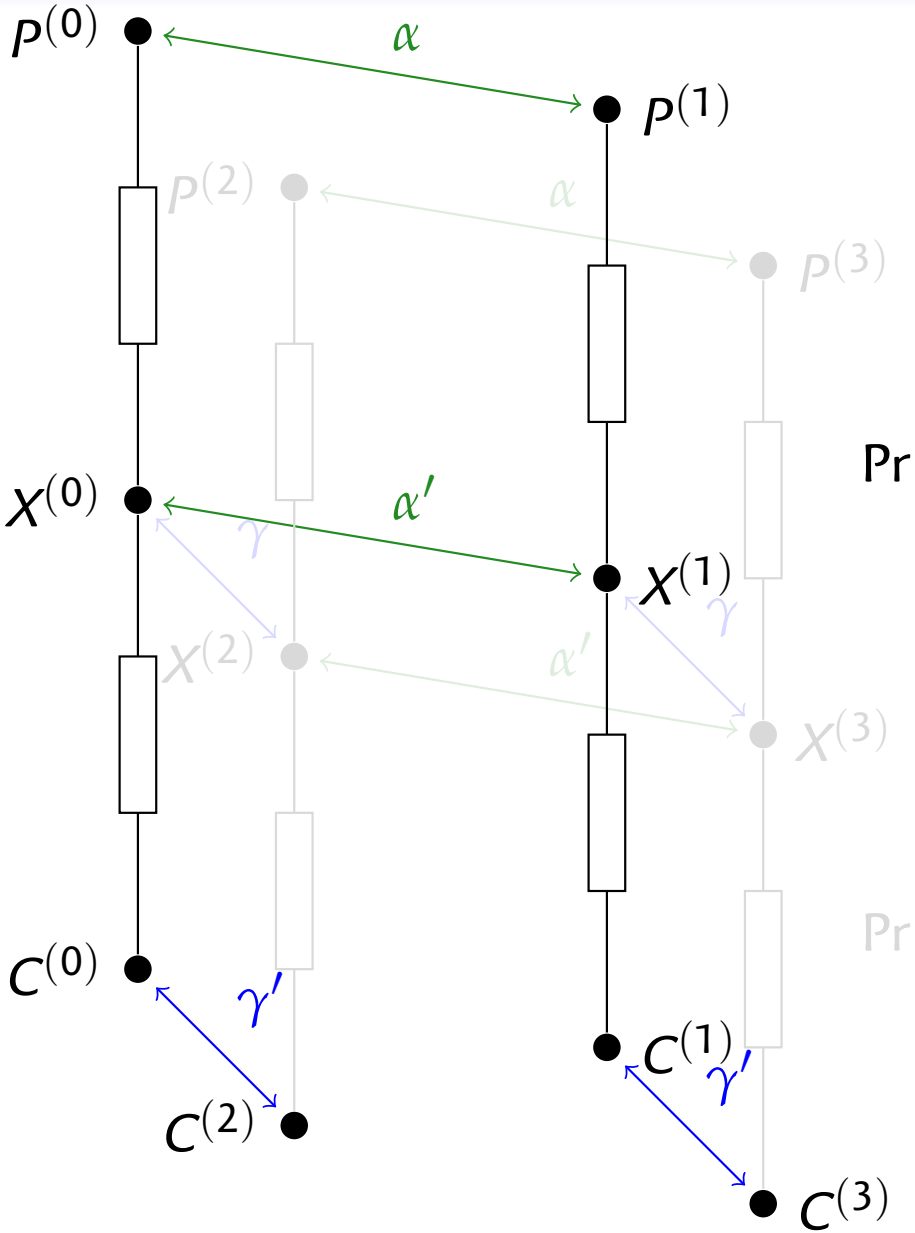
$$\Pr[\gamma \longrightarrow \gamma'] = p_b$$

1. Start with $P^{(0)}, P^{(1)}$
2. Compute $C^{(0)}, C^{(1)}$
3. Build $C^{(2)}, C^{(3)}$
4. Compute $P^{(2)}, P^{(3)}$

$$C = \frac{1}{p_a} \frac{1}{p_b^2} \frac{1}{p_a}$$

$$P^{(0)} \oplus P^{(1)} = \alpha$$
$$P^{(2)} \oplus P^{(3)} = \alpha$$
$$C^{(0)} \oplus C^{(1)} = \gamma'$$
$$C^{(2)} \oplus C^{(3)} = \gamma'$$

Boomerang Attacks



$$\Pr[\alpha \longrightarrow \alpha'] = p_a$$

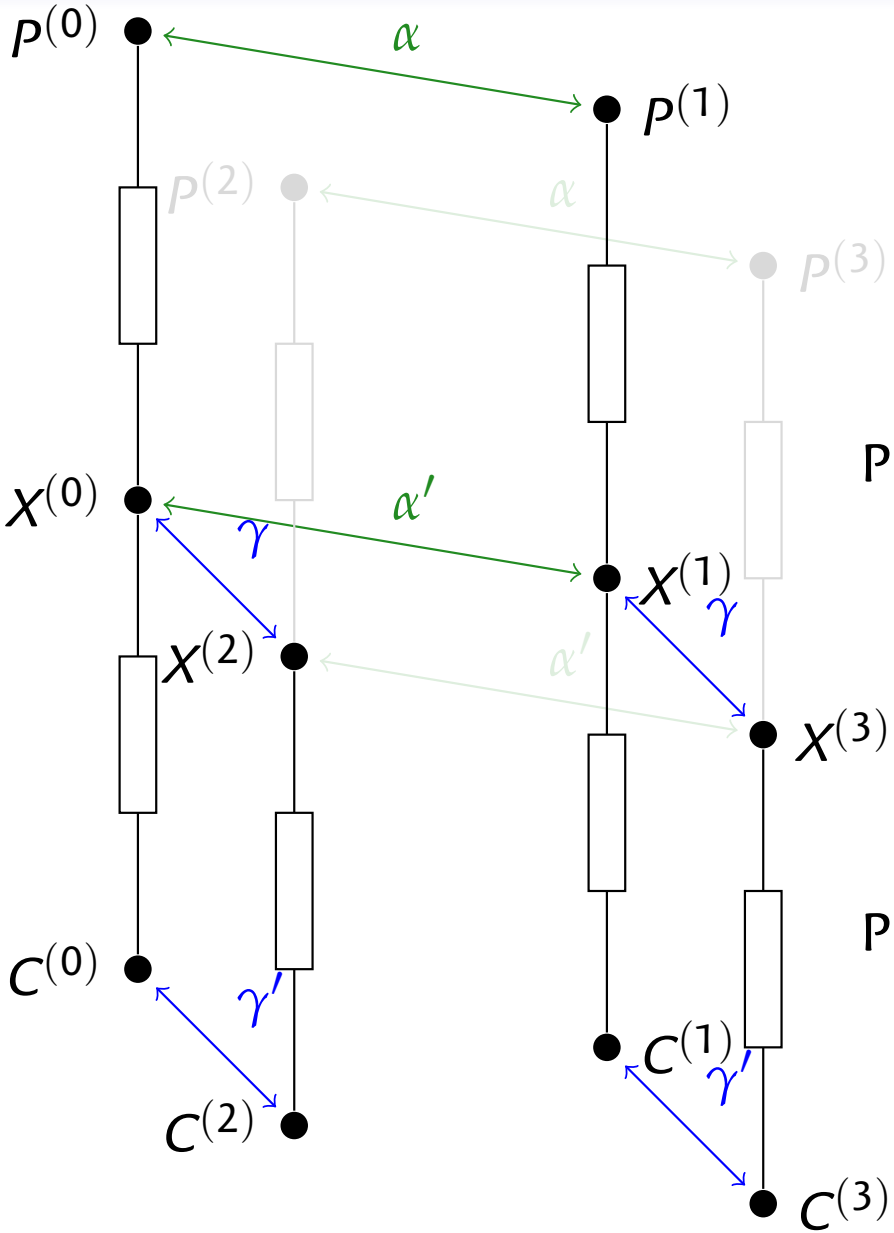
$$\Pr[\gamma \longrightarrow \gamma'] = p_b$$

1. Start with $P^{(0)}, P^{(1)}$
2. Compute $C^{(0)}, C^{(1)}$
3. Build $C^{(2)}, C^{(3)}$
4. Compute $P^{(2)}, P^{(3)}$

$$C = \frac{1}{p_a} \frac{1}{p_b^2} \frac{1}{p_a}$$

$$P^{(0)} \oplus P^{(1)} = \alpha$$
$$P^{(2)} \oplus P^{(3)} = \alpha$$
$$C^{(0)} \oplus C^{(1)} = \gamma'$$
$$C^{(2)} \oplus C^{(3)} = \gamma'$$

Boomerang Attacks



$$\Pr[\alpha \longrightarrow \alpha'] = p_a$$

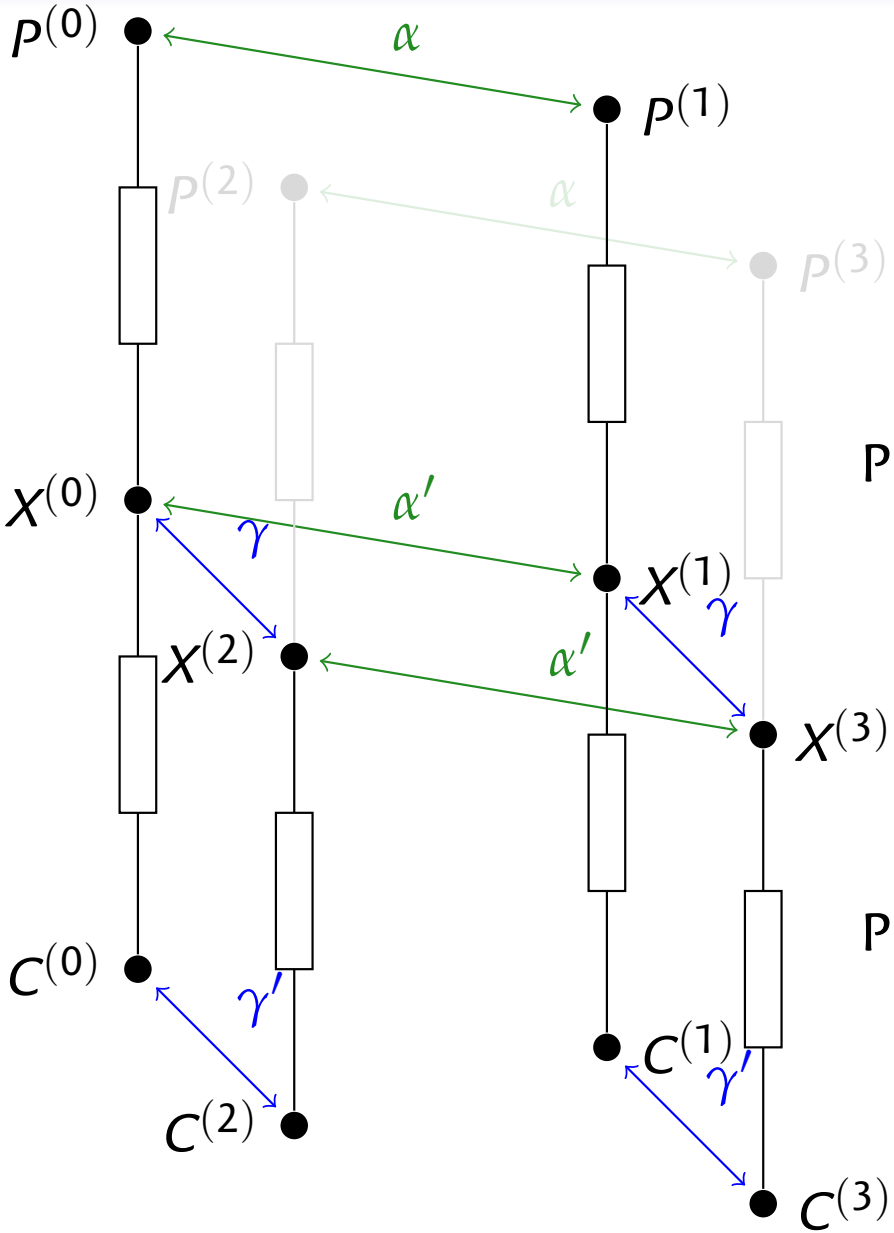
$$\Pr[\gamma \longrightarrow \gamma'] = p_b$$

1. Start with $P^{(0)}, P^{(1)}$
2. Compute $C^{(0)}, C^{(1)}$
3. Build $C^{(2)}, C^{(3)}$
4. Compute $P^{(2)}, P^{(3)}$

$$C = \frac{1}{p_a} \frac{1}{p_b^2} \frac{1}{p_a}$$

$$P^{(0)} \oplus P^{(1)} = \alpha$$
$$P^{(2)} \oplus P^{(3)} = \alpha$$
$$C^{(0)} \oplus C^{(1)} = \gamma'$$
$$C^{(2)} \oplus C^{(3)} = \gamma'$$

Boomerang Attacks



$$\Pr[\alpha \longrightarrow \alpha'] = p_a$$

$$\Pr[\gamma \longrightarrow \gamma'] = p_b$$

1. Start with $P^{(0)}, P^{(1)}$
2. Compute $C^{(0)}, C^{(1)}$
3. Build $C^{(2)}, C^{(3)}$
4. Compute $P^{(2)}, P^{(3)}$

$$C = \frac{1}{p_a} \frac{1}{p_b^2} \frac{1}{p_a}$$

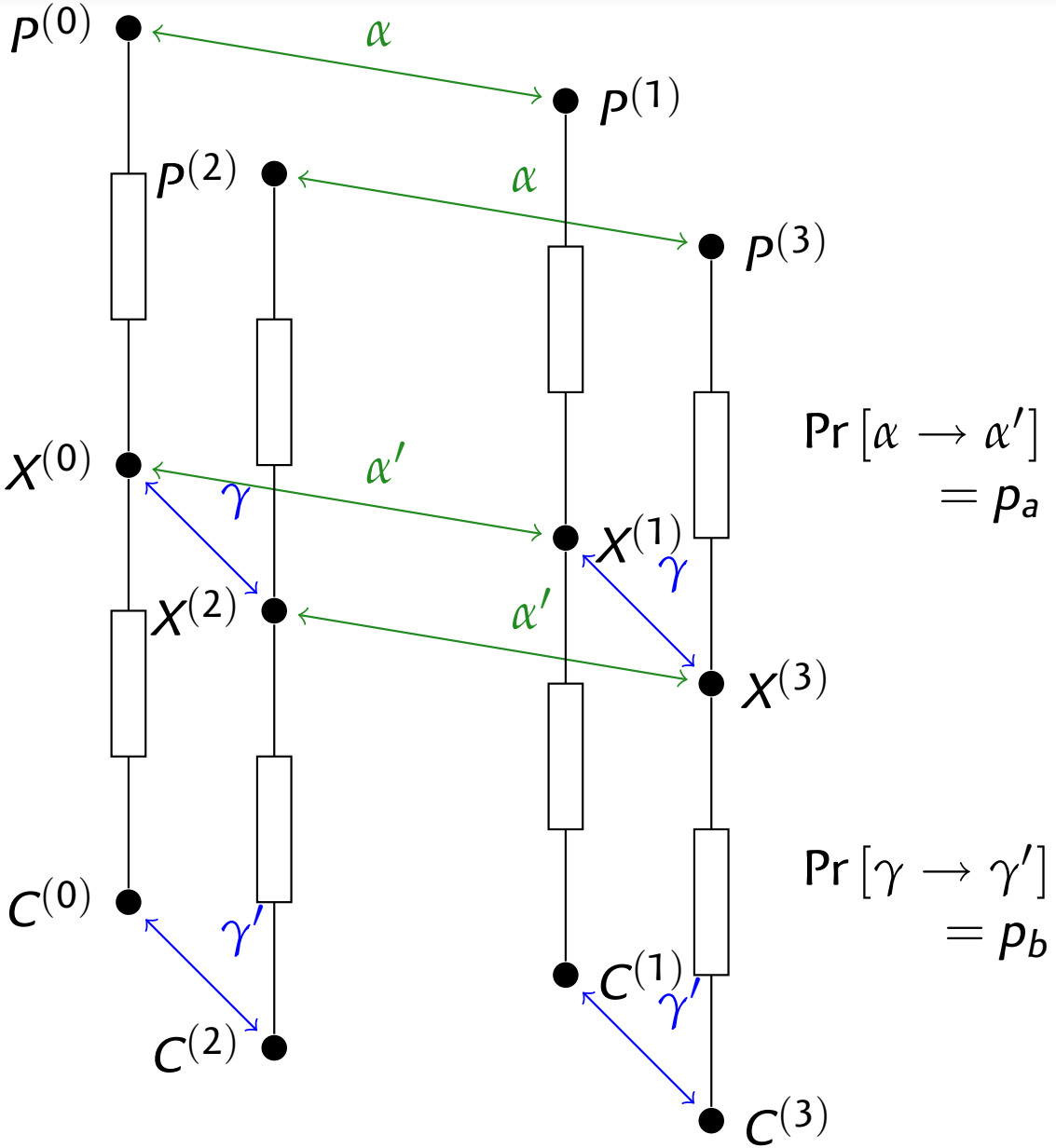
$$P^{(0)} \oplus P^{(1)} = \alpha$$

$$P^{(2)} \oplus P^{(3)} = \alpha$$

$$C^{(0)} \oplus C^{(1)} = \gamma'$$

$$C^{(2)} \oplus C^{(3)} = \gamma'$$

Boomerang Attacks



$$\Pr[\alpha \rightarrow \alpha'] = p_a$$

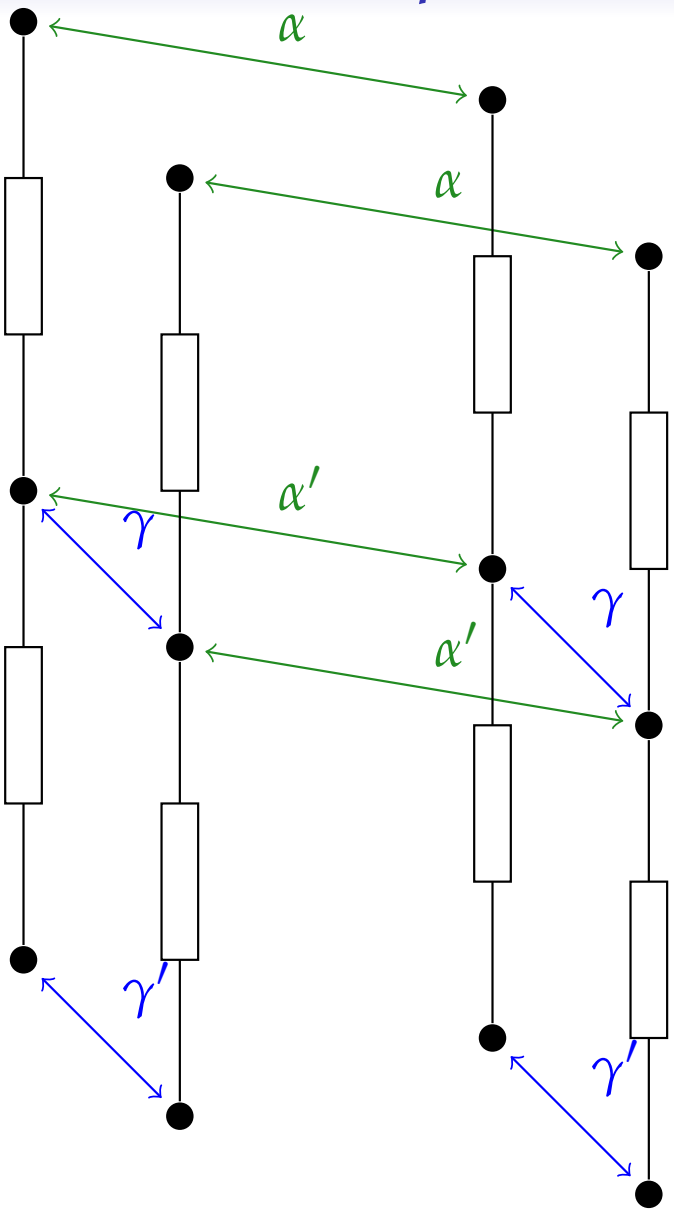
$$\Pr[\gamma \rightarrow \gamma'] = p_b$$

1. Start with $P^{(0)}, P^{(1)}$
2. Compute $C^{(0)}, C^{(1)}$
3. Build $C^{(2)}, C^{(3)}$
4. Compute $P^{(2)}, P^{(3)}$

$$C = \frac{1}{p_a} \frac{1}{p_b^2} \frac{1}{p_a}$$

$$P^{(0)} \oplus P^{(1)} = \alpha$$
$$P^{(2)} \oplus P^{(3)} = \alpha$$
$$C^{(0)} \oplus C^{(1)} = \gamma'$$
$$C^{(2)} \oplus C^{(3)} = \gamma'$$

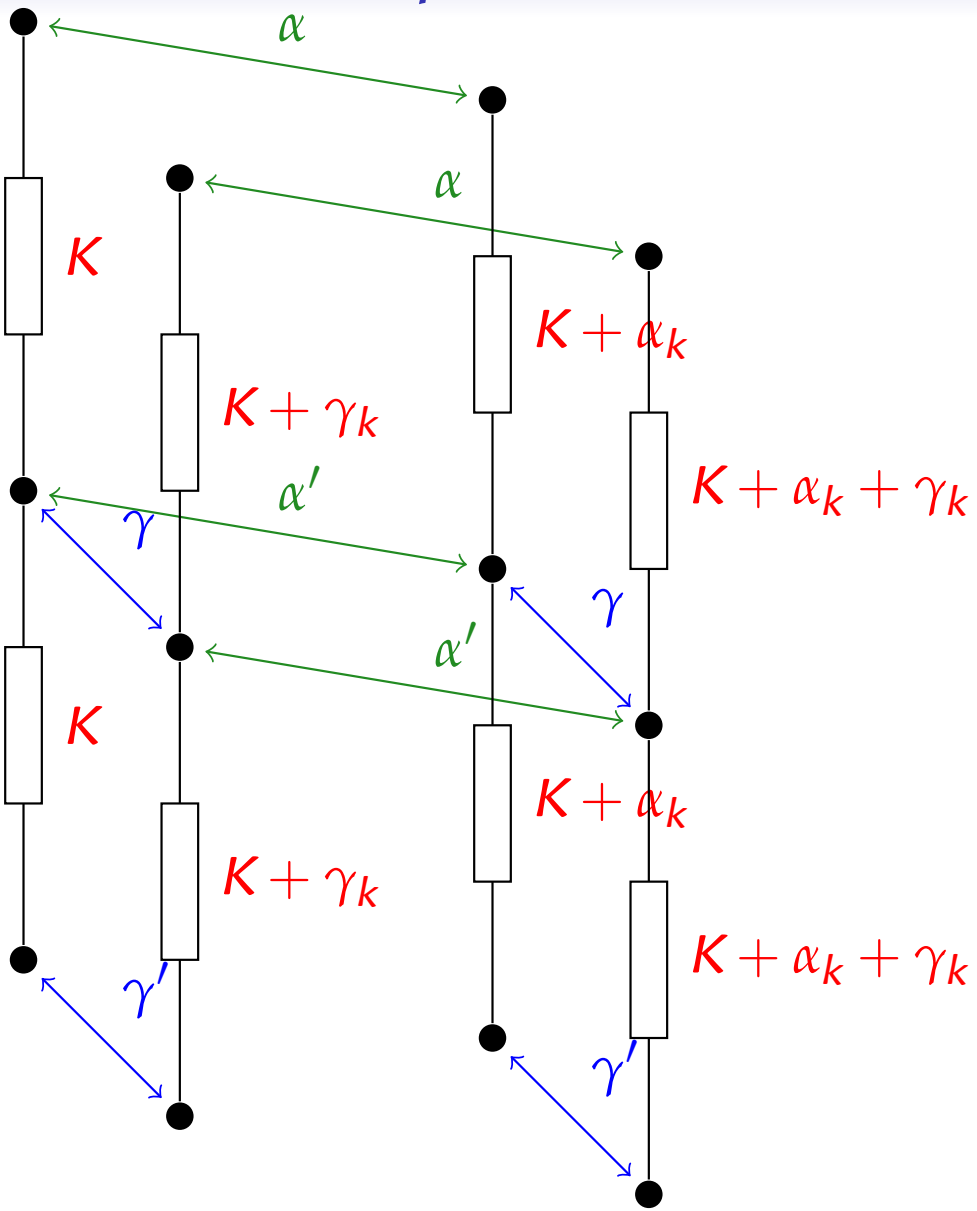
Improvements to the Boomerang Attack



1. Amplified probabilities
   - Do not specify α' and γ
   - $\hat{p}_a = \sqrt{\sum_{\alpha'} \Pr[\alpha \rightarrow \alpha']}$
     $\hat{p}_b = \sqrt{\sum_{\gamma} \Pr[\gamma \rightarrow \gamma']}$

2. Related-key
   - $p_a = \Pr\left[\alpha \xrightarrow{\alpha_k} \alpha'\right]$
     $p_b = \Pr\left[\gamma \xrightarrow{\gamma_k} \gamma'\right]$

Improvements to the Boomerang Attack



1. Amplified probabilities
   - Do not specify α' and γ
   - $\hat{p}_a = \sqrt{\sum_{\alpha'} \Pr[\alpha \rightarrow \alpha']}$
     $\hat{p}_b = \sqrt{\sum_{\gamma} \Pr[\gamma \rightarrow \gamma']}$

2. Related-key
   - $p_a = \Pr\left[\alpha \xrightarrow{\alpha_k} \alpha'\right]$
     $p_b = \Pr\left[\gamma \xrightarrow{\gamma_k} \gamma'\right]$

Boomerang in the Known-key Setting

- A boomerang attack gives a quartet:

$$P^{(0)} \oplus P^{(1)} = \alpha \qquad P^{(2)} \oplus P^{(3)} = \alpha \qquad \sum P^{(i)} = 0$$
$$C^{(0)} \oplus C^{(1)} = \gamma' \qquad C^{(2)} \oplus C^{(3)} = \gamma' \qquad \sum C^{(i)} = 0$$

- Even if the key is known, this is hard:
  - With fixed α, γ', complexity 2^n
  - With fixed α, and $\sum C^{(i)} = 0$, complexity $2^{n/2}$
  - $\sum P^{(i)} = 0, \sum C^{(i)} = 0$, best attack $2^{n/2}$, lower bound $2^{n/3}$

- With a known key, one can start from the middle
  - Message modification

Boomerang Attacks on Hash Functions

- A (related-key) boomerang attack gives a quartet:

$$\sum P^{(i)} = 0 \qquad \sum C^{(i)} = 0 \qquad \sum K^{(i)} = 0$$

- Most hash functions are based on a block cipher:

Davies-Meyer $f(h, m) = E_m(h) \oplus h$

Matyas-Meyer-Oseas $f(h, m) = E_h(m) \oplus m$

- The boomerang quartet gives:

$$\sum h^{(i)} = 0 \qquad \sum m^{(i)} = 0 \qquad \sum f(h^{(i)}, m^{(i)}) = 0$$

DM	$h^{(i)} = P^{(i)}$	$m^{(i)} = K^{(i)}$	$f(h^{(i)}, m^{(i)}) = P^{(i)} \oplus C^{(i)}$
MMO	$h^{(i)} = K^{(i)}$	$m^{(i)} = P^{(i)}$	$f(h^{(i)}, m^{(i)}) = P^{(i)} \oplus C^{(i)}$

- In general this is hard:
  - $\sum f(h, m) = 0$, best attack $2^{n/3}$, lower bound $2^{n/4}$
  - $\sum f(h, m) = \sum h = \sum m = 0$, best attack $2^{n/2}$, lower bound $2^{n/3}$

New Technique: Using Auxiliary Paths

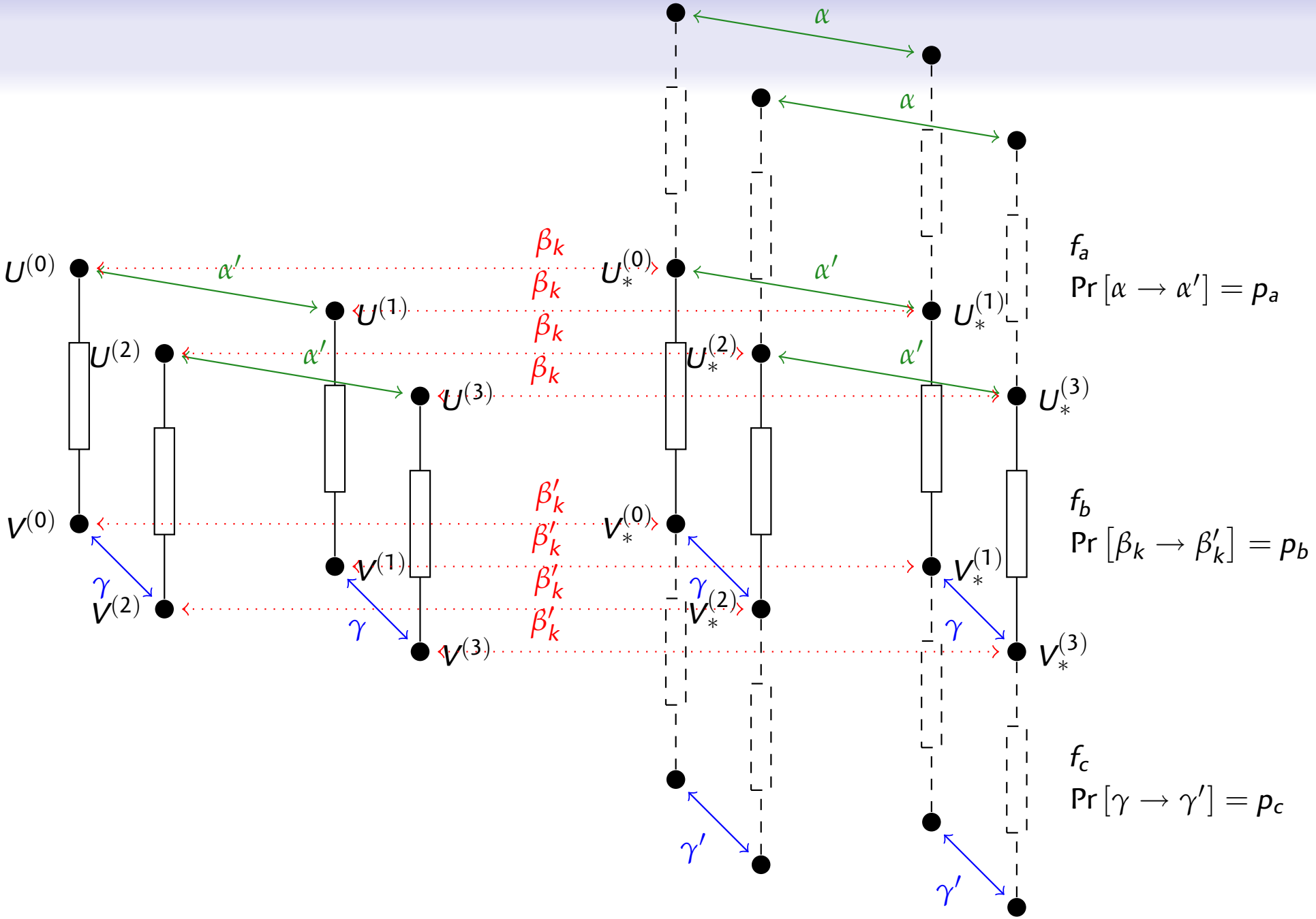
- Divide f in three sub-functions: $f = f_c \circ f_b \circ f_a$
  - for f_a, $\alpha \rightarrow \alpha'$ with probability p_a
  - for f_b, $\beta_j \rightarrow \beta'_j$ with probability p_b
  - for f_c, $\gamma \rightarrow \gamma'$ with probability p_c

- Start with a boomerang quartet for f_b:

$$U^{(1)} = U^{(0)} + \alpha' \qquad U^{(3)} = U^{(2)} + \alpha'$$
$$V^{(2)} = V^{(0)} + \gamma \qquad V^{(2)} = V^{(1)} + \gamma$$

- Construct $U_*^{(i)} = U^{(i)} + \beta_j$.
  With probability p_b^4, $V_*^{(i)} = V^{(i)} + \beta'_j$:

$$U_*^{(1)} = U_*^{(0)} + \alpha' \qquad U_*^{(3)} = U_*^{(2)} + \alpha'$$
$$V_*^{(2)} = V_*^{(0)} + \gamma \qquad V_*^{(2)} = V_*^{(1)} + \gamma$$


α
α
$U^{(0)}$
α'
β_k
β_k
$U^{(1)}$
β_k
$U^{(2)}$
α'
β_k
$U^{(3)}$
$U_*^{(0)}$
α'
$U_*^{(1)}$
$U_*^{(2)}$
α'
$U_*^{(3)}$
f_a
$\Pr[\alpha \to \alpha'] = p_a$
$V^{(0)}$
β'_k
β'_k
$V^{(1)}$
β'_k
$V^{(2)}$
β'_k
$V^{(3)}$
$V_*^{(0)}$
$V_*^{(1)}$
$V_*^{(2)}$
$V_*^{(3)}$
f_b
$\Pr[\beta_k \to \beta'_k] = p_b$
γ
γ
γ
γ
γ'
γ'
f_c
$\Pr[\gamma \to \gamma'] = p_c$

New Technique: Using Auxiliary Paths

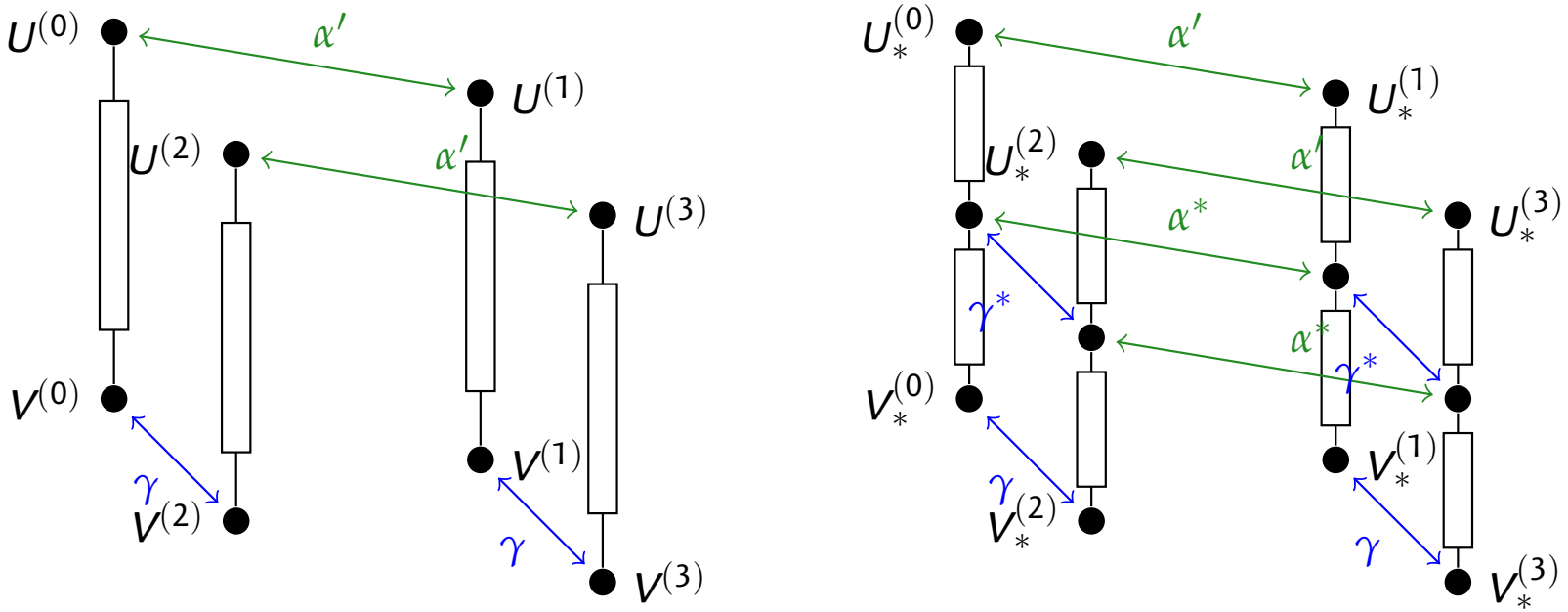
- Also with related-key paths

- Similar to "Boomerang" of Joux and Peyrin
- Similar to message modifications for Boomerang attacks
  - Blake [BNR '11]
  - SHA-2 [ML '11]
  - HAVAL [Sasaki '11]
  - Skein/Threefish [ACMPV '09, Chen & Jia '10]

- Complexity: $\frac{1}{p_a^2 p_c^2}\left(\frac{C}{b \cdot p_b^4} + 1\right)$
  - Cost C to build an initil quartet
  - b paths with probability p_b for f_b

- Very efficient with a large family of probability 1 paths

Building the Initial Quartet

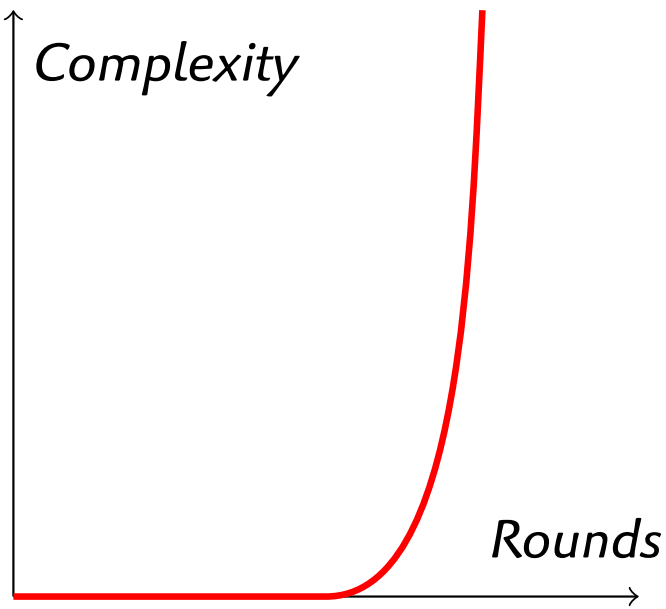


- Usually by extending the top and bottom paths
- Special tricks depending on the function...

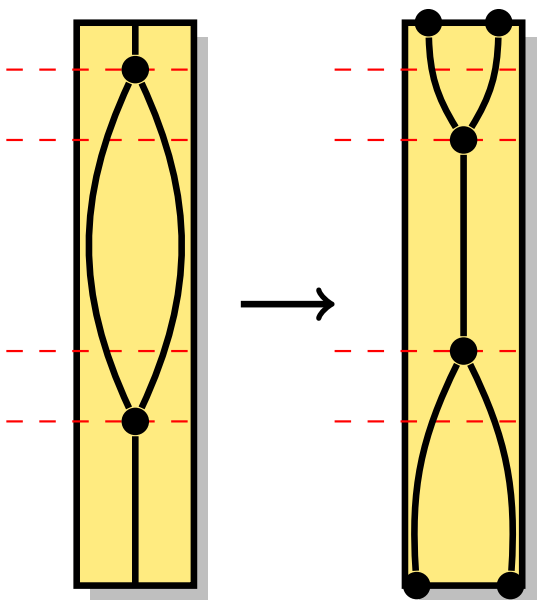
Application to ARX designs

- Several recent design are based on the ARX design
  - Use only Addition, Rotation, Xor
  - Skein, Blake are SHA-3 finalists

- Short RK paths with high probability



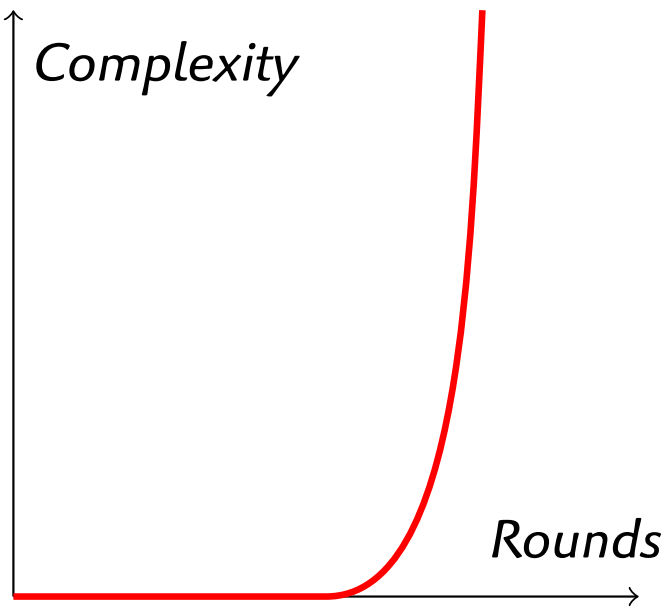
- Hard to build controlled characteristics



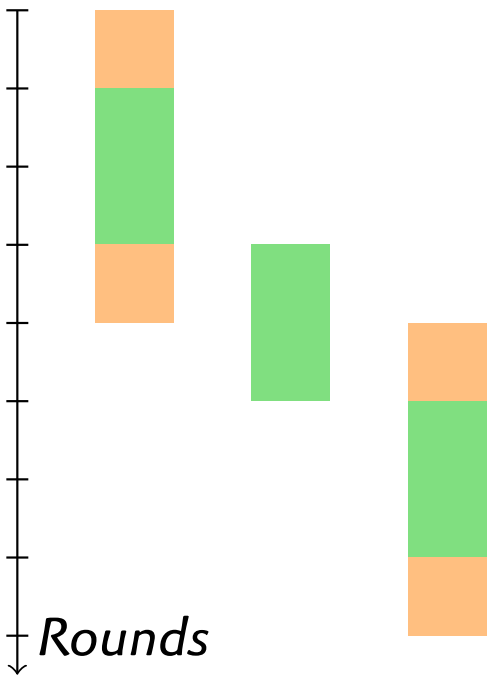
Application to ARX designs

- Several recent design are based on the ARX design
  - Use only Addition, Rotation, Xor
  - Skein, Blake are SHA-3 finalists

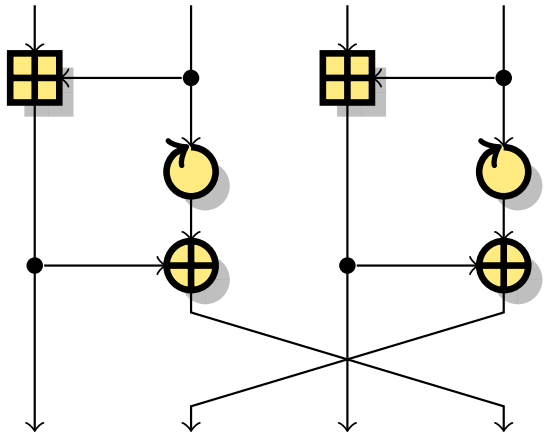
- Short RK paths with high probability



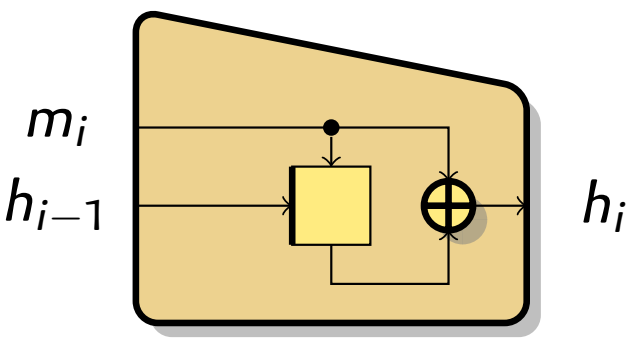
- Using auxiliary paths



Skein



Threefish-256 round



MMO mode

- SHA-3 finalist
- ARX design
  - 64-bit words
  - $\texttt{MIX}(a, b) :=$
    $\Big((a + b) \bmod 2^{64}, (b \lll R) \oplus c\Big)$
  - Word permutations
  - Key addition every four rounds
- Threefish-256:
  - 256-bit key: K_0, K_1, K_2, K_3
  - 128-bit tweak: T_0, T_1
  - 256-bit text

Skein: Differential Trails

Key schedule (Threefish-256):

- 256-bit key: K_0, K_1, K_2, K_3
- 128-bit tweak: T_0, T_1
- $K_4 := K_0 \oplus K_1 \oplus K_2 \oplus K_3 \oplus C$
- $T_2 := T_0 \oplus T_1$

Round				
0	K_0	$K_1 + T_0$	$K_2 + T_1$	$K_3 + 0$
4	K_1	$K_2 + T_1$	$K_3 + T_2$	$K_4 + 1$
8	K_2	$K_3 + T_2$	$K_4 + T_0$	$K_0 + 2$
12	K_3	$K_4 + T_0$	$K_0 + T_1$	$K_1 + 3$
16	K_4	$K_0 + T_1$	$K_1 + T_2$	$K_2 + 4$

16-round trail:

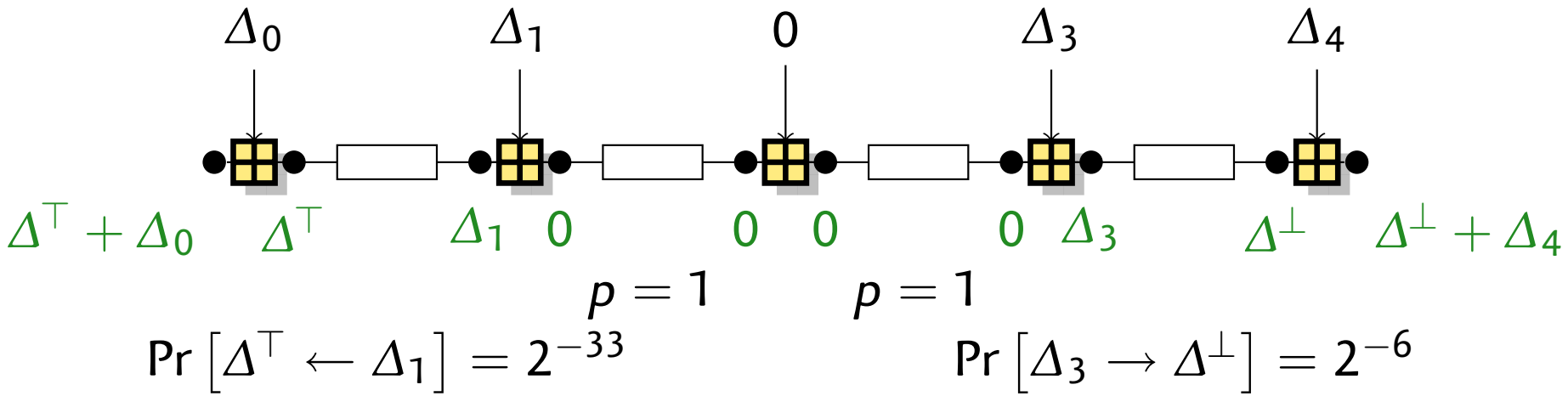
Skein: Differential Trails

Key schedule (Threefish-256):

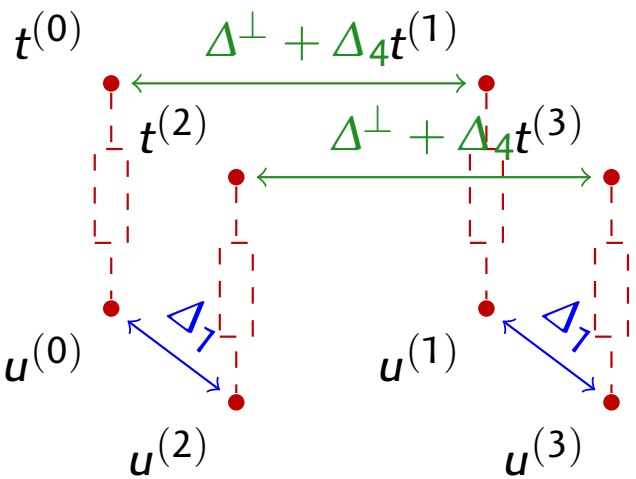
- 256-bit key: K_0, K_1, K_2, K_3
- 128-bit tweak: T_0, T_1
- $K_4 := K_0 \oplus K_1 \oplus K_2 \oplus K_3 \oplus C$
- $T_2 := T_0 \oplus T_1$

Round				
0	K_0	$K_1 + T_0$	$K_2 + T_1$	$K_3 + 0$
4	K_1	$K_2 + T_1$	$K_3 + T_2$	$K_4 + 1$
8	K_2	$K_3 + T_2$	$K_4 + T_0$	$K_0 + 2$
12	K_3	$K_4 + T_0$	$K_0 + T_1$	$K_1 + 3$
16	K_4	$K_0 + T_1$	$K_1 + T_2$	$K_2 + 4$

16-round trail:



Skein: Description of the Attack

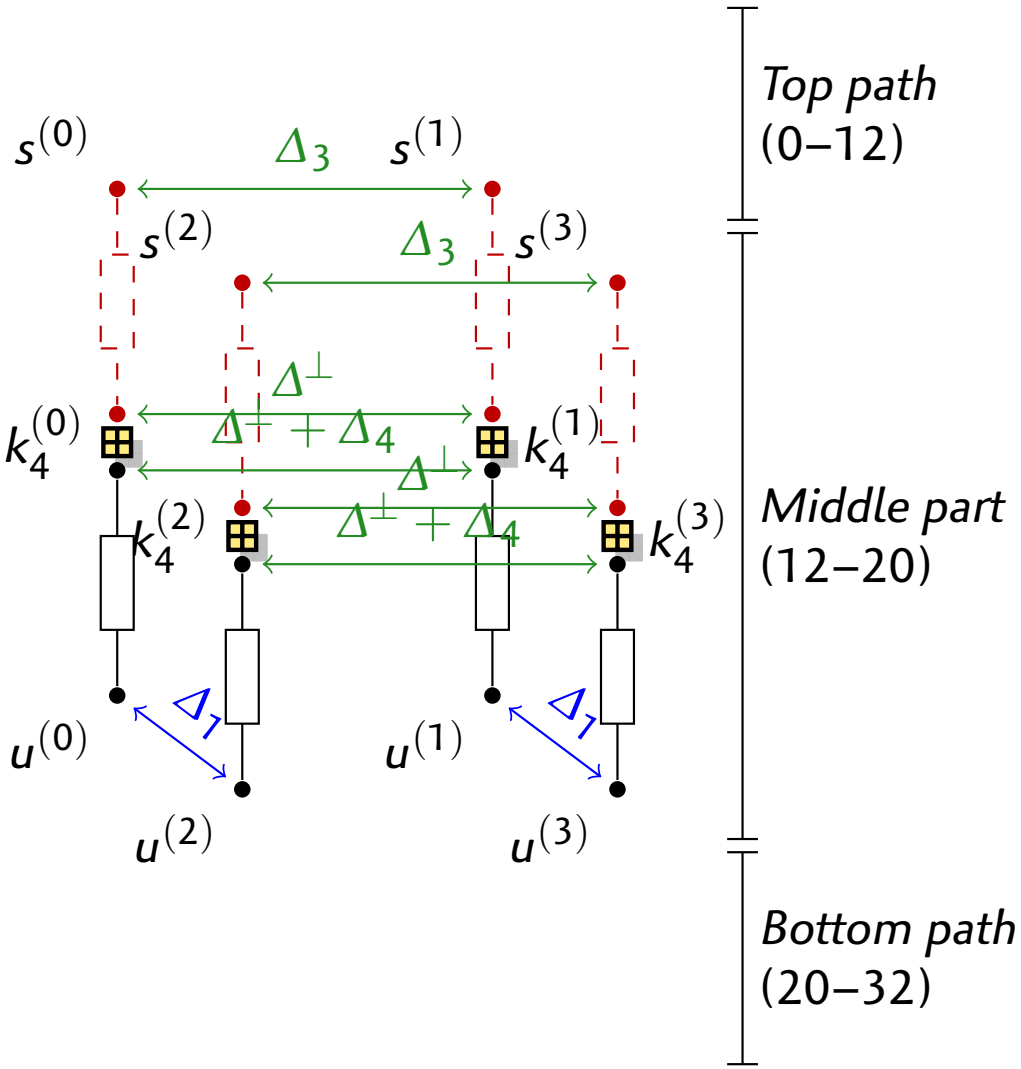


Top path
(0–12)

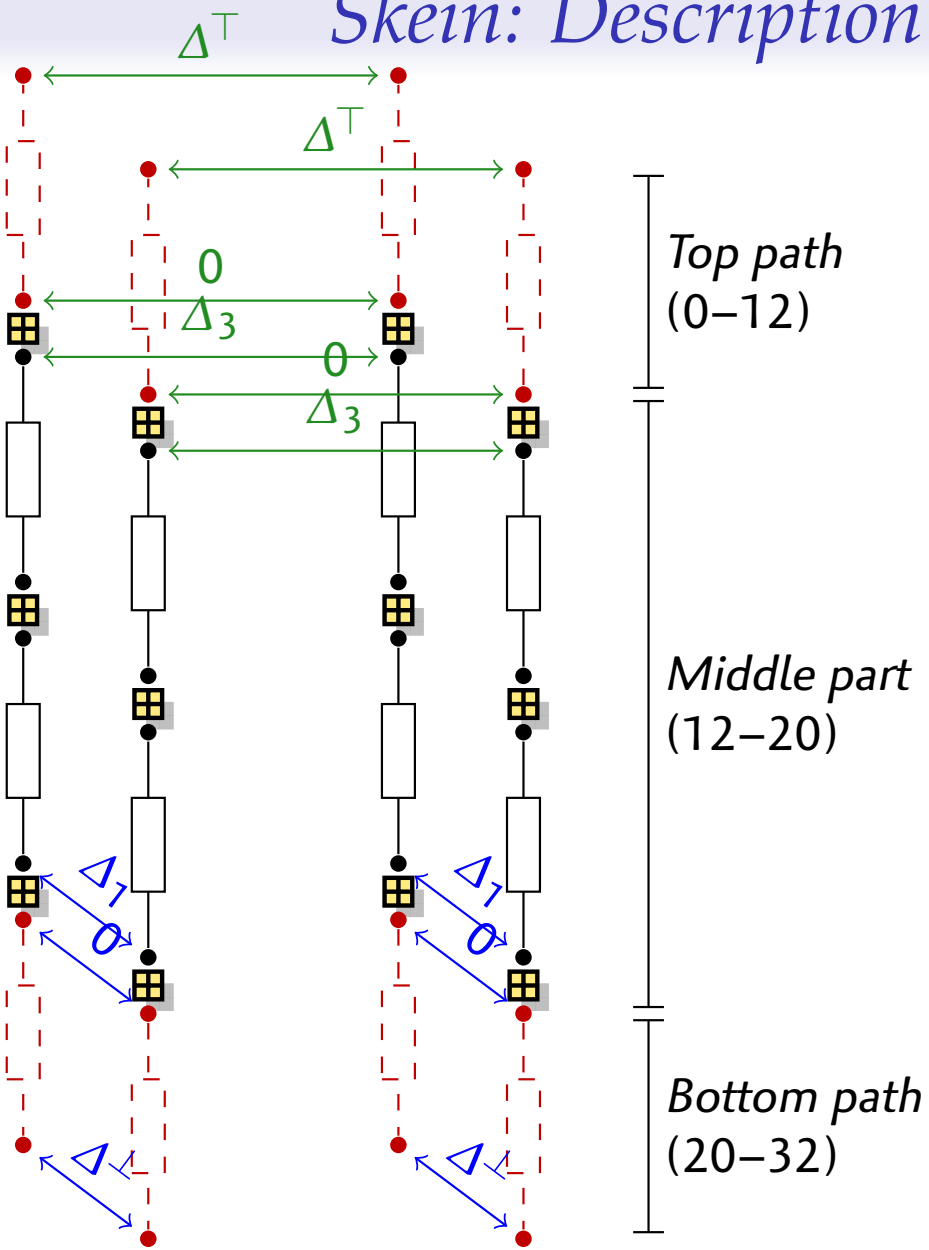
Middle part
(12–20)

Bottom path
(20–32)

Skein: Description of the Attack



Skein: Description of the Attack

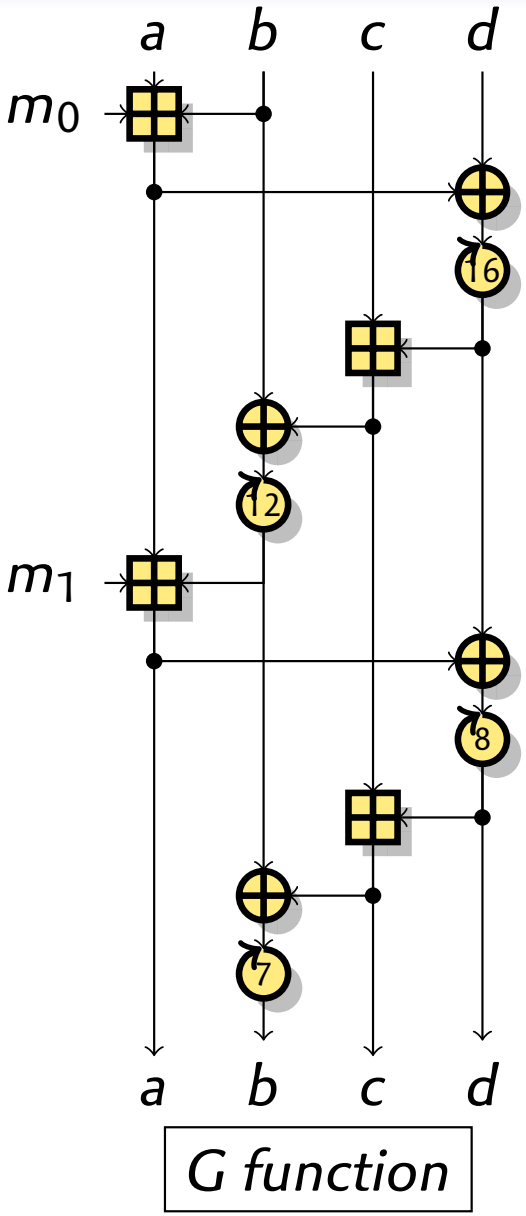


Skein: Results

Attack	CF/KP	Rounds	CF/KP calls	Ref.
Unknown Key				
Near collisions (Skein-256)	CF	24	2^{60}	[CANS '10]
Boomerang dist. (Threefish-512)	KP	32	2^{189}	[ISPEC '10] [1]
Key Recovery (Threefish-512)	KP	34	$2^{474.4}$	[ISPEC '10] [1]
Key Recovery (Threefish-512)	KP	32	2^{312}	[AC '09]
Open key				
Boomerang dist. (Threefish-512)	KP	35	2^{478}	[AC '09]
Near collisions (Skein-256)	CF	32	2^{105}	[ePrint '11] [1]
Boomerang dist. (Skein-256)	CF and KP	24	2^{18}	
Boomerang dist. (Threefish-256)	KP	28	2^{21}	
Boomerang dist. (Skein-256)	CF	28	2^{24}	
Boomerang dist. (Threefish-256)	KP	32	2^{57}	
Boomerang dist. (Skein-256)	CF	32	2^{114}	

[1] *problems with paths*

Blake



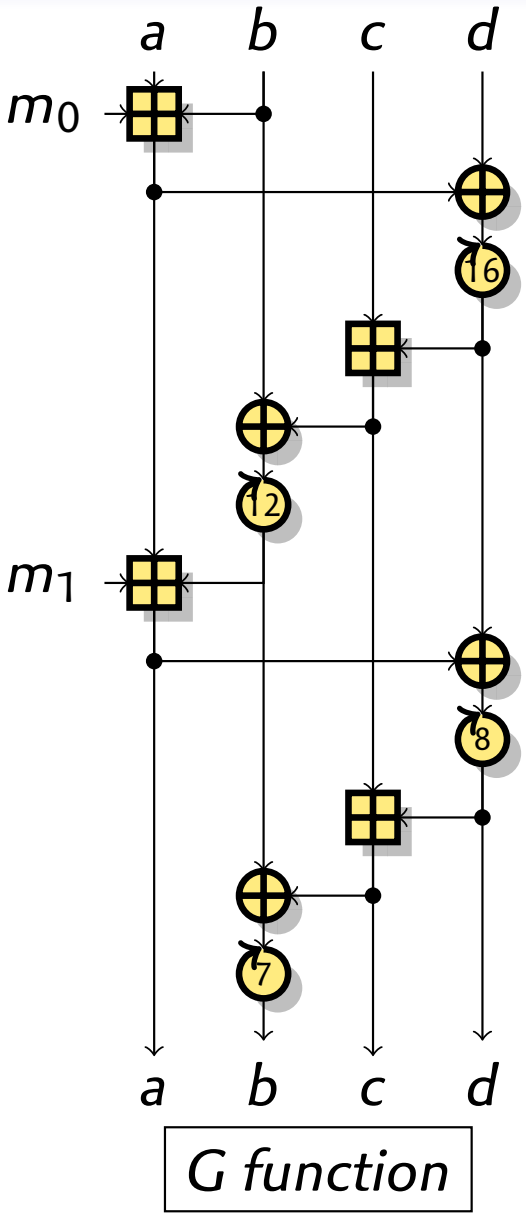
G function

- State is 4×4 matrix:

a_0	a_1	a_2	a_3
b_0	b_1	b_2	b_3
c_0	c_1	c_2	c_3
d_0	d_1	d_2	d_3

- Column step:
  $G(a_0, b_0, c_0, d_0)$
  $G(a_1, b_1, c_1, d_1)$
  $G(a_2, b_2, c_2, d_2)$
  $G(a_3, b_3, c_3, d_3)$
- Diagonal step:
  $G(a_0, b_1, c_2, d_3)$
  $G(a_1, b_2, c_3, d_0)$
  $G(a_2, b_3, c_0, d_1)$
  $G(a_3, b_0, c_1, d_2)$

Blake



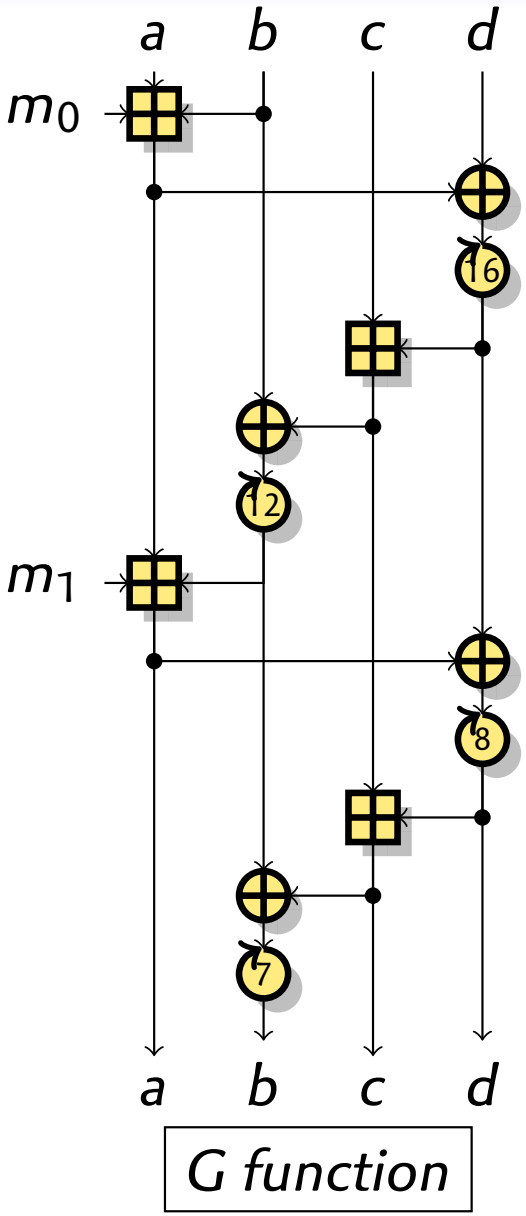
G function

- State is 4×4 matrix:

a_0	a_1	a_2	a_3
b_0	b_1	b_2	b_3
c_0	c_1	c_2	c_3
d_0	d_1	d_2	d_3

- Column step:
  $G(a_0, b_0, c_0, d_0)$
  $G(a_1, b_1, c_1, d_1)$
  $G(a_2, b_2, c_2, d_2)$
  $G(a_3, b_3, c_3, d_3)$
- Diagonal step:
  $G(a_0, b_1, c_2, d_3)$
  $G(a_1, b_2, c_3, d_0)$
  $G(a_2, b_3, c_0, d_1)$
  $G(a_3, b_0, c_1, d_2)$

Blake



G function

- State is 4×4 matrix:

a_0	a_1	a_2	a_3
b_0	b_1	b_2	b_3
c_0	c_1	c_2	c_3
d_0	d_1	d_2	d_3

- Column step:
  $G(a_0, b_0, c_0, d_0)$
  $G(a_1, b_1, c_1, d_1)$
  $G(a_2, b_2, c_2, d_2)$
  $G(a_3, b_3, c_3, d_3)$
- Diagonal step:
  $G(a_0, b_1, c_2, d_3)$
  $G(a_1, b_2, c_3, d_0)$
  $G(a_2, b_3, c_0, d_1)$
  $G(a_3, b_0, c_1, d_2)$

Blake: Differential Trails

- Key schedule: permutation based

σ_3 :	7	9	3	1	13	12	11	14	2	6	5	10	4	0	15	8
σ_4 :	9	0	5	7	2	4	10	15	14	1	11	12	6	8	3	13

- Choose a message word used
  - at the beginning of a round
  - at the end of the next round

- 4-round trail:

Blake: Differential Trails

- Key schedule: permutation based

σ_3 :	7	9	3	1	13	12	11	14	2	6	5	10	4	0	15	8
σ_4 :	9	0	5	7	2	4	10	15	14	1	11	12	6	8	3	13

- Choose a message word used
  - at the beginning of a round
  - at the end of the next round

- 4-round trail:

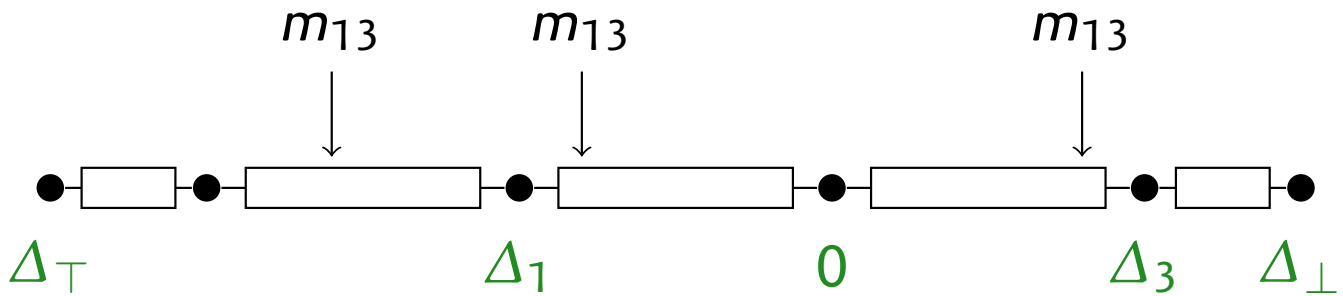
Blake: Differential Trails

- Key schedule: permutation based

σ_3 :	7	9	3	1	13	12	11	14	2	6	5	10	4	0	15	8
σ_4 :	9	0	5	7	2	4	10	15	14	1	11	12	6	8	3	13

- Choose a message word used
  - at the beginning of a round
  - at the end of the next round

- 4-round trail:



$p = 1$ $p = 1/2$

$$\Pr\left[\Delta^\top \leftarrow \Delta_1\right] = 2^{-6} \cdot 2^{-42} \qquad \Pr\left[\Delta_3 \rightarrow \Delta^\perp\right] = 2^{-24}$$

Blake: Description of the Attack

The hard part is the middle round

- Column step is part of the top path
- Diagonal step is part of the bottom path

1. Find (state, message) candidates for each diagonal G function
   - Start with middle quartets with all differences fixed
2. Look for combinations of candidates that follow the first part of the diagonal step
   - Use the message to randomize
3. Look for candidates that follow the full diagonal step
   - Use the message to randomize

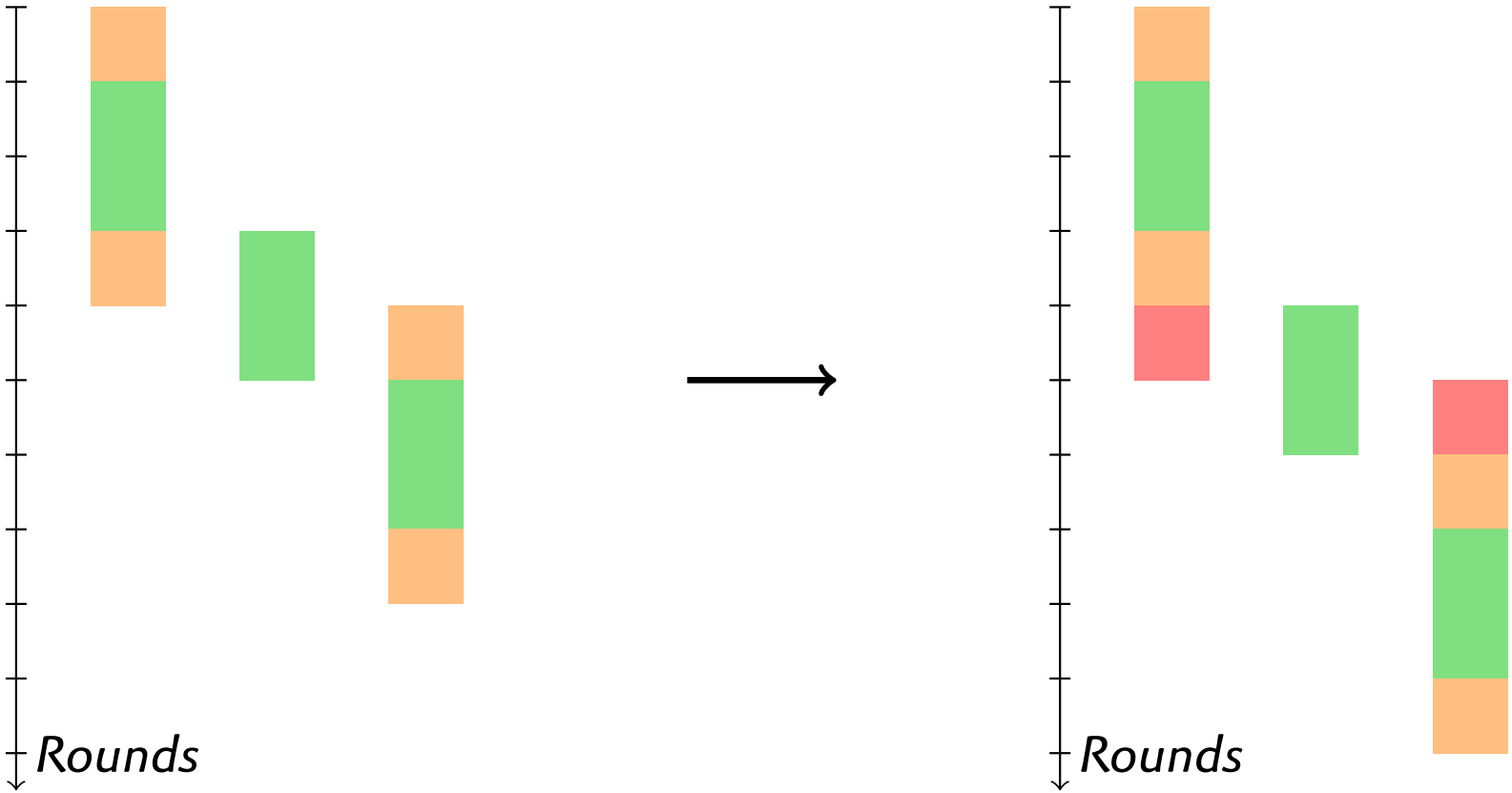
Blake-256: Results

Attack	CF/KP	Rounds	CF/KP calls	Ref.
Unknown Key				
Boomerang dist.	KP	7	2^{122}	[FSE '11]
Boomerang dist.	KP	8	2^{242}	[FSE '11] [1]
Open Key				
Boomerang dist.	CF w/ Init	7	2^{232}	[FSE '11]
Boomerang dist.	KP	7	2^{32}	
Boomerang dist.	KP	8	2^{1xx}	

[1] *problems with paths*

Limitations of the Technique

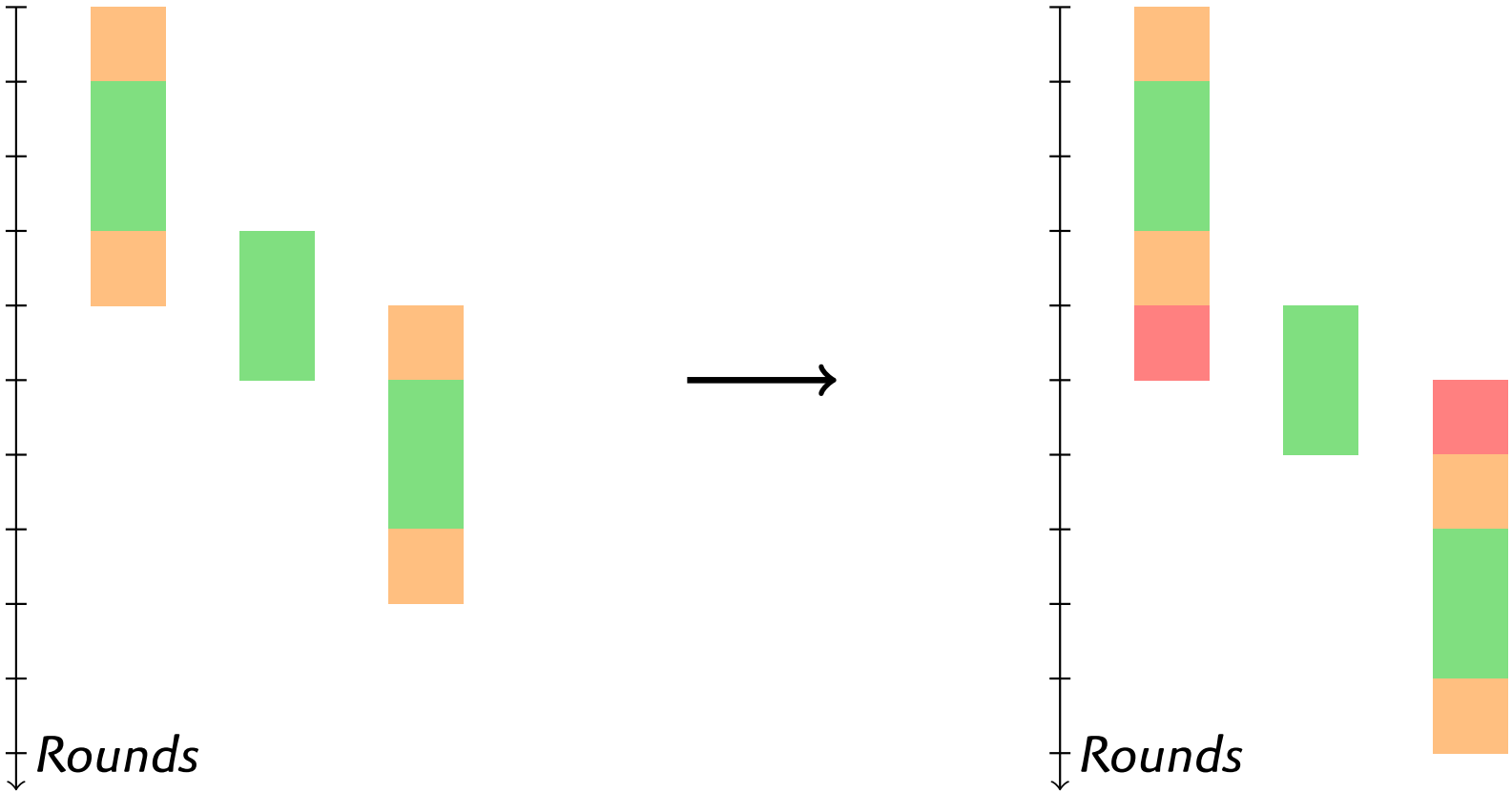
Why not attack more rounds?



Paths are incompatible!

Limitations of the Technique

Why not attack more rounds?



Paths are incompatible!

Incompatibilities in Boomerang Paths

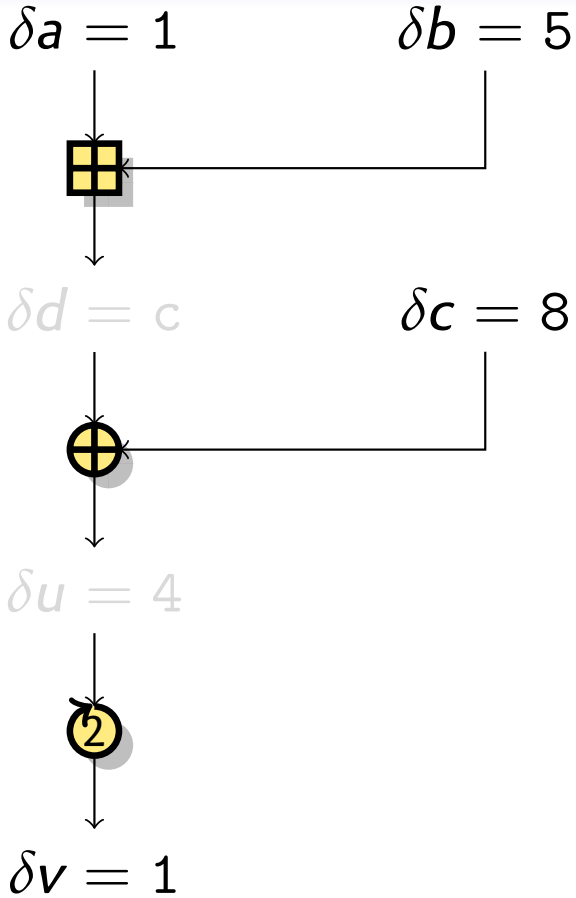
- For a Boomerang attack, we usually assume that the path are independent
- We are building a quartet $X^{(0)}, X^{(1)}, X^{(2)}, X^{(3)}$:

$$X^{(1)} = X^{(0)} + \alpha' \qquad X^{(3)} = X^{(2)} + \alpha'$$
$$X^{(2)} = X^{(0)} + \gamma \qquad X^{(2)} = X^{(1)} + \gamma$$

$$(X^{(0)}, X^{(1)}) \xrightarrow{f_a^{(-1)}} \alpha \qquad (X^{(2)}, X^{(3)}) \xrightarrow{f_a^{(-1)}} \alpha$$
$$(X^{(0)}, X^{(2)}) \xrightarrow{f_b} \gamma' \qquad (X^{(1)}, X^{(3)}) \xrightarrow{f_b} \gamma'$$

- But these events are not independent! [Murphy 2011]
- To explain this behaviour, we have to study differential paths

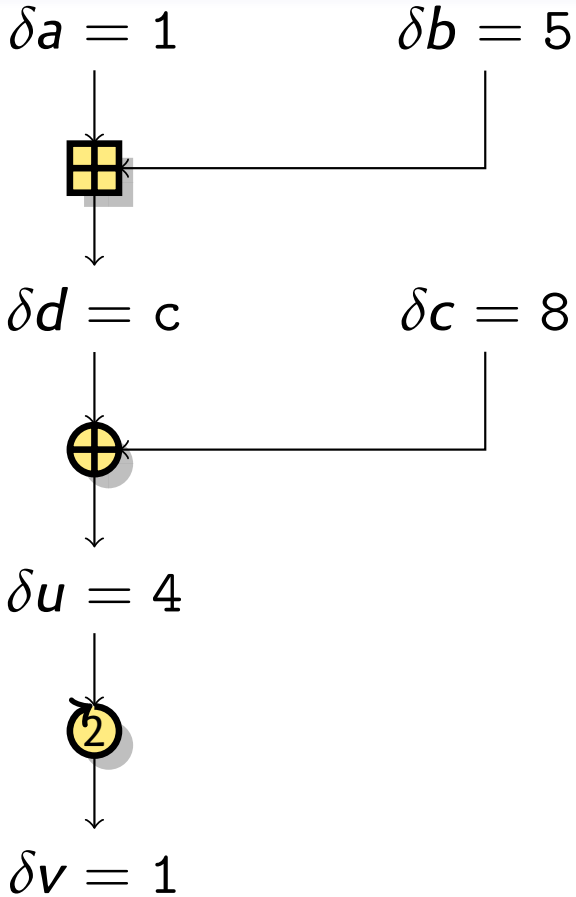
Differential Characteristic



$$c = a + b$$
$$u = c + d$$
$$v = u \lll 2$$

- Choose a difference operation: $\oplus$
- A differential only specifies the input and output difference
- A difference characteristic specifies the difference of each internal variable
  - Compute probability for each operation
- Wang introduced a signed difference.
  - A path defines a set of good pairs: $\{x, x' : x' = x \oplus \alpha\}$
  - We want to capture: $\{x, x' : x' = x \oplus \alpha \text{ and } x' = x \boxplus \beta\}$
  - Sign each active bit

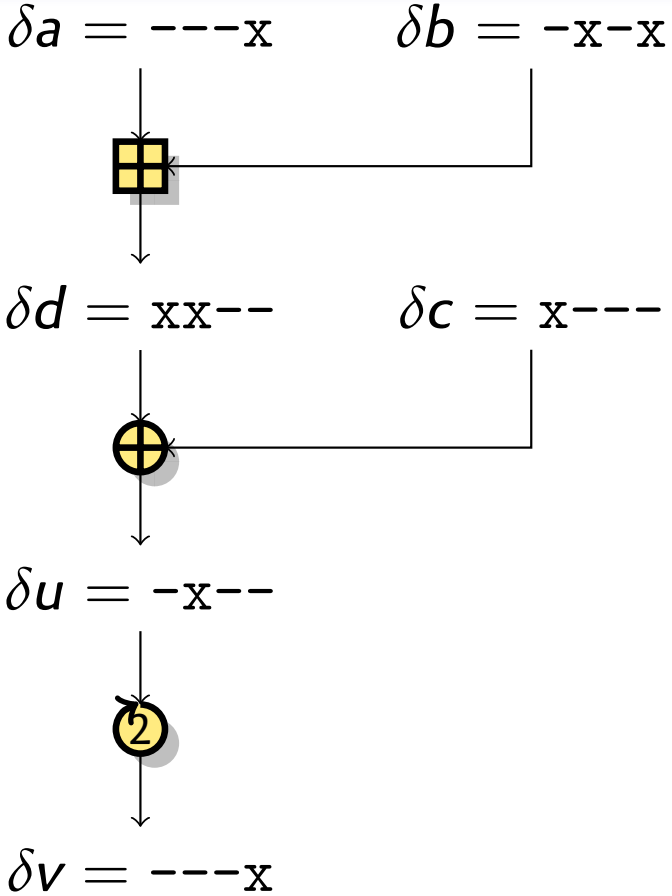
Differential Characteristic



$$c = a + b$$
$$u = c + d$$
$$v = u \lll 2$$

- Choose a difference operation: $\oplus$
- A differential only specifies the input and output difference
- A difference characteristic specifies the difference of each internal variable
  - Compute probability for each operation
- Wang introduced a signed difference.
  - A path defines a set of good pairs: $\{x, x' : x' = x \oplus \alpha\}$
  - We want to capture: $\{x, x' : x' = x \oplus \alpha \text{ and } x' = x \boxplus \beta\}$
  - Sign each active bit

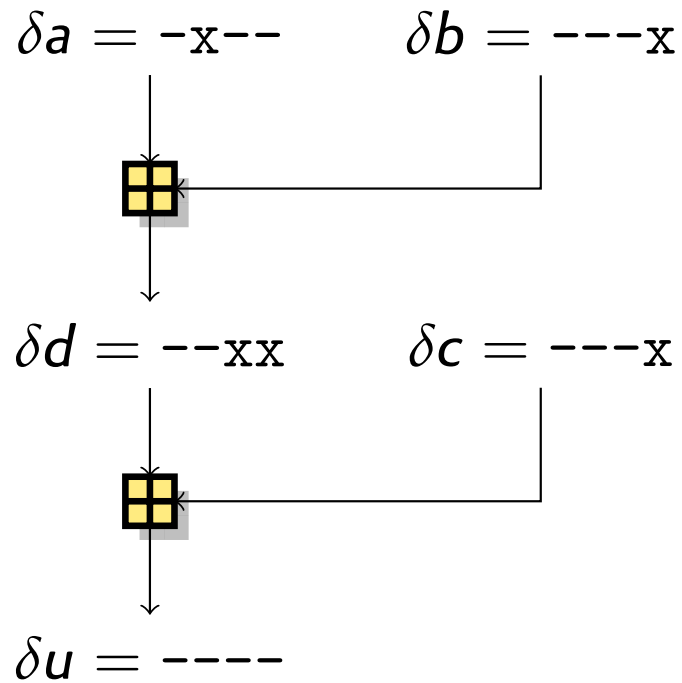
Differential Characteristic



$$c = a + b$$
$$u = c + d$$
$$v = u \lll 2$$

- Choose a difference operation: $\oplus$
- A differential only specifies the input and output difference
- A difference characteristic specifies the difference of each internal variable
  - Compute probability for each operation
- Wang introduced a signed difference.
  - A path defines a set of good pairs: $\{x, x' : x' = x \oplus \alpha\}$
  - We want to capture: $\{x, x' : x' = x \oplus \alpha \text{ and } x' = x \boxplus \beta\}$
  - Sign each active bit

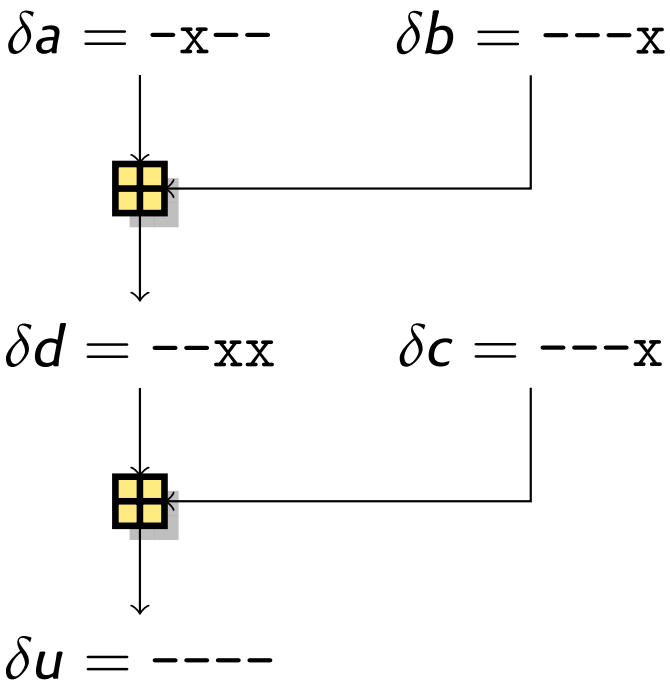
Problems with Xor-Characteristics



$$\boxed{\begin{aligned} c &= a + b \\ u &= c + d \end{aligned}}$$

- Probability: $2^{-3} \cdot 2^{-2}$
- Obviously wrong if you consider modular differences
  - $\delta a \rightsquigarrow \pm 4$
  - $\delta b \rightsquigarrow \pm 1$
  - $\delta c \rightsquigarrow \pm 1$

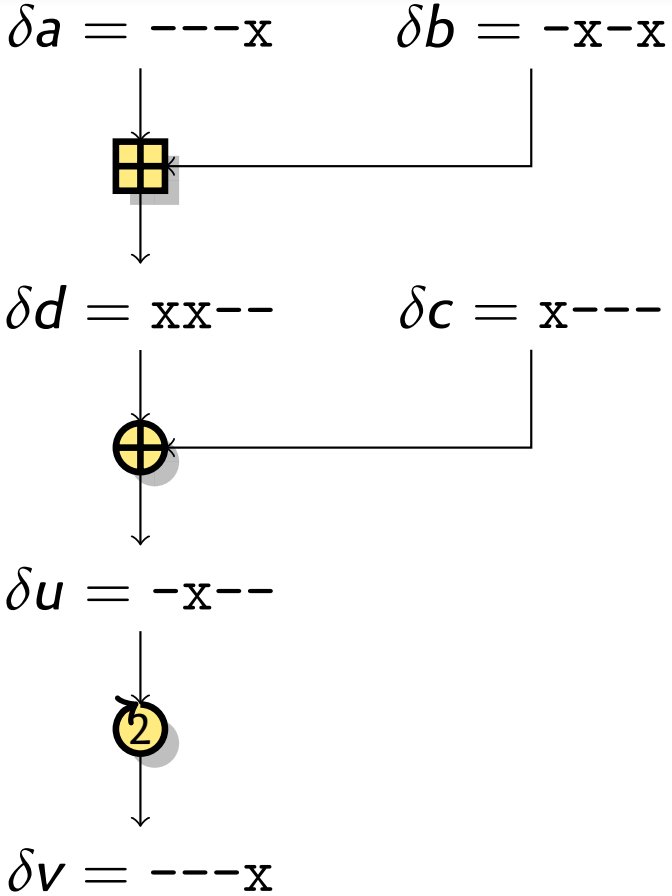
Problems with Xor-Characteristics



$$\boxed{\begin{aligned} c &= a + b \\ u &= c + d \end{aligned}}$$

- Probability: $2^{-3} \cdot 2^{-2}$
- Obviously wrong if you consider modular differences
  - $\delta a \rightsquigarrow \pm 4$
  - $\delta b \rightsquigarrow \pm 1$
  - $\delta c \rightsquigarrow \pm 1$

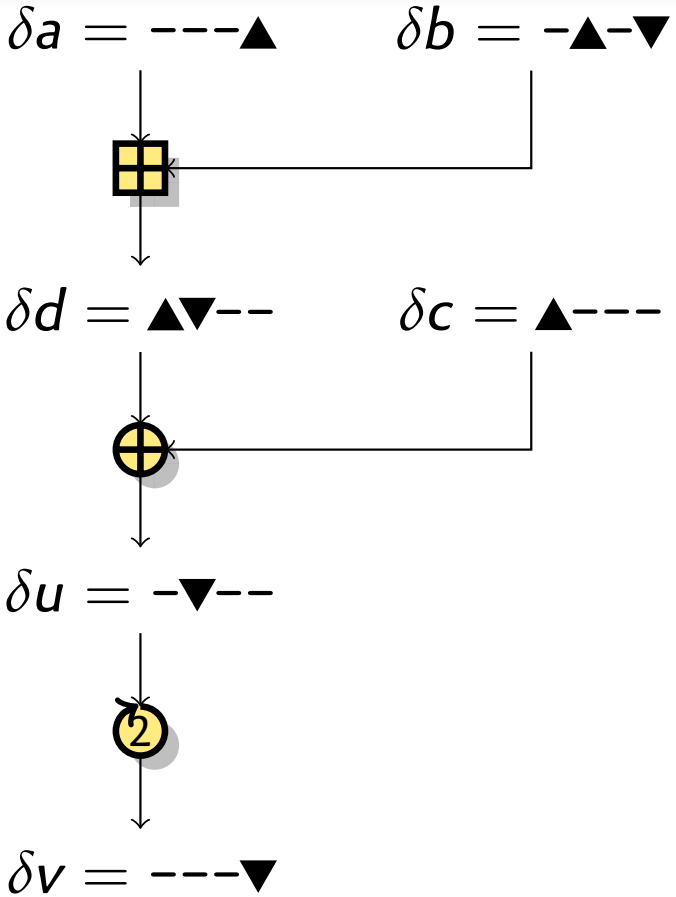
Differential Characteristic



$$c = a + b$$
$$u = c + d$$
$$v = u \lll 2$$

- Choose a difference operation: $\oplus$
- A differential only specifies the input and output difference
- A difference characteristic specifies the difference of each internal variable
  - Compute probability for each operation
- Wang introduced a signed difference.
  - A path defines a set of good pairs: $\{x, x' : x' = x \oplus \alpha\}$
  - We want to capture: $\{x, x' : x' = x \oplus \alpha \text{ and } x' = x \boxplus \beta\}$
  - Sign each active bit

Differential Characteristic



$$c = a + b$$
$$u = c + d$$
$$v = u \lll 2$$

- Choose a difference operation: $\oplus$
- A differential only specifies the input and output difference
- A difference characteristic specifies the difference of each internal variable
  - Compute probability for each operation
- Wang introduced a signed difference.
  - A path defines a set of good pairs: $\{x, x' : x' = x \oplus \alpha\}$
  - We want to capture: $\{x, x' : x' = x \oplus \alpha \text{ and } x' = x \boxplus \beta\}$
  - Sign each active bit

Generalized constraints [De Cannière & Rechberger 06]

	(x, x'):	$(0,0)$	$(0,1)$	$(1,0)$	$(1,1)$
?	*anything*	✓	✓	✓	✓
-	$x = x'$	✓	-	-	✓
x	$x \neq x'$	-	✓	✓	-
0	$x = x' = 0$	✓	-	-	-
u	$(x, x') = (0,1)$	-	✓	-	-
n	$(x, x') = (1,0)$	-	-	✓	-
1	$x = x' = 0$	-	-	-	✓
#	*incompatible*	-	-	-	-
3	$x = 0$	✓	✓	-	-
5	$x' = 0$	✓	-	✓	-
7		✓	✓	✓	-
A	$x' = 1$	-	✓	-	✓
B		✓	✓	-	✓
C	$x = 1$	-	-	✓	✓
D		✓	-	✓	✓
E		-	✓	✓	✓

S-Systems

Definition

T-function $\forall t$, t bits of the output can be computed from t bits of the input.

S-function *There exist a state S so that:* $\forall t$, bit t of the output and state $S[t]$ can be computed from bit t of the input, and the state $S[t-1]$.

S-system $f(P, x) = 0$
f is an S-function, P is a parameter, x is an unknown

- Addition, Xor, and Boolean operations are S-functions

Solving S-Systems

Important Example

$$x \oplus \Delta = x \boxplus \delta$$

- On average one solution
- Easy to solve because it's a T-function.
  - Guess LSB, check, and move to next bit
- How easy exactly?
- Backtracking is exponential in the worst case:
  $x \oplus 0x80000000 = x$
- For random δ, Δ, most of the time the system is inconsistent

Solving S-Systems

Important Example

$$x \oplus \Delta = x \boxplus \delta$$

- On average one solution
- Easy to solve because it's a T-function.
  - Guess LSB, check, and move to next bit
- How easy exactly?
- Backtracking is exponential in the worst case:
  $x \oplus 0x80000000 = x$
- For random δ, Δ, most of the time the system is inconsistent

Solving S-Systems

Important Example

$$x \oplus \Delta = x \boxplus \delta$$

- On average one solution
- Easy to solve because it's a T-function.
  - Guess LSB, check, and move to next bit

- How easy exactly?
- Backtracking is exponential in the worst case:
  $x \oplus 0\text{x}80000000 = x$

- For random δ, Δ, most of the time the system is inconsistent

Solving S-Systems

Important Example

$$x \oplus \Delta = x \boxplus \delta$$

- On average one solution
- Easy to solve because it's a T-function.
  - Guess LSB, check, and move to next bit

- How easy exactly?
- Backtracking is exponential in the worst case:
  $x \oplus 0\text{x}80000000 = x$

- For random δ, Δ, most of the time the system is inconsistent

Transition Automata

We use automata to study S-systems: [Mouha *et. al*]

- States represent the carries
- Transitions are labeled with the variables

Carry transitions for $x \oplus \Delta = x \boxplus \delta$.

c	Δ	δ	x	c′
0	0	0	0	0
0	0	0	1	0
0	0	1	0	-
0	0	1	1	-
0	1	0	0	-
0	1	0	1	-
0	1	1	0	0
0	1	1	1	1

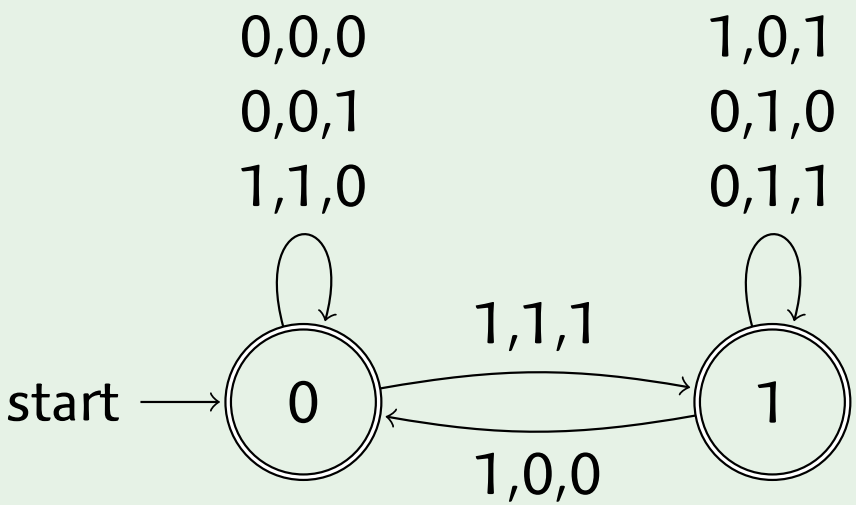
c	Δ	δ	x	c′
1	0	0	0	-
1	0	0	1	-
1	0	1	0	1
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	-
1	1	1	1	-

Transition Automata

We use automata to study S-systems: [Mouha *et. al*]

- States represent the carries
- Transitions are labeled with the variables

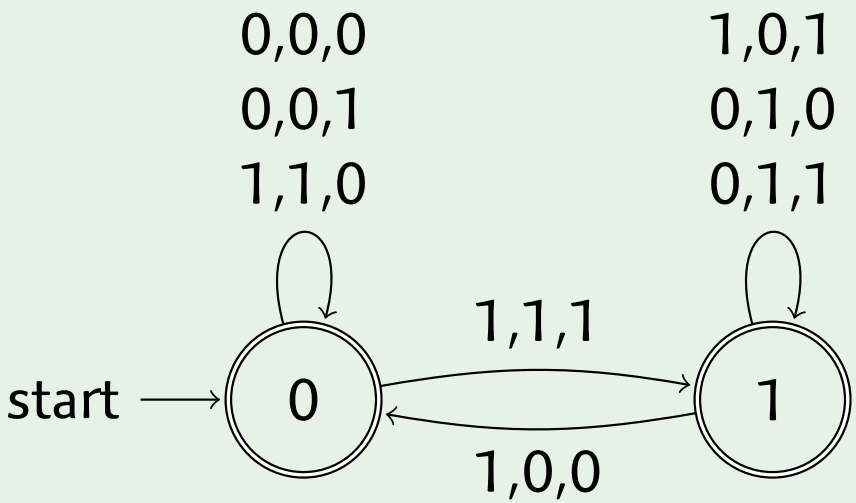
Carry transitions for $x \oplus \Delta = x \boxplus \delta$. *The edges are indexed by* Δ, δ, x



Decision Automata

- Remove x from the transitions
- Convert the non-deterministic automata to deterministic.

Carry transitions for $x \oplus \Delta = x \boxplus \delta$. *The edges are indexed by Δ, δ, x*

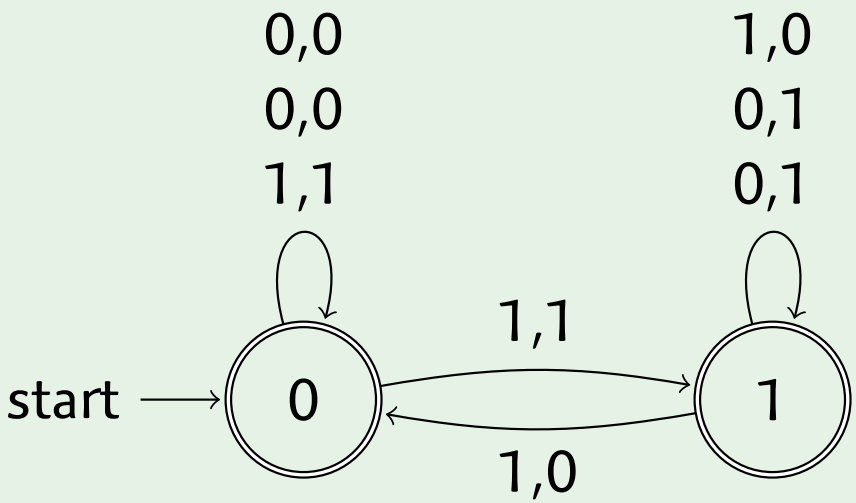


- Can decide whether a given Δ, δ is compatible.

Decision Automata

- Remove x from the transitions
- Convert the non-deterministic automata to deterministic.

Decision automaton for $x \oplus \Delta = x \boxplus \delta$. *The edges are indexed by Δ, δ*

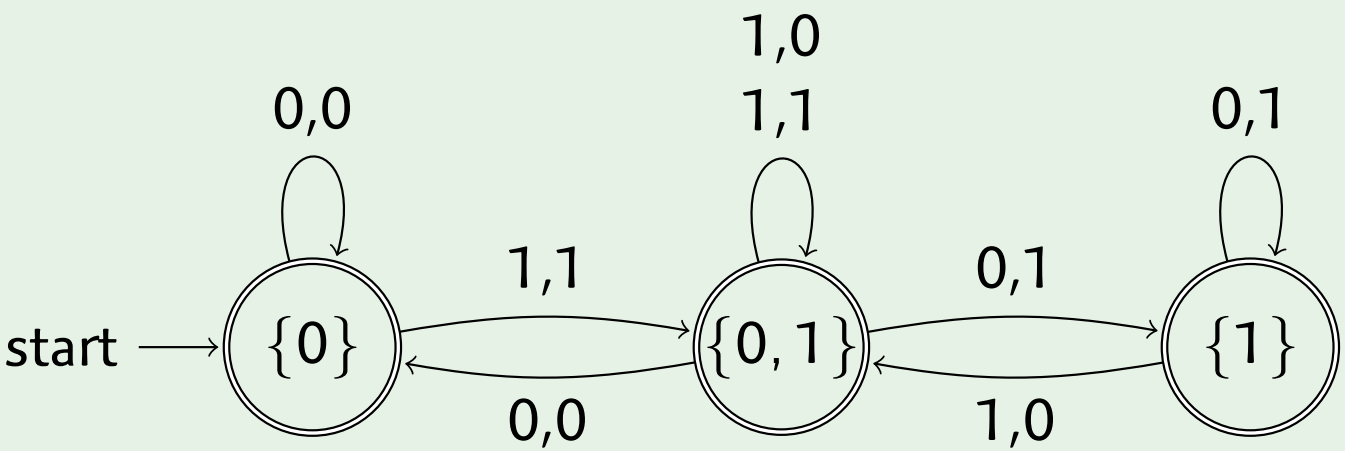


- Can decide whether a given Δ, δ is compatible.

Decision Automata

- Remove x from the transitions
- Convert the non-deterministic automata to deterministic.

Decision automaton for $x \oplus \Delta = x \boxplus \delta$. *The edges are indexed by Δ, δ*



- Can decide whether a given Δ, δ is compatible.

Solving S-systems

Take an S-system with variables and parameters.
e.g. $x \oplus \Delta = x \boxplus \delta$

1. Compute carry transitions
2. Build transition automaton
3. Remove variables and compute equivalent deterministic automaton

- For each values of the parameters:
  - Test if system is coherent in linear time
  - Count solutions in linear time
  - Find a solution in linear time

Can also study properties of the systems.

Generalized Characteristics

- We can write generalized constraints as an S-system:

$$P_0 = 0 \Rightarrow (x, x') \neq (0,0) \qquad P_1 = 0 \Rightarrow (x, x') \neq (0,1)$$
$$P_2 = 0 \Rightarrow (x, x') \neq (1,0) \qquad P_3 = 0 \Rightarrow (x, x') \neq (1,1)$$

- We can now compute the probability of a generalized characteristic.
  - Addition, Xor, Boolean functions are S-functions
  - Rotations just rotate the constraints

	(x, x'):	$(0,0)$	$(0,1)$	$(1,0)$	$(1,1)$	P_0	P_1	P_2	P_3
?	*anything*	✓	✓	✓	✓	1	1	1	1
-	$x = x'$	✓	-	-	✓	1	0	0	1
x	$x \neq x'$	-	✓	✓	-	0	1	1	0
0	$x = x' = 0$	✓	-	-	-	1	0	0	0
u	$(x, x') = (0,1)$	-	✓	-	-	0	1	0	0
n	$(x, x') = (1,0)$	-	-	✓	-	0	0	1	0
1	$x = x' = 0$	-	-	-	✓	0	0	0	1
#	*incompatible*	-	-	-	-	0	0	0	0

New Constraints

- Carry propagation leads to constraints of the form $x^{[i]} = x^{[i-1]}$
- We use new constraints to capture this information
- We consider subsets of $\left\{x^{[i]}, x'^{[i]}, x^{[i-1]}\right\}$ instead of $\left\{x^{[i]}, x'^{[i]}\right\}$
- This can still be written as an S-system with Boolean filtering on $x, x', x \boxplus x$.

New Constraints Table

	$(x \oplus x', x \oplus 2x, x)$:	$(0,0,0)$	$(0,0,1)$	$(0,1,0)$	$(0,1,1)$	$(1,0,0)$	$(1,0,1)$	$(1,1,0)$	$(1,1,1)$
?	*anything*	✓	✓	✓	✓	✓	✓	✓	✓
-	$x = x'$	✓	✓	✓	✓	-	-	-	-
x	$x \neq x'$	-	-	-	-	✓	✓	✓	✓
0	$x = x' = 0$	✓	-	✓	-	-	-	-	-
u	$(x, x') = (0, 1)$	-	-	-	-	✓	-	✓	-
n	$(x, x') = (1, 0)$	-	-	-	-	-	✓	-	✓
1	$x = x' = 0$	-	✓	-	✓	-	-	-	-
#	*incompatible*	-	-	-	-	-	-	-	-
3	$x = 0$	✓	-	✓	-	✓	-	✓	-
C	$x = 1$	-	✓	-	✓	-	✓	-	✓
5	$x' = 0$	✓	-	✓	-	-	✓	-	✓
A	$x' = 1$	-	✓	-	✓	✓	-	✓	-
=	$x = x' = 2x$	✓	✓	-	-	-	-	-	-
!	$x = x' \neq 2x$	-	-	✓	✓	-	-	-	-
>	$x \neq x' = 2x$	-	-	-	-	✓	✓	-	-
<	$x \neq x' \neq 2x$	-	-	-	-	-	-	✓	✓

Propagation of constraints

We use S-systems to propagate constraints:

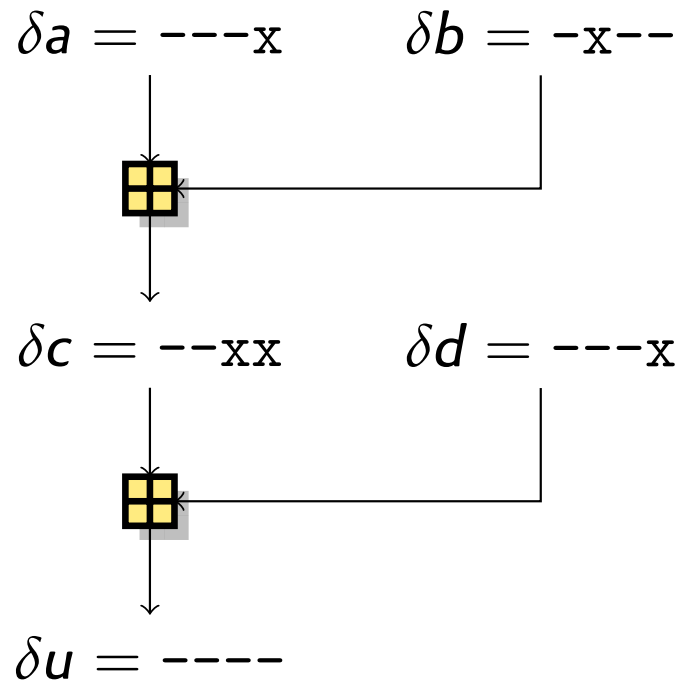
1. Split subsets in two smaller subsets
2. If one subset gives zero solutions,
   the characteristic can be restricted to the other subset.

`?` $\longrightarrow$ `-/x,3/C,5/A`	`-` $\longrightarrow$ `0/1,=/!`	`x` $\longrightarrow$ `u/n,>/<`	
`3` $\longrightarrow$ `0/u`	`C` $\longrightarrow$ `1/n`	`5` $\longrightarrow$ `0/n`	`A` $\longrightarrow$ `1/u`
`=` $\longrightarrow$ `0/1`	`!` $\longrightarrow$ `0/1`	`>` $\longrightarrow$ `u/n`	`<` $\longrightarrow$ `u/n`

Summary

- Wang looks for *sufficient* conditions.
- We compute *necessary* conditions.
- This allows to detect cases of incompatibility
- We use a graphic tool to tune paths
- To finish this talk, we will show some problems that can appear in differential paths

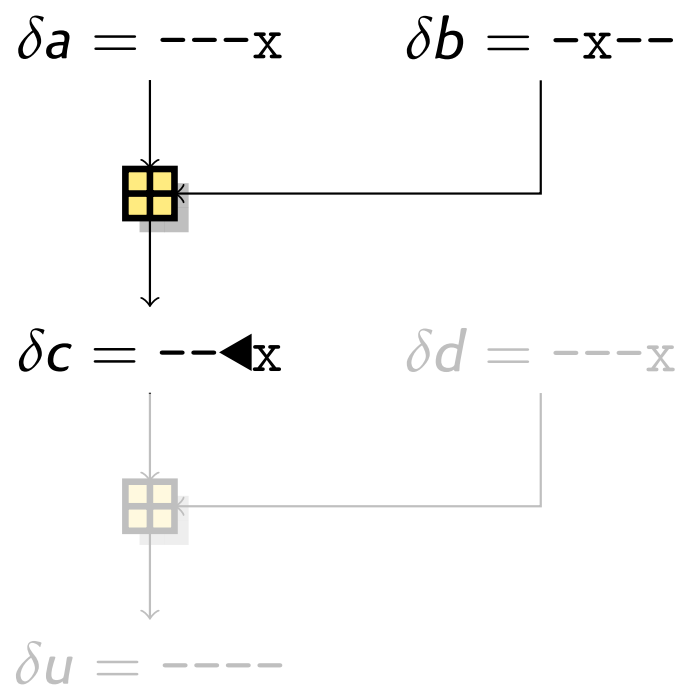
Detecting problems



$$\boxed{\begin{aligned} c &= a + b \\ u &= c + d \end{aligned}}$$

- Consider the 1st addition
  - Constraint: $c^{[1]} = c^{[0]}$
- Consider the 2nd addition
  - Constraint: $c^{[1]} \neq c^{[0]}$
- Incompatible!
  - Detected with the new constraints

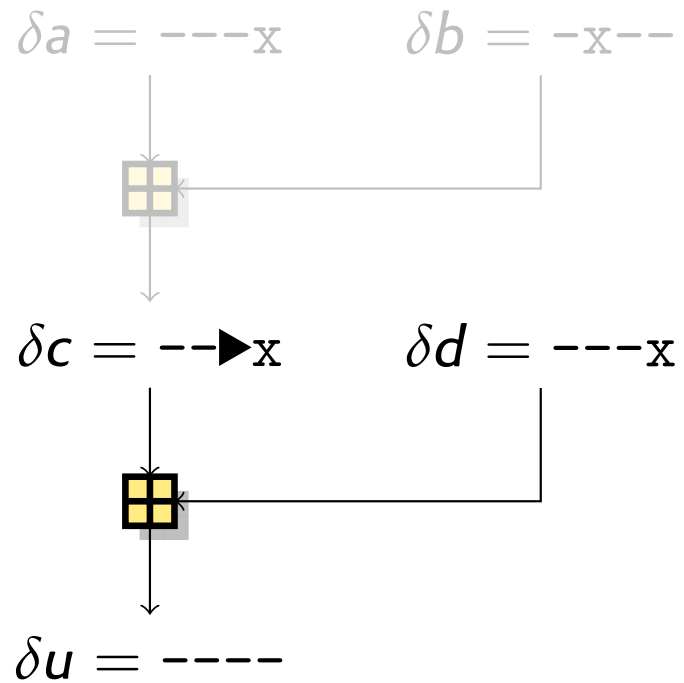
Detecting problems



$$c = a + b$$
$$u = c + d$$

- Consider the 1st addition
  - Constraint: $c^{[1]} = c^{[0]}$
- Consider the 2nd addition
  - Constraint: $c^{[1]} \neq c^{[0]}$
- Incompatible!
  - Detected with the new constraints

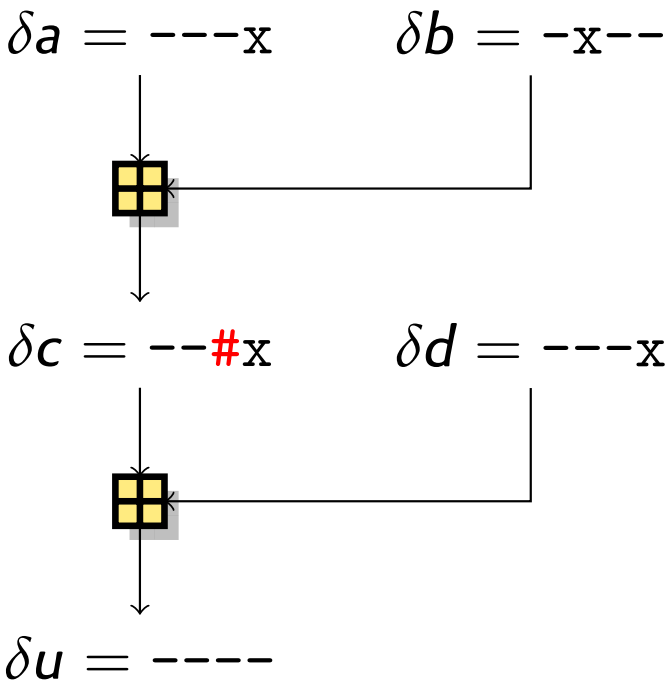
Detecting problems



$$c = a + b$$
$$u = c + d$$

- Consider the 1st addition
  - Constraint: $c^{[1]} = c^{[0]}$
- Consider the 2nd addition
  - Constraint: $c^{[1]} \neq c^{[0]}$
- Incompatible!
  - Detected with the new constraints

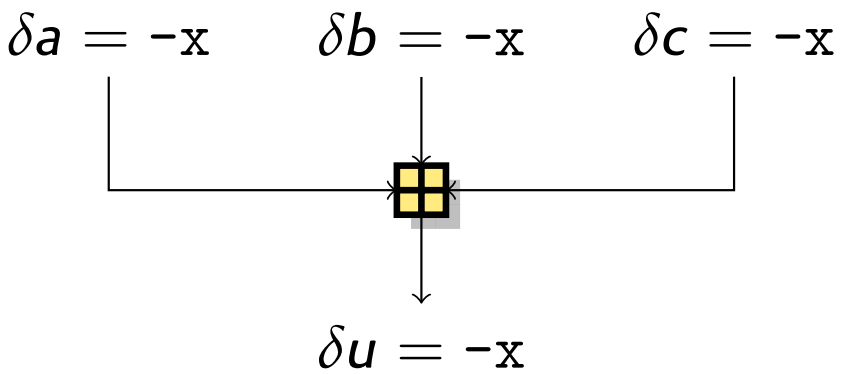
Detecting problems



$$\boxed{\begin{aligned} c &= a + b \\ u &= c + d \end{aligned}}$$

- Consider the 1st addition
  - Constraint: $c^{[1]} = c^{[0]}$
- Consider the 2nd addition
  - Constraint: $c^{[1]} \neq c^{[0]}$
- Incompatible!
  - Detected with the new constraints

Incompatibility in the low bits



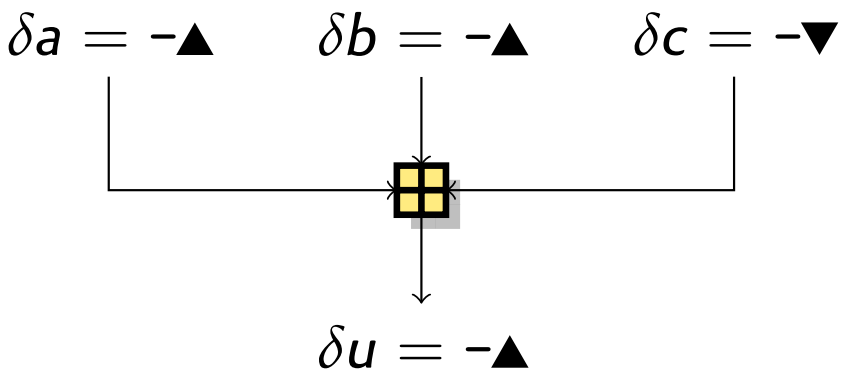
$$\boxed{u = a + b + c}$$

Active bit:

	x	x′
a	0	1
b	0	1
c	1	0
u	0	1
a+b+c	1	0

- Wlog, we can put signs
- Compute $a + b + c$
- Contradiction!

Incompatibility in the low bits



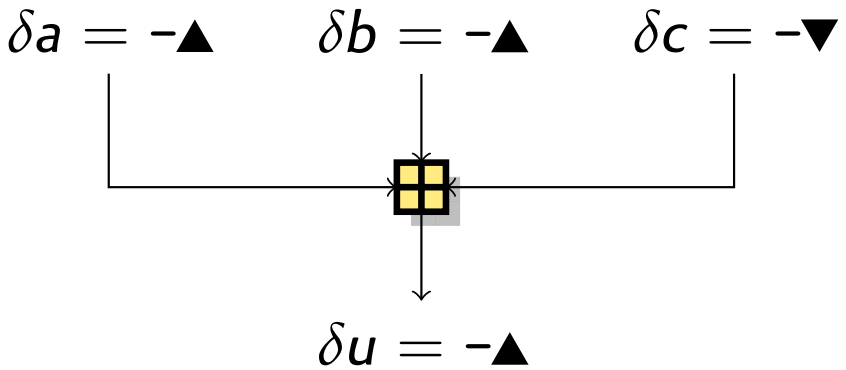
$$u = a + b + c$$

Active bit:

	x	x′
a	0	1
b	0	1
c	1	0
u	0	1
a+b+c	1	0

- Wlog, we can put signs
- Compute $a + b + c$
- Contradiction!

Incompatibility in the low bits



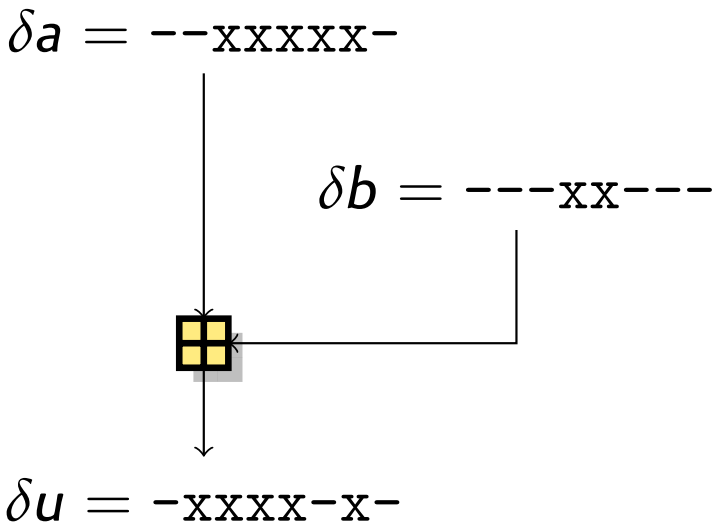
$$\boxed{u = a + b + c}$$

Active bit:

	x	x′
a	0	1
b	0	1
c	1	0
u	0	1
a+b+c	1	0

- Wlog, we can put signs
- Compute $a + b + c$
- Contradiction!

Carry incompatibility

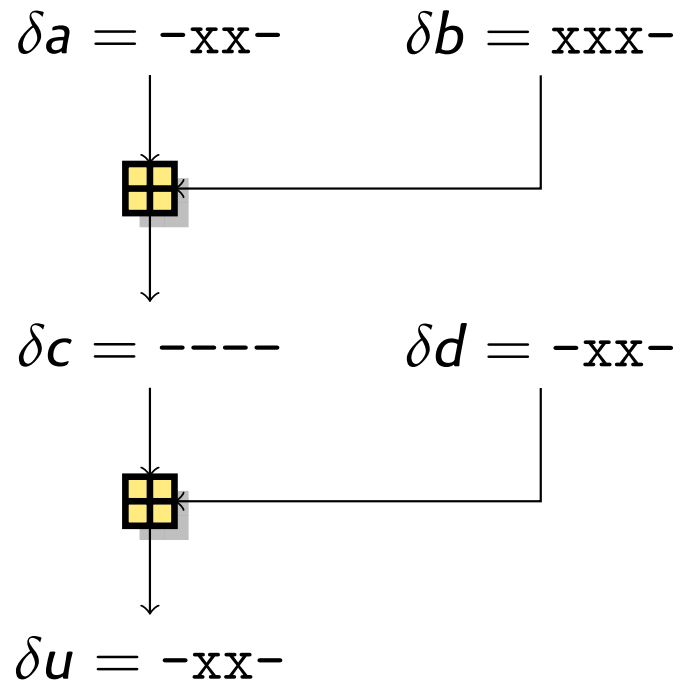


$$\boxed{u = a + b}$$

Incompatible!

No simple explanation...

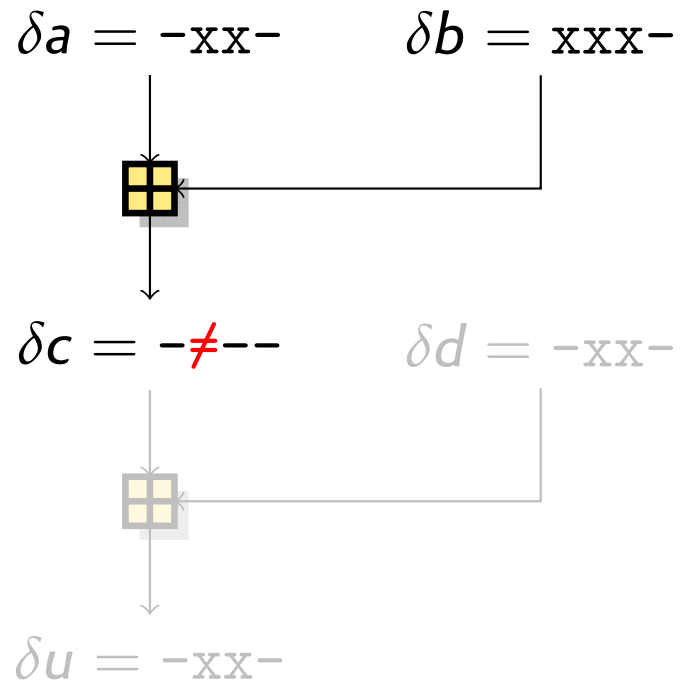
Carry incompatibility (II)



$$\boxed{\begin{aligned} c &= a + b \\ u &= c + d \end{aligned}}$$

- Consider the 1st addition
  - Constraint: $c^{[2]} \neq c^{[3]}$
- Consider the 2nd addition
  - Constraint: $c^{[2]} = c^{[3]}$
- Incompatible!
  - Detected with the new constraints

Carry incompatibility (II)

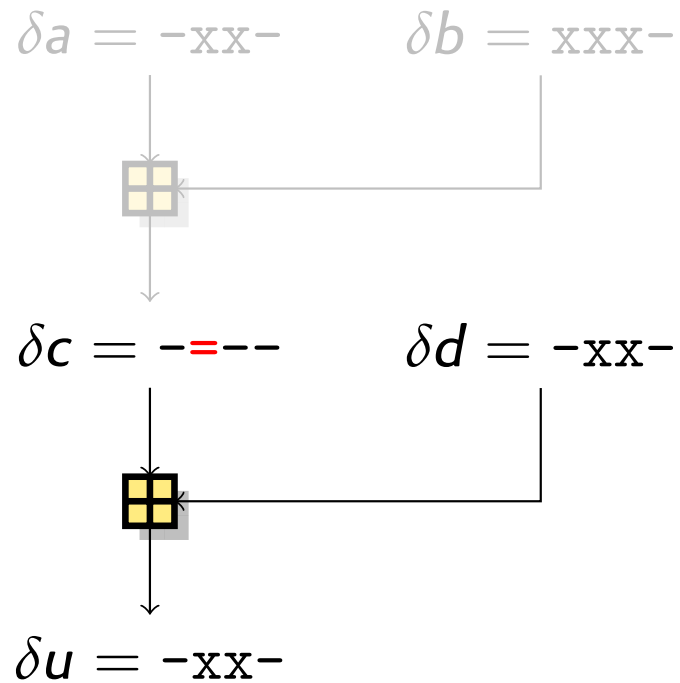


$c = a + b$

$u = c + d$

- Consider the 1st addition
  - Constraint: $c^{[2]} \neq c^{[3]}$
- Consider the 2nd addition
  - Constraint: $c^{[2]} = c^{[3]}$
- Incompatible!
  - Detected with the new constraints

Carry incompatibility (II)

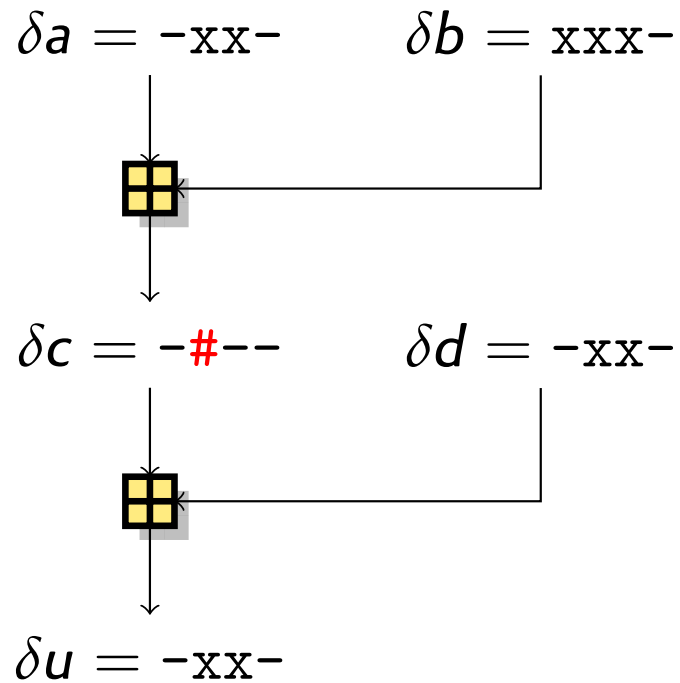


$c = a + b$

$u = c + d$

- Consider the 1st addition
  - Constraint: $c^{[2]} \neq c^{[3]}$
- Consider the 2nd addition
  - Constraint: $c^{[2]} = c^{[3]}$
- Incompatible!
  - Detected with the new constraints

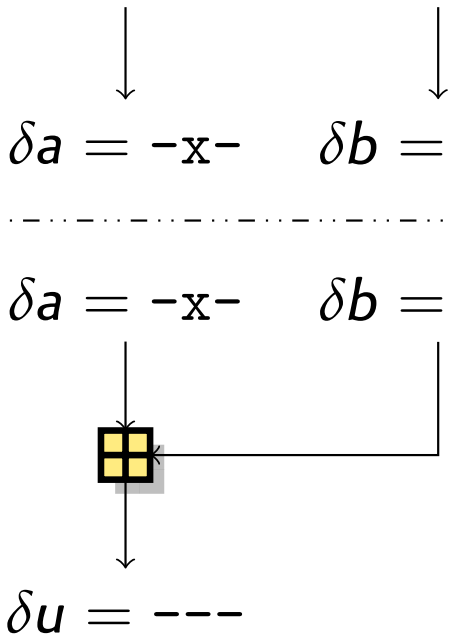
Carry incompatibility (II)



$$\boxed{\begin{aligned} c &= a + b \\ u &= c + d \end{aligned}}$$

- Consider the 1st addition
  - Constraint: $c^{[2]} \neq c^{[3]}$
- Consider the 2nd addition
  - Constraint: $c^{[2]} = c^{[3]}$
- Incompatible!
  - Detected with the new constraints

Boomerang incompatibility



Top path: $(a^{(0)}, b^{(0)}; a^{(2)}, b^{(2)})\ (a^{(1)}, b^{(1)}; a^{(3)}, b^{(3)})$

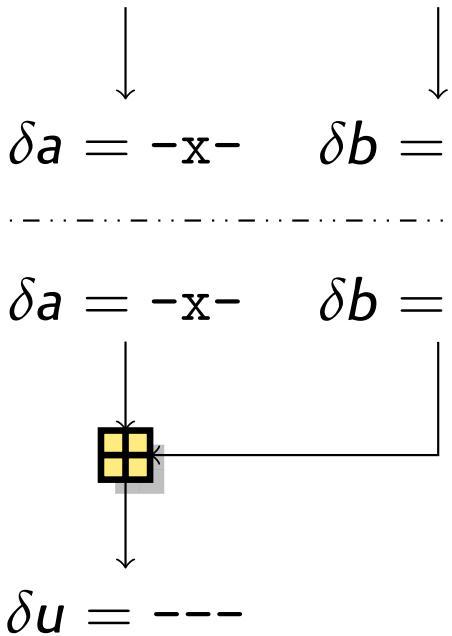
Bottom path: $(a^{(0)}, b^{(0)}; a^{(1)}, b^{(1)})\ (a^{(2)}, b^{(2)}; a^{(3)}, b^{(3)})$

$$\boxed{u = a + b}$$

	$x^{(0)}$	$x^{(1)}$	$x^{(2)}$	$x^{(3)}$
a	0	1	1	0
b	1	0	0	1

- Wlog, assume $a^{(0)} = 0$
- Compute $a^{(i)}$, deduce sign of b
- Contradiction for b!

Boomerang incompatibility



Top path: $(a^{(0)}, b^{(0)}; a^{(2)}, b^{(2)})\ (a^{(1)}, b^{(1)}; a^{(3)}, b^{(3)})$

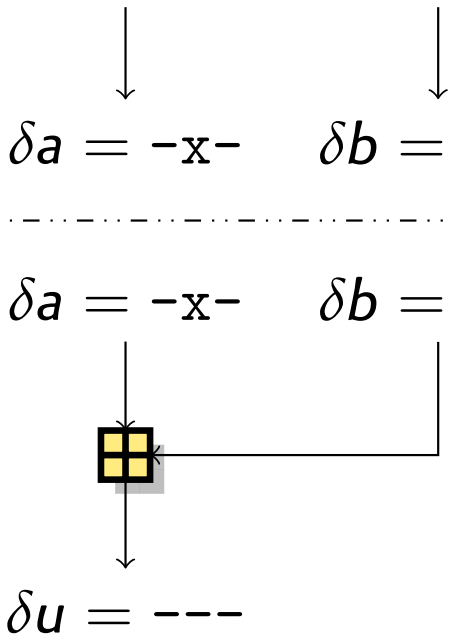
Bottom path: $(a^{(0)}, b^{(0)}; a^{(1)}, b^{(1)})\ (a^{(2)}, b^{(2)}; a^{(3)}, b^{(3)})$

$$\boxed{u = a + b}$$

	$x^{(0)}$	$x^{(1)}$	$x^{(2)}$	$x^{(3)}$
a	0	1	1	0
b	1	0	0	1

- Wlog, assume $a^{(0)} = 0$
- Compute $a^{(i)}$, deduce sign of b
- Contradiction for b!

Boomerang incompatibility



Top path: $(a^{(0)}, b^{(0)}; a^{(2)}, b^{(2)})\ (a^{(1)}, b^{(1)}; a^{(3)}, b^{(3)})$

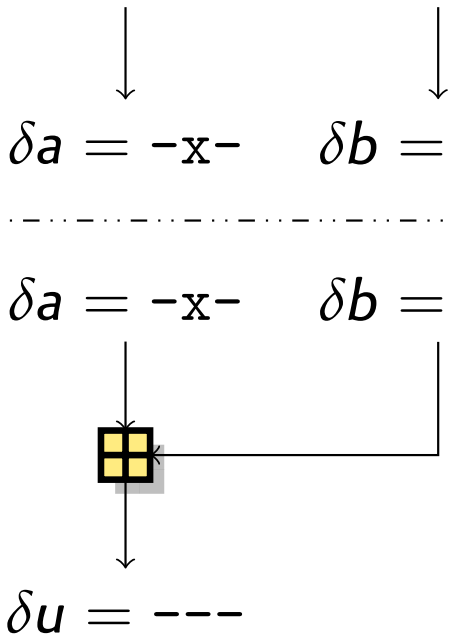
Bottom path: $(a^{(0)}, b^{(0)}; a^{(1)}, b^{(1)})\ (a^{(2)}, b^{(2)}; a^{(3)}, b^{(3)})$

$$\boxed{u = a + b}$$

	$x^{(0)}$	$x^{(1)}$	$x^{(2)}$	$x^{(3)}$
a	0	1	1	0
b	1	0	0	1

- Wlog, assume $a^{(0)} = 0$
- Compute $a^{(i)}$, deduce sign of b
- Contradiction for b!

Boomerang incompatibility



Top path: $(a^{(0)}, b^{(0)}; a^{(2)}, b^{(2)})\ (a^{(1)}, b^{(1)}; a^{(3)}, b^{(3)})$

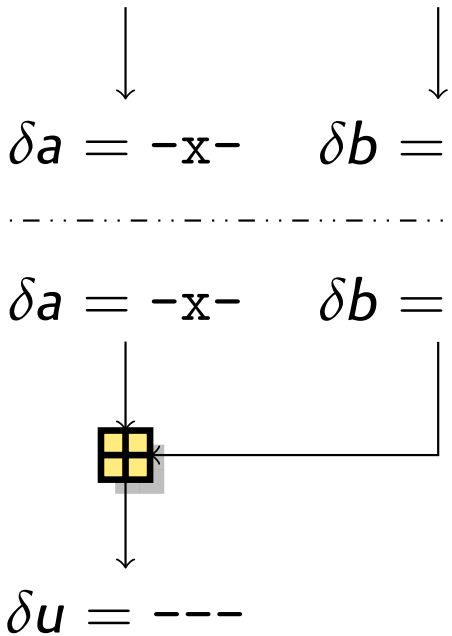
Bottom path: $(a^{(0)}, b^{(0)}; a^{(1)}, b^{(1)})\ (a^{(2)}, b^{(2)}; a^{(3)}, b^{(3)})$

$$\boxed{u = a + b}$$

	$x^{(0)}$	$x^{(1)}$	$x^{(2)}$	$x^{(3)}$
a	0	1	1	0
b	1	0	0	1

- Wlog, assume $a^{(0)} = 0$
- Compute $a^{(i)}$, deduce sign of b
- Contradiction for b!

Boomerang incompatibility



Top path: $(a^{(0)}, b^{(0)}; a^{(2)}, b^{(2)})\ (a^{(1)}, b^{(1)}; a^{(3)}, b^{(3)})$

Bottom path: $(a^{(0)}, b^{(0)}; a^{(1)}, b^{(1)})\ (a^{(2)}, b^{(2)}; a^{(3)}, b^{(3)})$

$$\boxed{u = a + b}$$

	$x^{(0)}$	$x^{(1)}$	$x^{(2)}$	$x^{(3)}$
a	0	1	1	0
b	1	0	0	1

- Wlog, assume $a^{(0)} = 0$
- Compute $a^{(i)}$, deduce sign of b
- Contradiction for b!

Conclusion

1. Boomerang attack on hash functions
   - Use auxiliary path to avoid middle rounds
   - Significant improvement over previous results
2. Analysis of differentials paths
   - New constraints for carries
   - Tools for constraint propagations
   - Problems found in several previous works