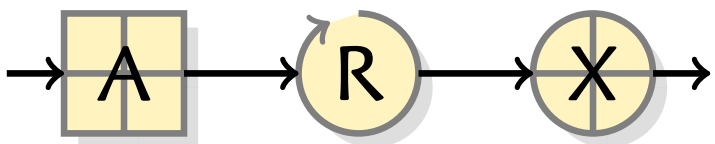
Differential Attacks against ARX Designs

Gaëtan Leurent

UCL Crypto Group & University of Luxembourg

Early Symmetric Crypto 2013



References

G. L.
Analysis of Differential Attacks in ARX Constructions
Asiacrypt 2012

G. L.
Construction of Differential Characteristics in ARX Designs
Application to Skein
ePrint 2012/668

ARX Constructions

Two main categories of designs in symmetric cryptography:

ARX designs

- Additions, Rotations, Xors
- Inspired by MD/SHA
- Lots of light rounds

SBox designs

- S-Boxes and Linear Layers
- Inspired by the AES
- Few heavy rounds

ARX designs

- Hash: Skein, BLAKE (2 of the 5 SHA-3 finalists)
- Stream: Slasa20, ChaCha
- Block: TEA, XTEA, HIGHT
- SipHash

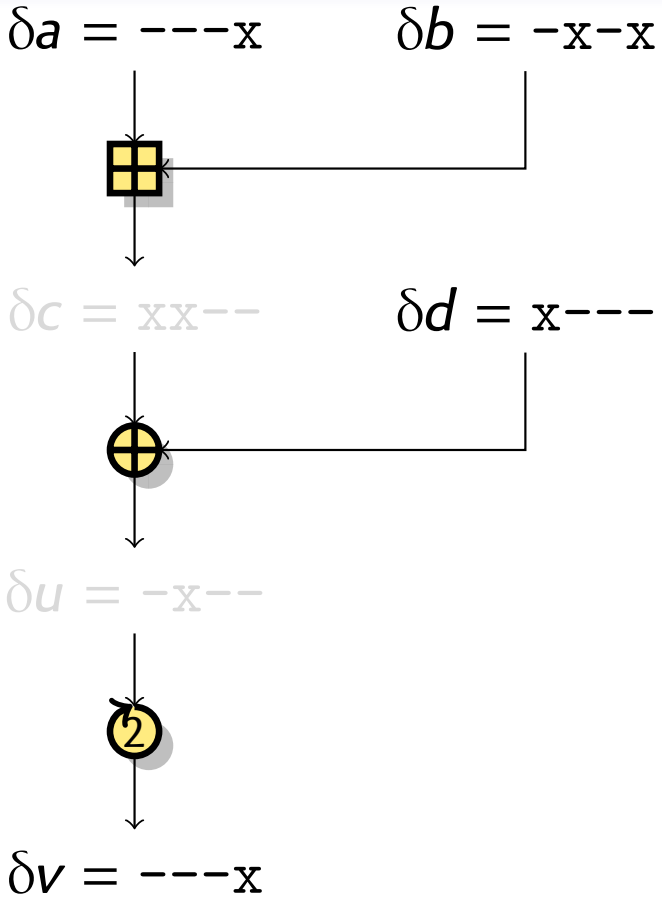
Differential attacks against ARX

- Most of the cryptanalysis of ARX designs is bit-twiddling
  - As opposed to SBox based designs
- Building/Verifying differential trails for ARX designs is hard
  - Many trails built by hand
  - Problems with MD5 and SHA-1 attacks [Manuel, DCC 2011]
  - Problems with differential trails
    - SHACAL [Wang, Keller & Dunkelman, SAC 2007]
  - Problems reported with boomerang attacks (incompatible trails):
    - HAVAL [Sasaki, SAC 2011]
    - SHA-256 [BLMN, Asiacrypt 2011]
- Some tools are described in literature, but many are not public

Outline

Outline

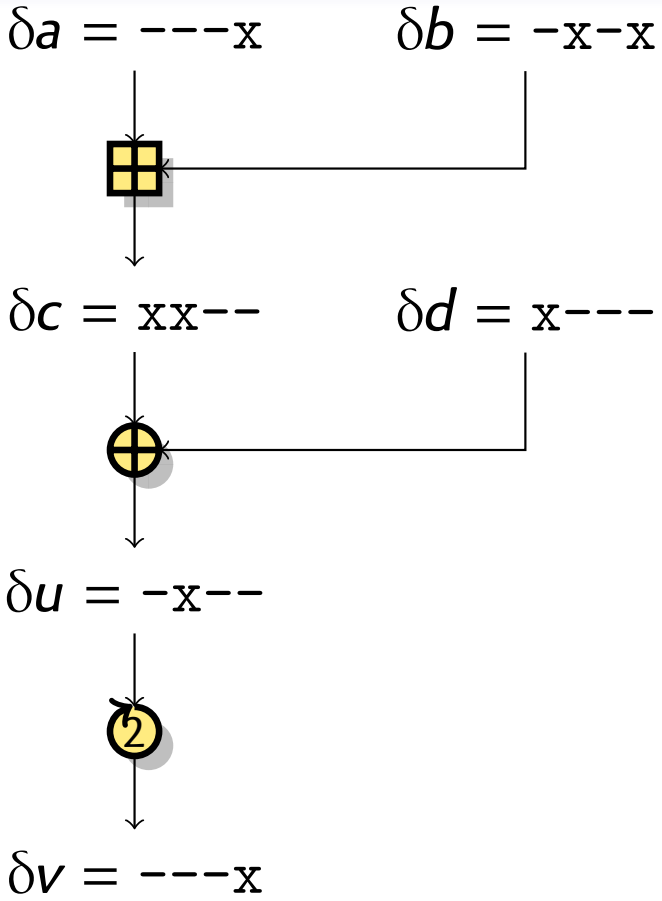
Differential Characteristic



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

- Choose a difference operation: $\oplus$
- A differential only specifies the input and output difference
- A differential characteristic specifies the difference of each internal variable
- Compute probability for each operation

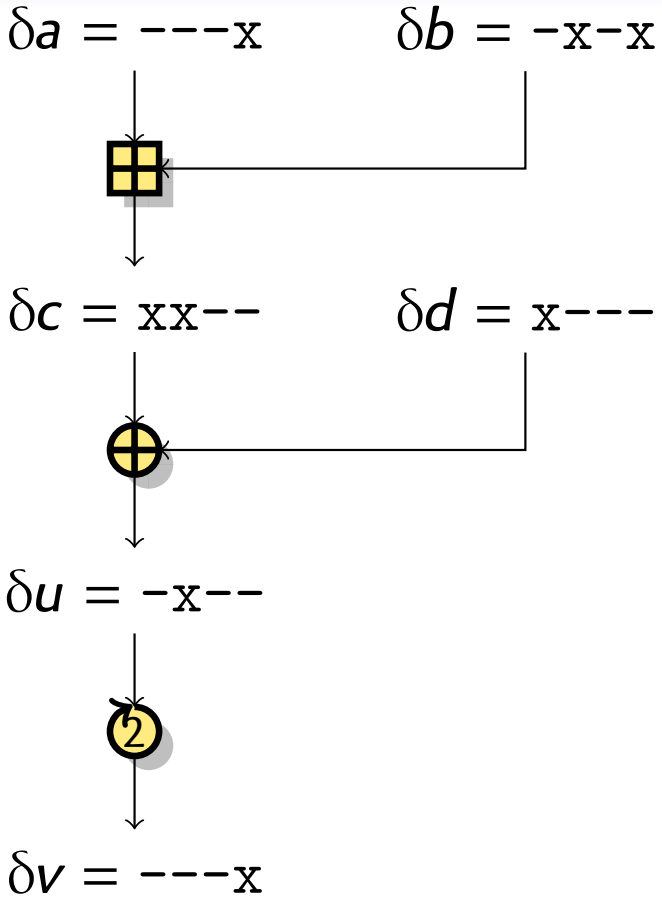
Differential Characteristic



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

- Choose a difference operation: $\oplus$
- A differential only specifies the input and output difference
- A differential characteristic specifies the difference of each internal variable
- Compute probability for each operation

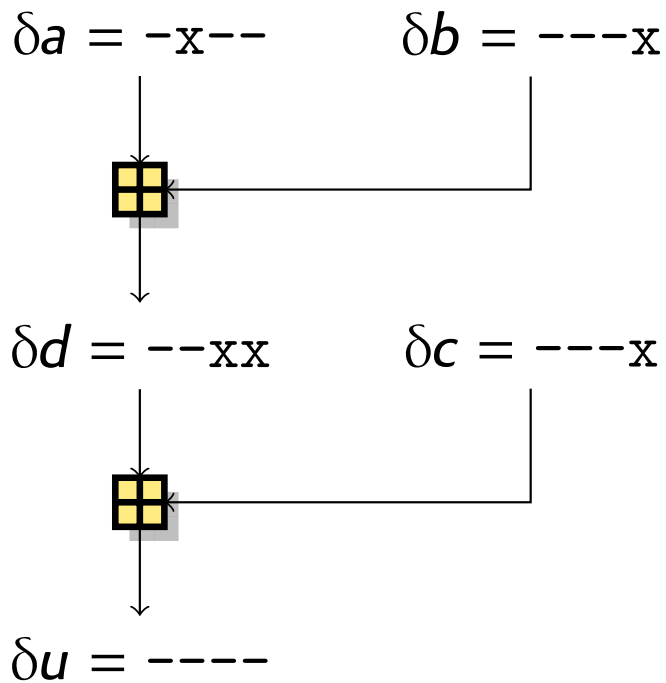
Differential Characteristic



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

- Choose a difference operation: $\oplus$
- A differential only specifies the input and output difference
- A differential characteristic specifies the difference of each internal variable
- Compute probability for each operation

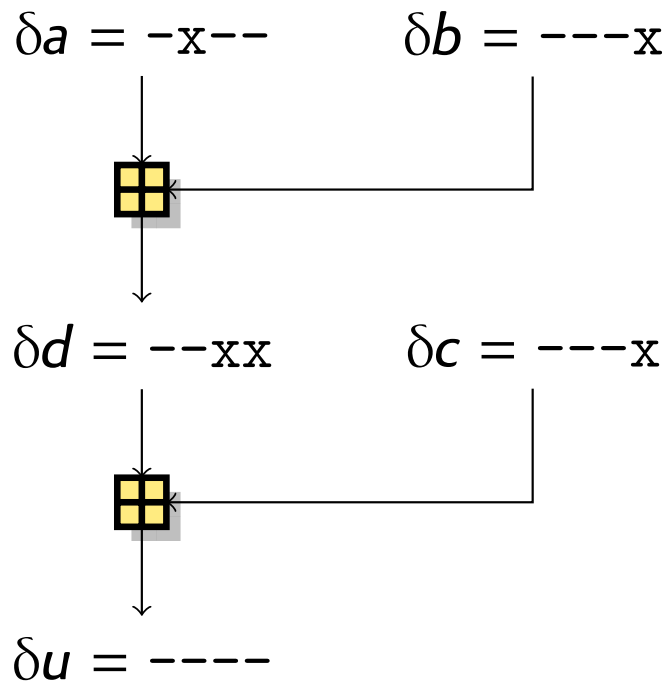
Problems with Xor-Characteristics



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

- 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



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

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

Signed difference

- A trail defines a set of good pairs:
  - $x^{[i]} \oplus x'^{[i]} = 0 \quad \Leftrightarrow \quad \left(x^{[i]}, x'^{[i]}\right) \in \left\{(0,0),(1,1)\right\}$
  - $x^{[i]} \oplus x'^{[i]} = 1 \quad \Leftrightarrow \quad \left(x^{[i]}, x'^{[i]}\right) \in \left\{(0,1),(1,0)\right\}$

- Wang introduced a signed difference:
  - $\delta\left(x^{[i]}, x'^{[i]}\right) = \ \ 0 \quad \Leftrightarrow \quad (x^{[i]}, x'^{[i]}) \in \left\{(0,0),(1,1)\right\}$
  - $\delta\left(x^{[i]}, x'^{[i]}\right) = +1 \quad \Leftrightarrow \quad (x^{[i]}, x'^{[i]}) \in \left\{(0,1)\right\}$
  - $\delta\left(x^{[i]}, x'^{[i]}\right) = -1 \quad \Leftrightarrow \quad (x^{[i]}, x'^{[i]}) \in \left\{(1,0)\right\}$
  - Captures both xor difference and modular difference

- Generalized constraints [De Cannière & Rechberger 06]

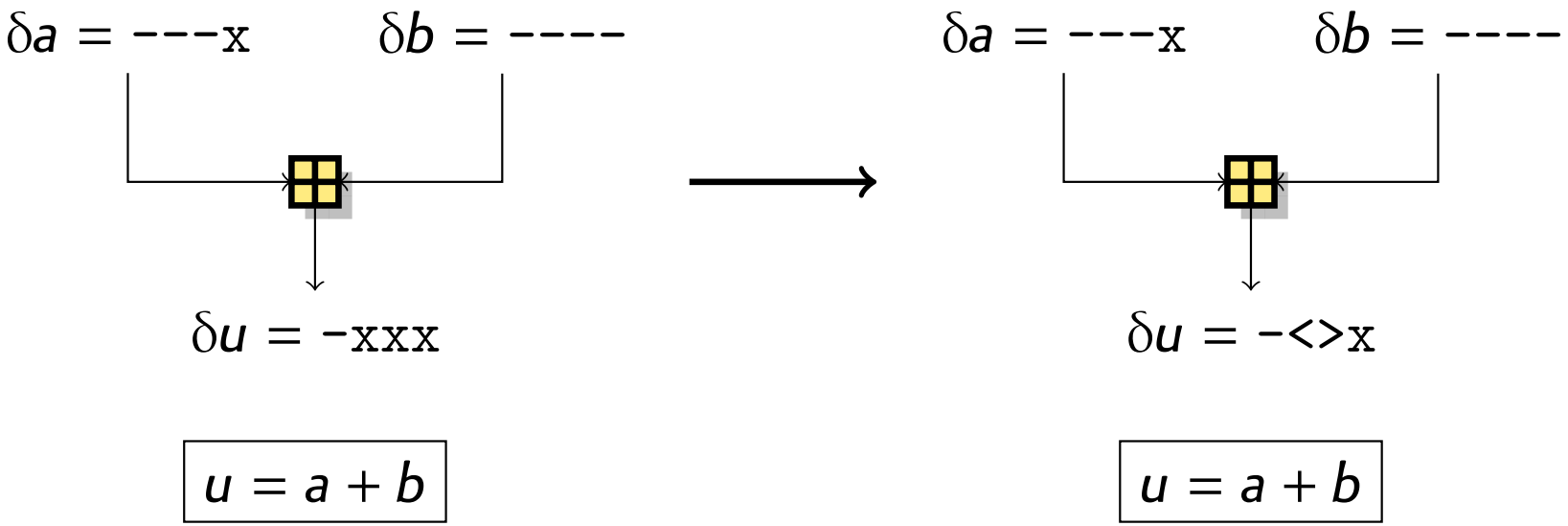
Generalized constraints [De Cannière & Rechberger 06]

	(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		-	✓	✓	✓

Outline

Multi-bit Constraints

- We study carry propagation



- Two possibilities:
  - $\delta a = \texttt{---u}$ and $\delta u = \texttt{-unn}$
  - $\delta a = \texttt{---n}$ and $\delta u = \texttt{-nuu}$
- Active bits signs are linked

- We introduce new constraints
  - $> \equiv \left\{\texttt{n}^{\texttt{n}}, \texttt{u}^{\texttt{u}}\right\}: x'^{[i]} \neq x^{[i]} = x^{[i-1]}$
  - $< \equiv \left\{\texttt{n}^{\texttt{u}}, \texttt{u}^{\texttt{n}}\right\}: x'^{[i]} \neq x^{[i]} \neq x^{[i-1]}$

Multi-bit Constraints

- Carry propagation leads to constraints of the form $x^{[i]} = x^{[i-1]}$
- We use multi-bit constraints to capture this information
  - We consider subsets of $\left\{(x^{[i]}, x'^{[i]}, x^{[i-1]})\right\}$ (1.5-bit), instead of $\left\{(x^{[i]}, x'^{[i]})\right\}$ (1-bit)
- Captures more accurately the behavior of modular addition
  - Only source of non-linearity in pure ARX designs (Boolean functions in MD/SHA)
  - More precise constraints allow less invalid characteristics

Generalization

- 1.5-bit constraints: subsets of $\left\{(x^{[i]}, x'^{[i]}, x^{[i-1]}\right\}$
  - Relations between carry extensions
- 2-bit constraints: subsets of $\left\{(x^{[i]}, x'^{[i]}, x^{[i-1]}, x'^{[i-1]})\right\}$
  - Describe exactly the set $\{x, x' | x' = x \boxplus \Delta\}$ for any Δ
- 2.5-bit constraints: subsets of $\left\{(x^{[i]}, x'^{[i]}, x^{[i-1]}, x'^{[i-1]}, x^{[i-2]}\right\}$
  - Relations between potential carry extensions

▸ See examples

Limitations in the Asiacrypt paper

- We have to use reduced sets of constraints
- Propagation for 2.5-bit constraints is slow

Generalization

- 1.5-bit constraints: subsets of $\left\{(x^{[i]}, x'^{[i]}, x^{[i-1]}\right\}$
  - Relations between carry extensions
- 2-bit constraints: subsets of $\left\{(x^{[i]}, x'^{[i]}, x^{[i-1]}, x'^{[i-1]})\right\}$
  - Describe exactly the set $\{x, x' | x' = x \boxplus \Delta\}$ for any Δ
- 2.5-bit constraints: subsets of $\left\{(x^{[i]}, x'^{[i]}, x^{[i-1]}, x'^{[i-1]}, x^{[i-2]}\right\}$
  - Relations between potential carry extensions

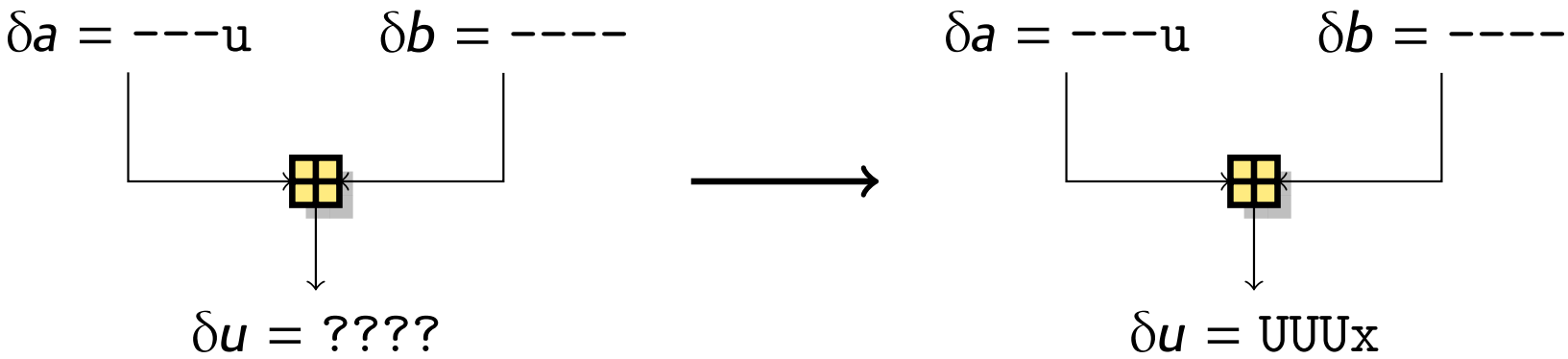
▸ See examples

Limitations in the Asiacrypt paper

- Solved using a new method for constraint propagation
- We use the full set of 2^{32} constraints

2-bit constraints

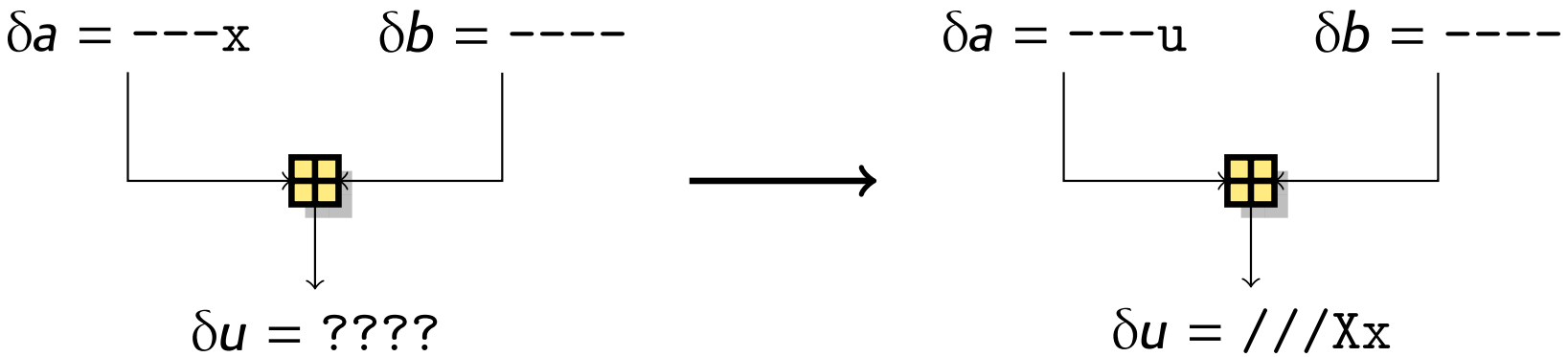
- 1.5-bit conditions extract information from carries when the xor difference is known
- What if the xor-sum is not known?



- Several possibilities:
  - $\delta u =$ ---u
  - $\delta u =$ --un
  - $\delta u =$ -unn
  - $\delta u =$ unnn
  - $\delta u =$ nnnn
- For middle bits, the pattern for bits i and $i-1$ is one of {--, -u, un, nn}
- We denote this as U

2.5-bit constraints

- 2-bit conditions extract information from carries when the sign of the input difference is known
- What if the sign is not known?



- Several possibilities:
  - δu = ---x
  - δu = --<x
  - δu = -<>x
  - δu = <>>x
  - δu = >>>x
- For middle bits, the pattern for bits i and $i-1$ is one of {--, -<, <>, >>}
- We denote this as /

Comparison

- Experiments with a few rounds of a reduced Skein (4-bit words and 6-bit words)
- We look at the number of accepted input/output differences

	2 rounds (total: 2^{32})		3 rounds (sparse)	
Method	Accepted	Fp.	Accepted	Fp.
Exhaustive search	$2^{25.1}$ (35960536)	0	$2^{18.7}$ (427667)	
2.5-bit full set	$2^{25.3}$ (40597936)	0.13	$2^{19.2}$ (619492)	0.4
2.5-bit constraints	$2^{25.3}$ (40820032)	0.14	$2^{19.5}$ (746742)	0.7
1.5-bit constraints	$2^{25.3}$ (40820032)	0.14	$2^{20.4}$ (1372774)	2.2
1-bit constraints	$2^{25.4}$ (43564288)	0.21	$2^{20.7}$ (1762857)	3.1
Check adds indep.	$2^{25.8}$ (56484732)	0.57		

Outline

Multi-bit Constraints as S-systems

We use the theory of S-functions to study multi-bit constraints [Mouha & *al.*, SAC 2010]

- We write a bitwise function f so that:

$$(x, x') \text{ is a right pair for } \Delta \Leftrightarrow f(x, x', x \boxplus x, \Delta) = 0$$

- We can count the number of solutions efficiently
  - Compute probabilities
  - Testing for zero or non-zero very efficient

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 set of states $\mathcal{S}$ so that:* $\forall t$, bit t of the output and state $S[t] \in \mathcal{S}$ can be computed from bit t of the input, and state $S[t-1]$.

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

- Operations mod 2^n, bitwise functions are T-functions:
  - Empty state for bitwise Boolean function
  - 1-bit state for addition (carry)
  - t states for multiplication by t

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

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	-

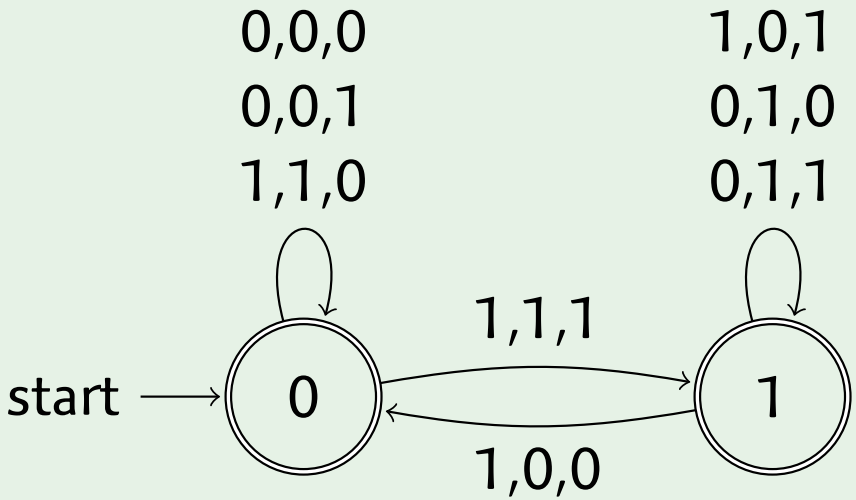
We use automata to study S-systems: [Mouha & *al.*, SAC 2010]

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

- Automaton accepts solutions to the system.
- Can count the number of solutions.

Transition Automata

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



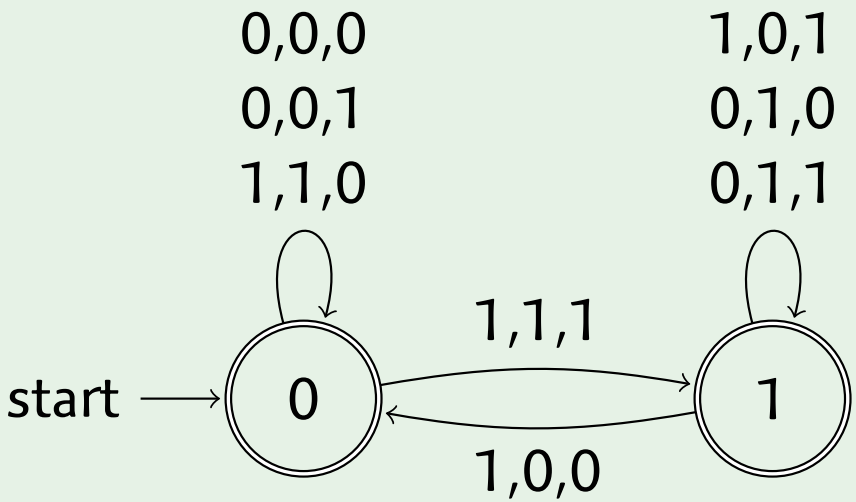
We use automata to study S-systems: [Mouha & *al.*, SAC 2010]

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

- Automaton accepts solutions to the system.
- Can count the number of solutions.

Transition Automata

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

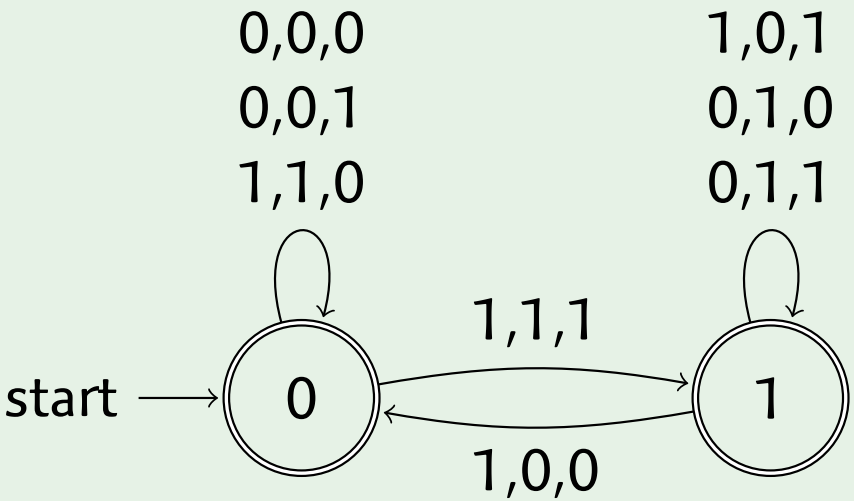


We use automata to study S-systems: [Mouha & *al.*, SAC 2010]

- States represent the carries
- Transitions are labeled with the variables
- Automaton accepts solutions to the system.
- Can count the number of solutions.

Decision Automata

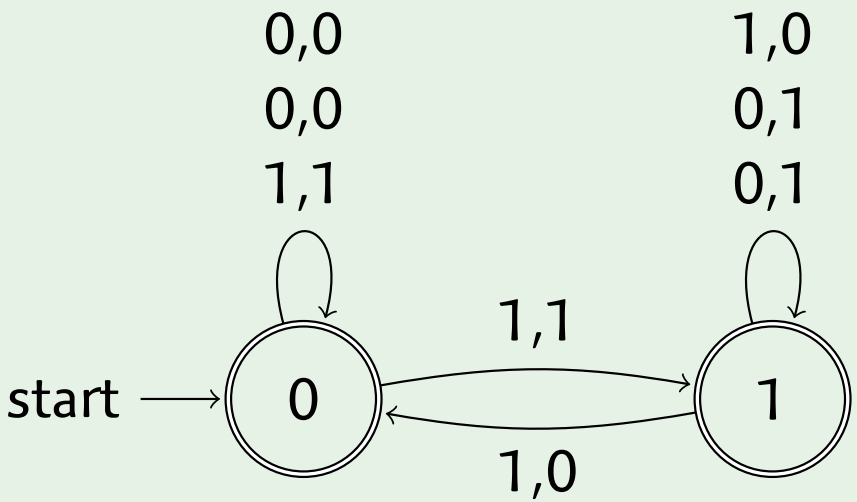
Carry transitions for $x \oplus \Delta = x \boxplus \delta$. *The edges are indexed by Δ, δ, x*



- Remove x from the transitions
- Can decide whether a given Δ, δ is compatible.
- Convert the non-deterministic automata to deterministic (optional).

Decision Automata

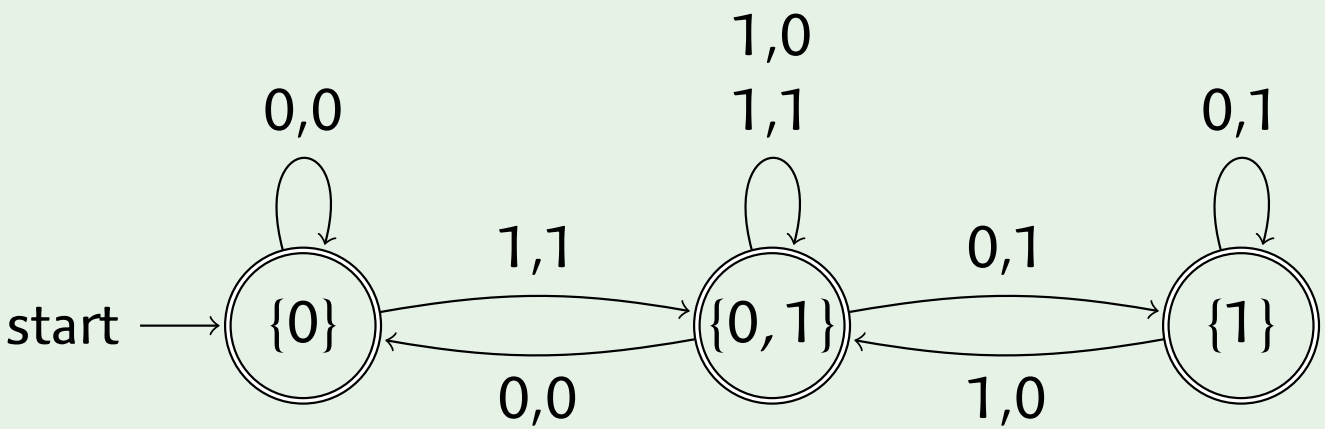
Decision automaton for $x \oplus \Delta = x \boxplus \delta$. *The edges are indexed by Δ, δ*



- Remove x from the transitions
- Can decide whether a given Δ, δ is compatible.
- Convert the non-deterministic automata to deterministic (optional).

Decision Automata

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

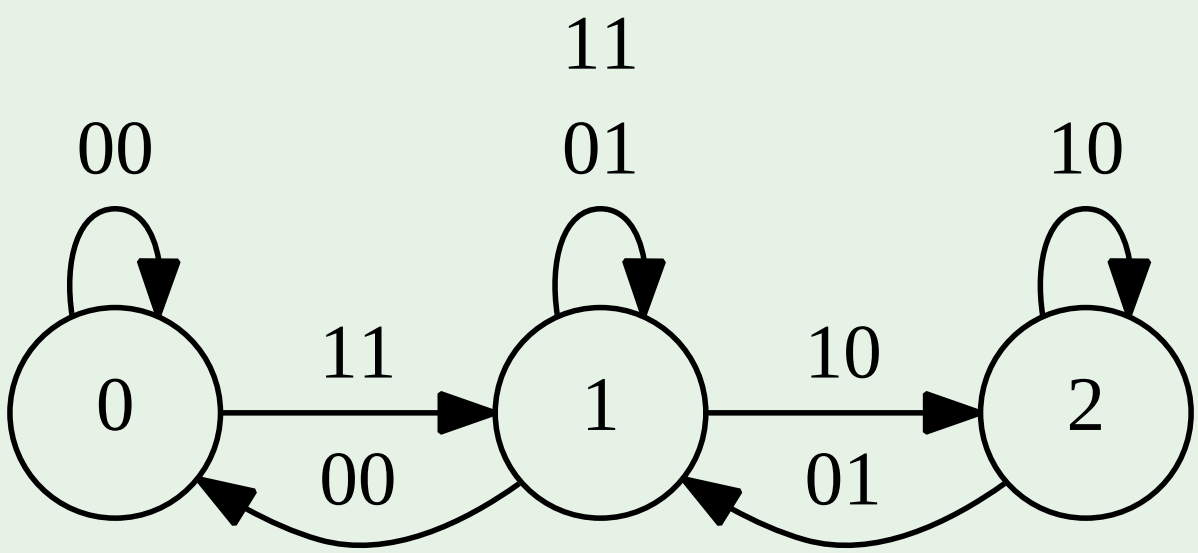


- Remove x from the transitions
- Can decide whether a given Δ, δ is compatible.
- Convert the non-deterministic automata to deterministic (optional).

Our Tool

1. Automatic construction of the automaton from a natural expression
   Useful to study properties of the system

```
build_fsm -e "V0+P0 == V0^P1" -d -g | dot -Teps
```



2. C functions to test compatibility, count solutions, or solve systems

Application to multi-bit constraints

1. For each operation $\odot$, write a system:

$$z = x \odot y$$
$$f(x, x', x \boxplus x, \Delta_x) = 0$$
$$f(y, y', y \boxplus y, \Delta_y) = 0$$
$$f(z, z', z \boxplus z, \Delta_z) = 0$$

2. Build the automaton

3. Count the number of solutions for given $\Delta_x, \Delta_y, \Delta_z$ (*i.e.* probability)

Limitations in the Asiacrypt paper

For the full set of 2^{32} 2.5-bit constraints, the system is too large.

More efficient technique

How to compute probabilities with the full set of 2^{32} 2.5-bit constraints?

1. Build the system for a set of 32 base constraints

▸ See details

2. Take the union of the transitions

Important property

- For 2-bit and 2.5-bit constraints,
  there is a single transition between any pair of states
- This gives an efficient constraints propagation

▸ See example

Outline

Verifying trails

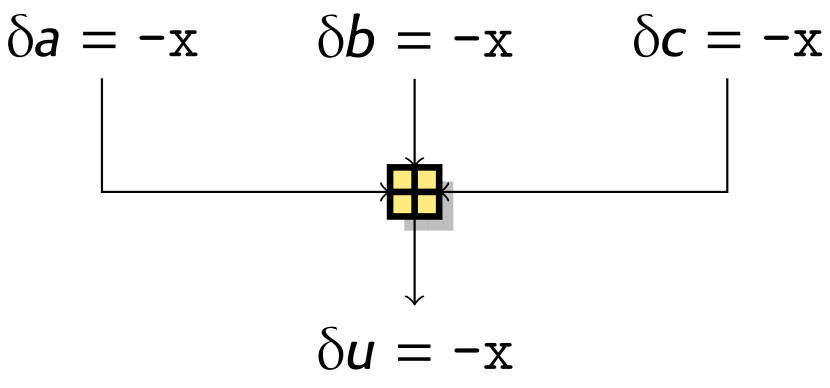
Problem

Most analysis assume that operations are independent
and multiply the probabilities.
But sometimes, operations are not independent...
Known problem in Boomerang attacks. [Murphy, TIT 2011]

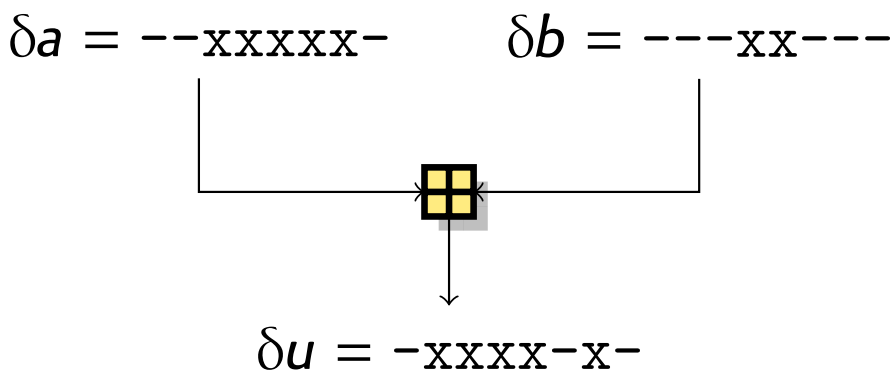
- We compute necessary conditions.
- This allows to detect cases of incompatibility
- We have detected problems in several published works
  - Incompatible trails seem to appear quite naturally

Incompatibility with additions

Some "natural" differentials do not work with additions:



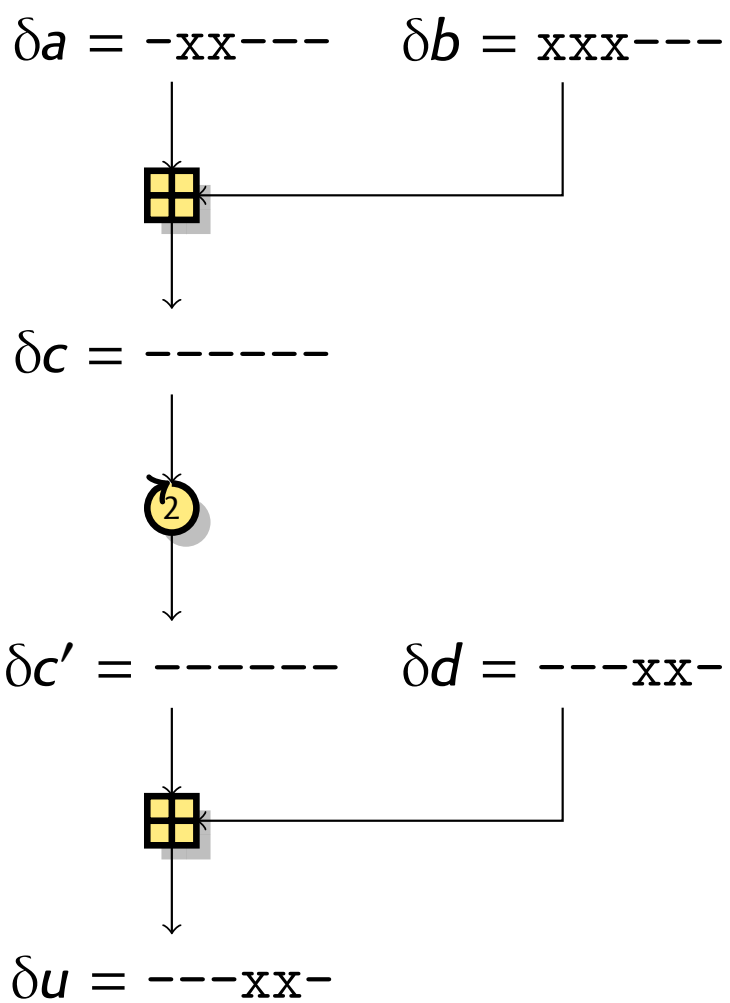
$u = a + b + c$



$u = a + b$

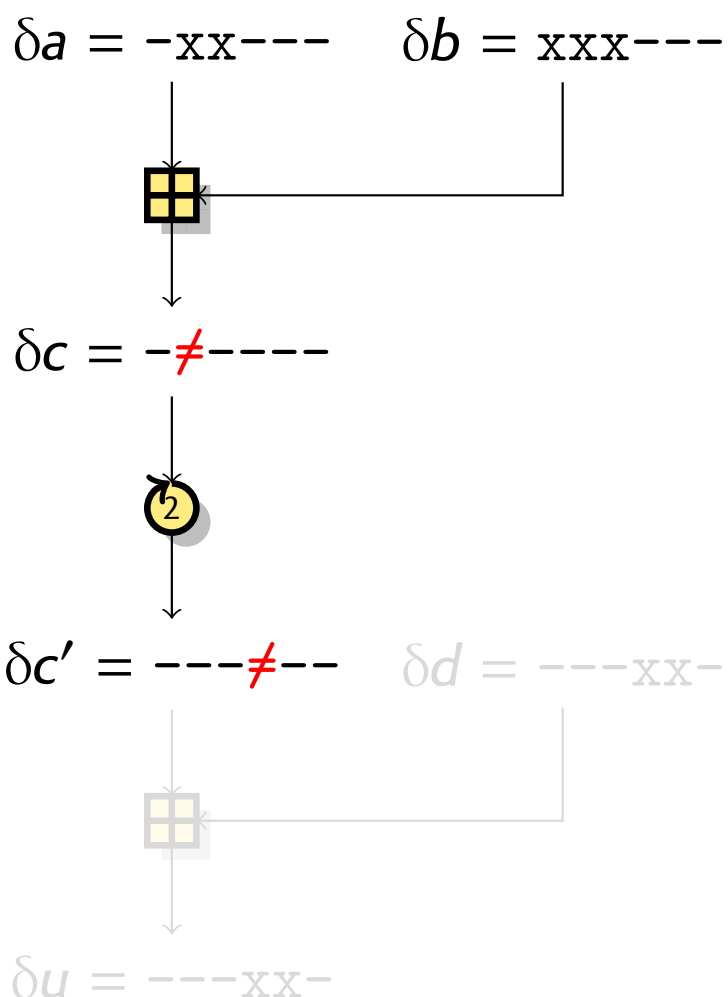
- Linearized trail
- Seems valid with signed difference
- Found in Skein near-collision [eprint 2011/148]

Carry incompatibility



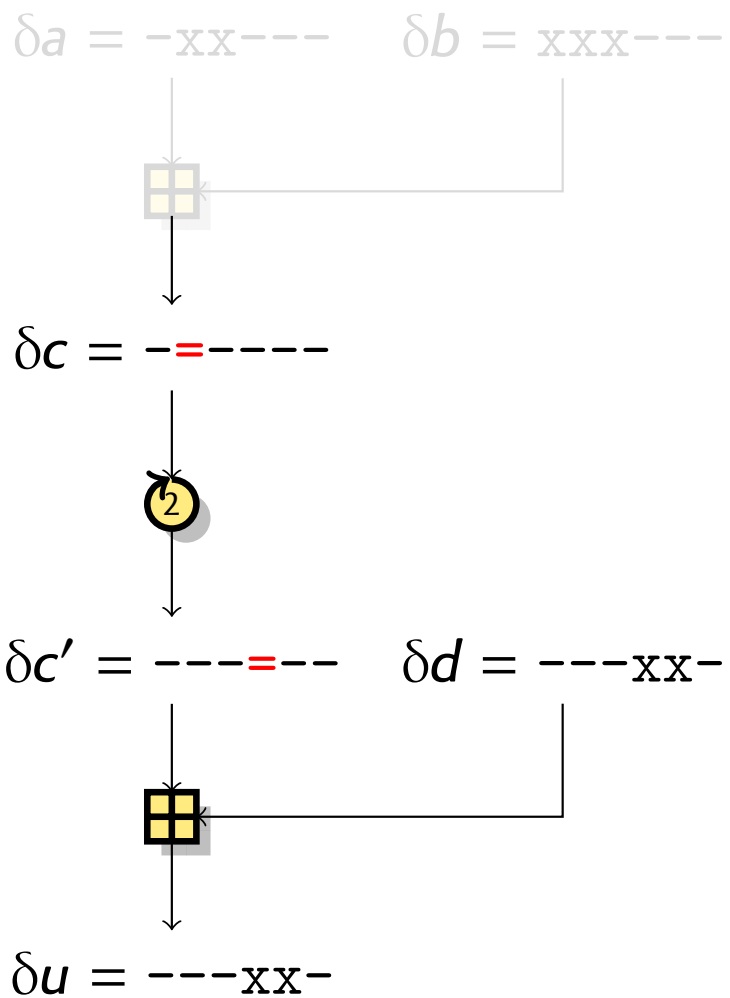
- Each operation has a non-zero probability
- Trail seems valid with signed difference
- Consider the 1st addition
  - Constraint: $c^{[4]} \neq c^{[5]}$
- Consider the 2nd addition
  - Constraint: $c'^{[2]} = c'^{[3]}$
- Incompatible!
  - Detected with the multi-bit constraints

Carry incompatibility



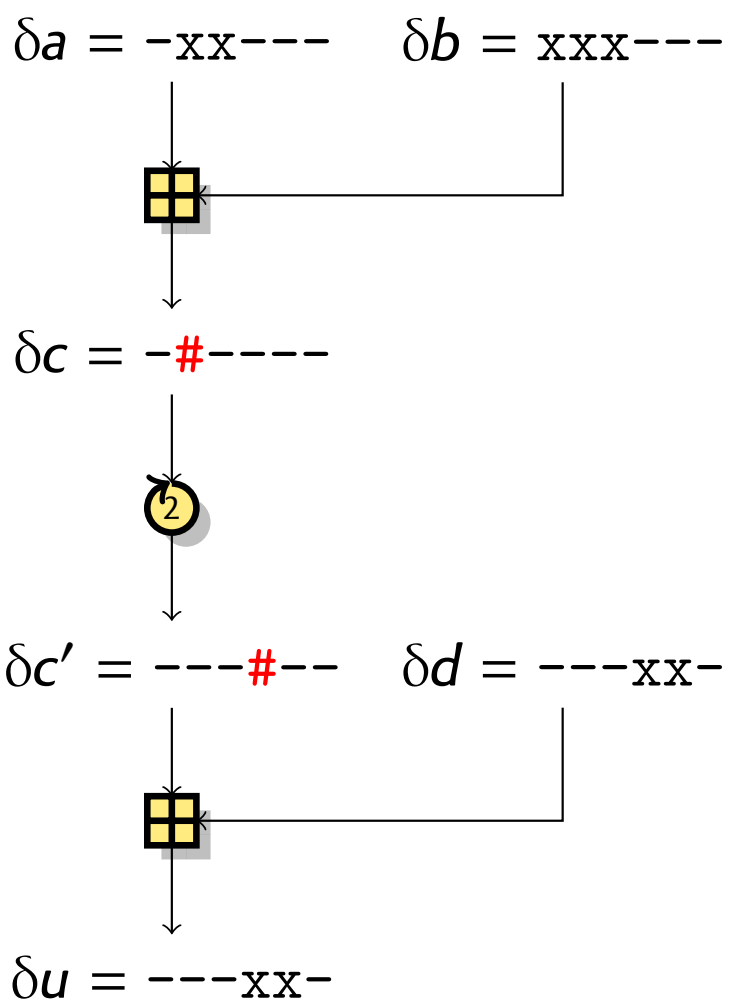
- Each operation has a non-zero probability
- Trail seems valid with signed difference
- Consider the 1st addition
  - Constraint: $c^{[4]} \neq c^{[5]}$
- Consider the 2nd addition
  - Constraint: $c'^{[2]} = c'^{[3]}$
- Incompatible!
  - Detected with the multi-bit constraints

Carry incompatibility



- Each operation has a non-zero probability
- Trail seems valid with signed difference
- Consider the 1st addition
  - Constraint: $c^{[4]} \neq c^{[5]}$
- Consider the 2nd addition
  - Constraint: $c'^{[2]} = c'^{[3]}$
- Incompatible!
  - Detected with the multi-bit constraints

Carry incompatibility



- Each operation has a non-zero probability
- Trail seems valid with signed difference
- Consider the 1st addition
  - Constraint: $c^{[4]} \neq c^{[5]}$
- Consider the 2nd addition
  - Constraint: $c'^{[2]} = c'^{[3]}$
- Incompatible!
  - Detected with the multi-bit constraints

Incompatibilities in Boomerang Characteristics

- For a Boomerang attack, we usually assume that the characteristics are independent

- We are building a quartet $X^{(0)}, X^{(1)}, X^{(2)}, X^{(3)}$:

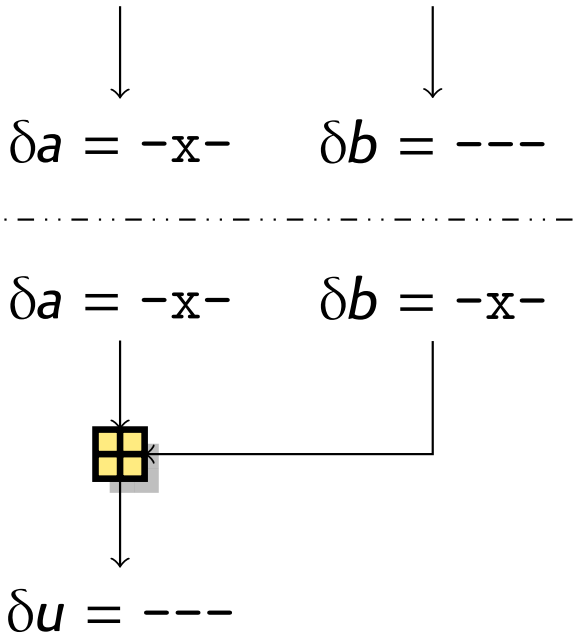
$$X^{(1)} = X^{(0)} + \alpha' \qquad X^{(3)} = X^{(2)} + \alpha'$$
$$X^{(2)} = X^{(0)} + \gamma \qquad X^{(2)} = X^{(1)} + \gamma$$

We expect:

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

- But these events are not independent! [Murphy, TIT 2011]

Boomerang incompatibility



$u = a + b$

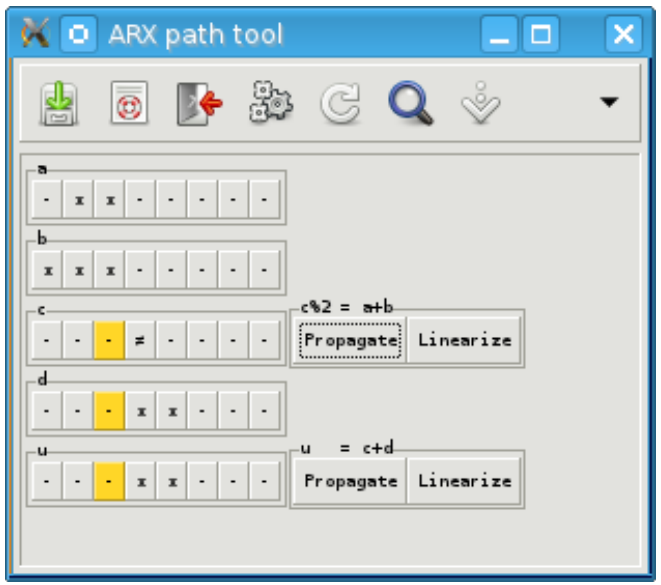
Top path: $(a^{(0)}, b^{(0)}; a^{(2)}, b^{(2)})\ (a^{(1)}, b^{(1)}; a^{(3)}, b^{(3)})$

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

- Pattern appears easily with linearized trails
  - Blake [Biryukov & al., FSE '11]
  - Skein [Chen & Jia, ISPEC '10]
- Impossible to satisfy

Graphical tool

- To study more complex cases, we have a graphical tool
- We can manually constrain some bits and propagate.



Verifying characteristics

Several published attacks are invalid.

- Boomerang attacks on Blake [Biryukov & *al.*, FSE 2011]
  - Basic linearized trails, with MSB difference
  - Proposed attack on 7/8 round for KP and 6/6.5 for CF do not work
  - 7-round KP attack can be made with the 6-round trail
  - 8-round KP attack and 6/6.5-round CF attack can be fixed using another active bit (non-MSB)
- Boomerang attacks on Skein-512 [Chen & Jia, ISPEC 2010]
  - Basic linearized trails, with MSB difference
  - Proposed attacks do not work on Skein-512
  - Similar trails work on Skein-256 [Leurent & Roy, CT-RSA 2012]
  - Can be fixed using another active bit [Yu, Chen & Wang, SAC 2012]
- Near-collision attack on Skein [eprint 2011/148]
  - Complex rebound-like handcrafted characteristic
  - Path is not satisfiable

Outline

Automatic constructions of differential characteristics

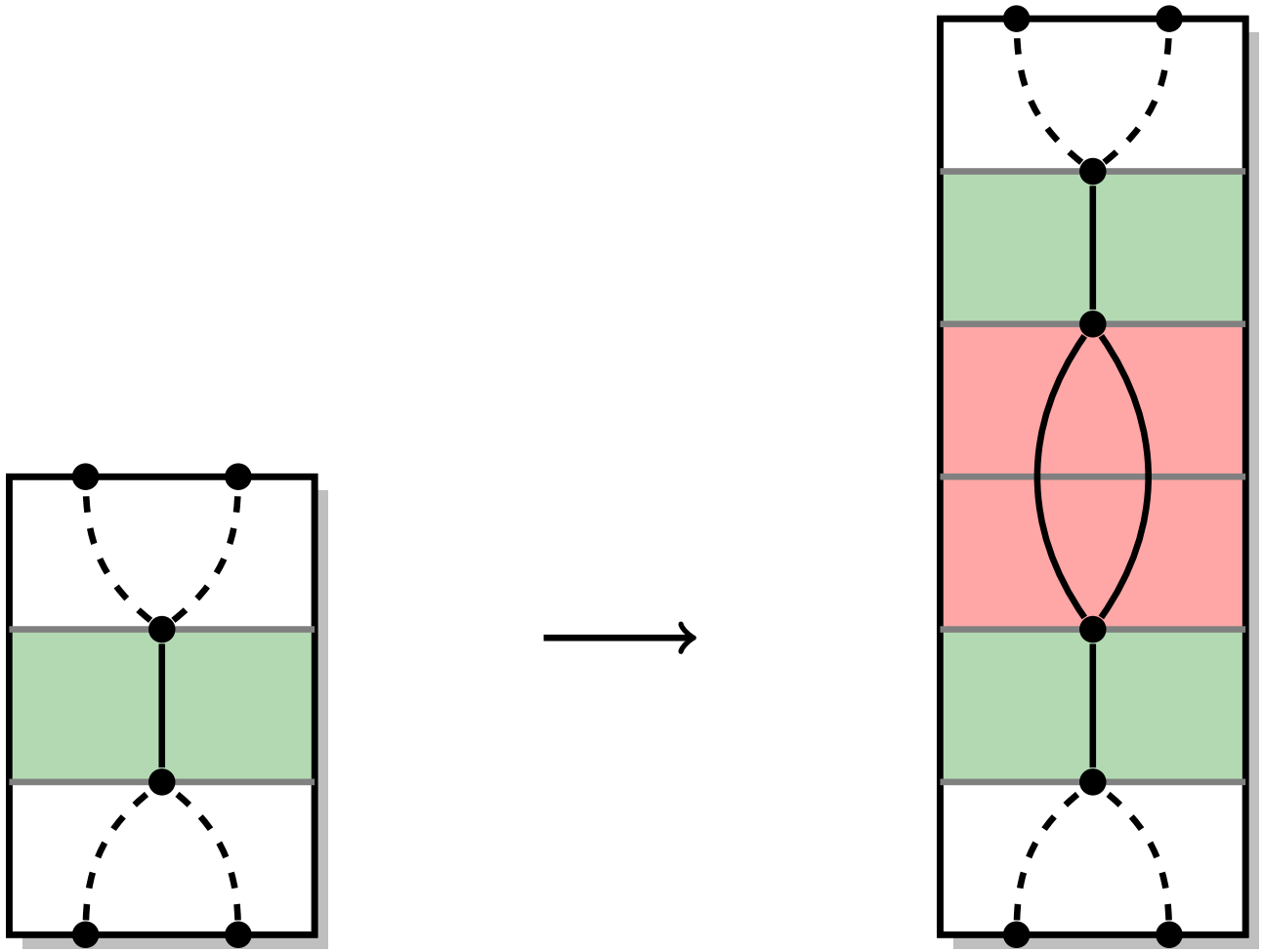
- Several algorithms have been given for MD/SHA function
  - MD4 [FLN, NIST HW 2007]
  - MD5 [Stevens *& al.*, CRYPTO 2009]
  - SHA-1 [De Cannière & Rechberger, Asiacrypt 2006]
  - SHA-2 [Mendel, Nad & Schläffer, Asiacrypt 2011]

- Leads to powerful attacks using special characteristics
  - HMAC-MD4 key recovery [FLN, CRYPTO 2007]
  - Rogue MD5 certificate [Stevens *& al.*, CRYPTO 2009]
  - Attack against combiners [Mendel, Rechberger & Schläffer, Asiacrypt 2009]

- Pure ARX designs seem harder
  - No absorbing Boolean functions
  - The only freedom is in the carry extensions

Main Setting

- We aim to connect two high-probability trails

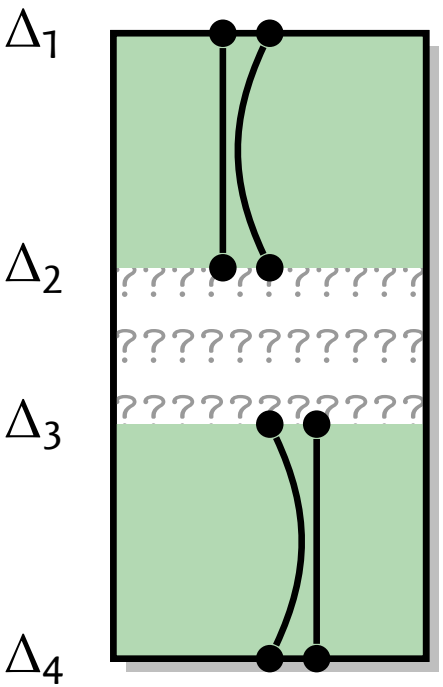


Simple propagation

Connecting trails

Algorithm

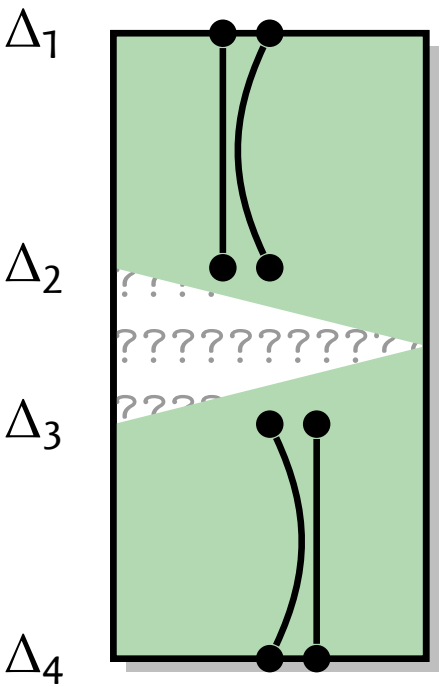
- Guess active bits in the middle and propagate
- Propagation will add necessary constraints (forced guess)



1. Initial characteristic
2. Propagation
3. Guessing
4. Propagation
5. ...
6. Final characteristic

Algorithm

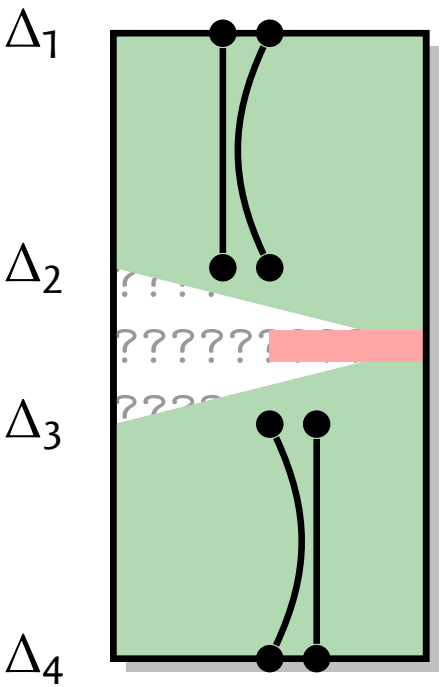
- Guess active bits in the middle and propagate
- Propagation will add necessary constraints (forced guess)



1. Initial characteristic
2. Propagation
3. Guessing
4. Propagation
5. ...
6. Final characteristic

Algorithm

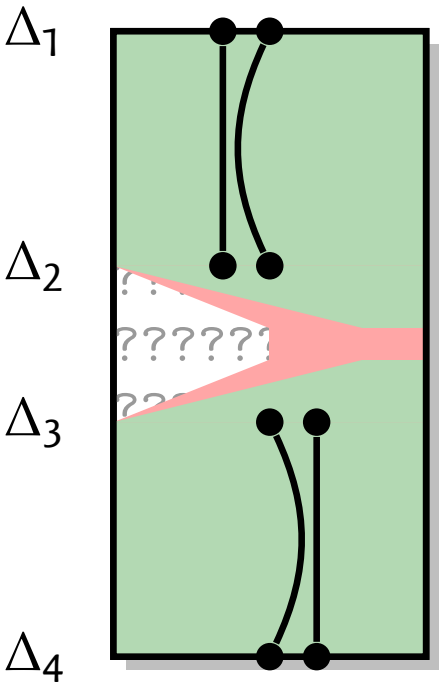
- Guess active bits in the middle and propagate
- Propagation will add necessary constraints (forced guess)



1. Initial characteristic
2. Propagation
3. Guessing
4. Propagation
5. ...
6. Final characteristic

Algorithm

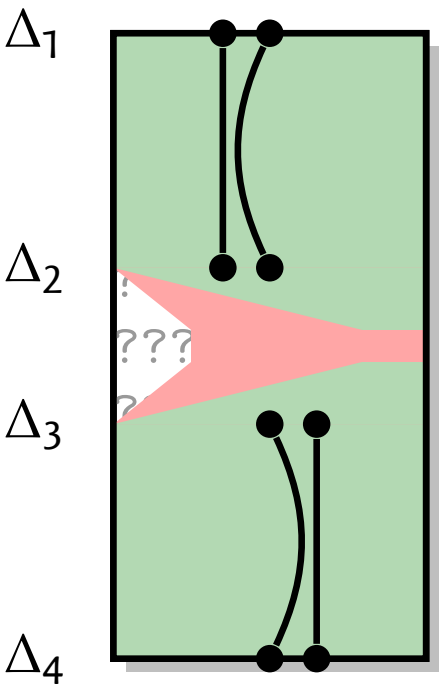
- Guess active bits in the middle and propagate
- Propagation will add necessary constraints (forced guess)



1. Initial characteristic
2. Propagation
3. Guessing
4. Propagation
5. ...
6. Final characteristic

Algorithm

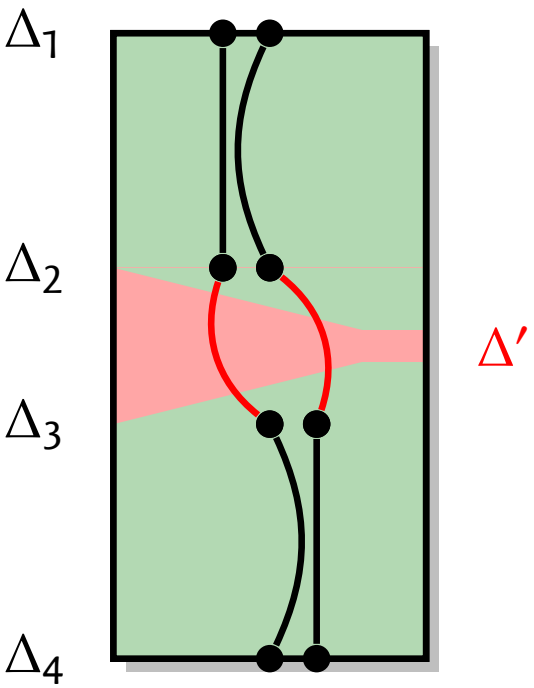
- Guess active bits in the middle and propagate
- Propagation will add necessary constraints (forced guess)



1. Initial characteristic
2. Propagation
3. Guessing
4. Propagation
5. ...
6. Final characteristic

Algorithm

- Guess active bits in the middle and propagate
- Propagation will add necessary constraints (forced guess)



1. Initial characteristic
2. Propagation
3. Guessing
4. Propagation
5. ...
6. Final characteristic

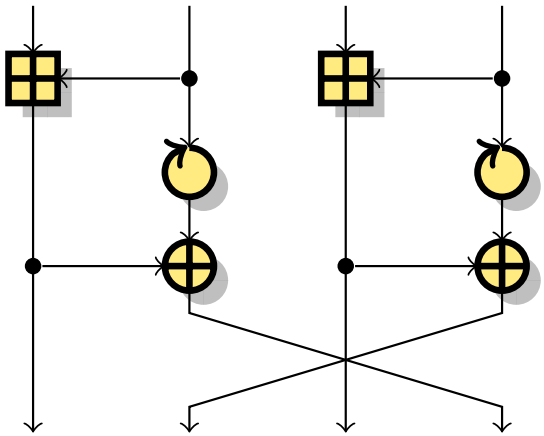
Algorithm

- Guess active bits in the middle and propagate
- Propagation will add necessary constraints (forced guess)

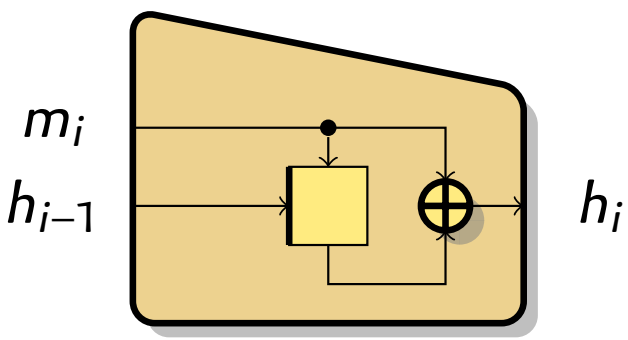
Extra tricks

- We specify in advance the words to be guessed
- We guess from LSB to MSB
- Use backtracking, stop after some time
- When it fails, remember the best guess and restart (hill climbing)

Skein



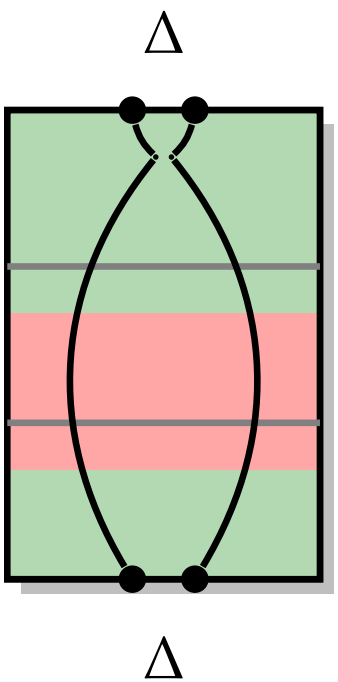
Threefish-256 round



MMO mode

- SHA-3 finalist
- ARX design
  - 64-bit words
  - $\texttt{MIX}_r(a, b) := ((a \boxplus b), (b \lll r) \oplus c)$
  - 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
- MMO mode
  - Chaining value is the key

Collision Attack



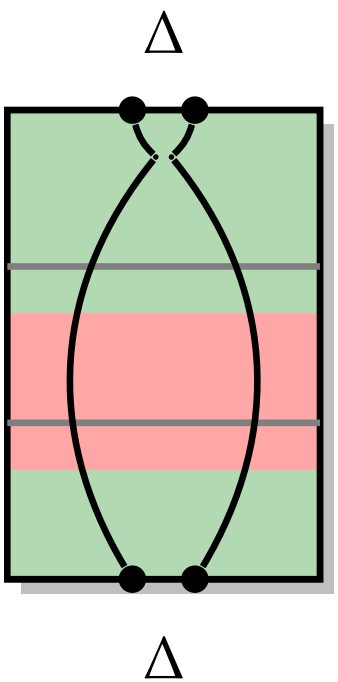
- Trails with no key difference
- Select a small difference Δ in the state
  - Build a trail $\Delta \rightarrow \Delta$
  - Collisions with the feed-forward

- Algorithm works for 3×4 rounds

Limitations

- Dense path
- Many key conditions
  - Only valid for some IVs.
  - Semi-free-start collision.

Collision Attack



- Trails with no key difference
- Select a small difference Δ in the state
  - Build a trail $\Delta \rightarrow \Delta$
  - Collisions with the feed-forward
- Algorithm works for 3×4 rounds

Limitations

- Dense path
- Many key conditions
  - Only valid for some IVs.
  - Semi-free-start collision.

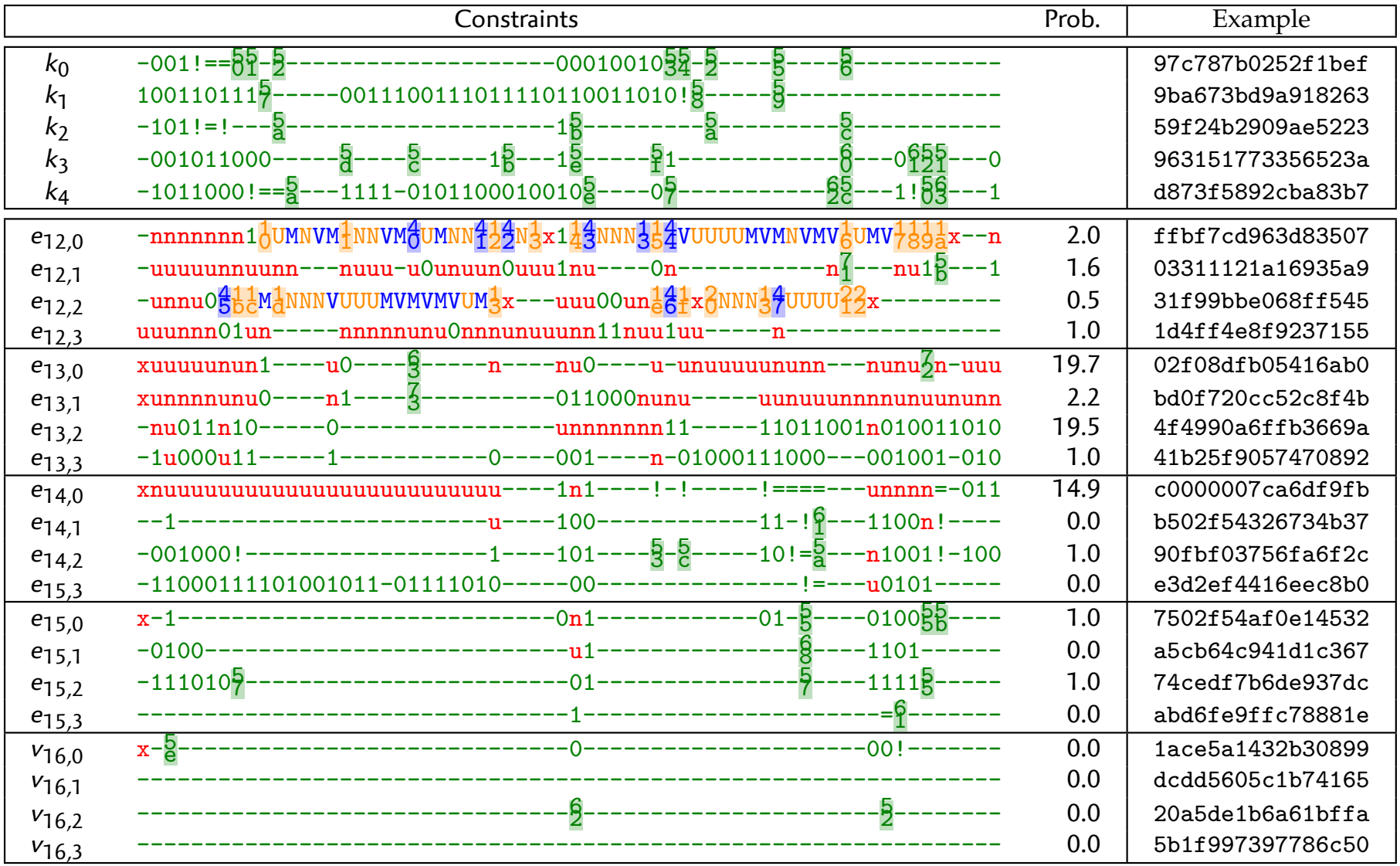
Differential path

	Constraints	Prob.	Example
k_0	-001!==[5]5-[5]---------- 00010010[5]5-[5]----[5]----[5]----------		97c787b0252f1bef
k_1	1001101117[5]----- 00111001110111101100110101[5]!8[5]----[5]----------		9ba673bd9a918263
k_2	-101!=!---[5]a---------- 1[5]---------[5]---------[5]----------		59f24b2909ae5223
k_3	-001011000-----[5]----d[5]----c[5]-----1[5]---1[5]-----f[5]1----------[8]---0[6]55---0		963151773356523a
k_4	-1011000!==[5]a---1111-0101100010010[5]e----07[5]----------[6]5---1![5]6---1		d873f5892cba83b7
$e_{4,0}$	x----------	0.0	55854848e8d2b7fa
$e_{4,1}$	----------	0.0	b45620969e3043f9
$e_{4,2}$	----------	0.0	a0fe049603be00b1
$e_{4,3}$	----------	0.0	4bc235e51a57e884
$e_{5,0}$	x----------	0.0	09db68df8702fbf3
$e_{5,1}$	----------	0.0	d86feb73899182ff
$e_{5,2}$	----------	0.0	ecc03a7b1e15e935
$e_{5,3}$	x----------	0.0	24e70858746a57b2
$e_{6,0}$	x----------	0.0	e24b545310947ef2
$e_{6,1}$	x----------1-----u----------	0.0	6122c59537fb62a9
$e_{6,2}$	x----------	0.0	11a742d3928040e7
$e_{6,3}$	x----------0----------[5]1--------	0.0	82f4a248ea489c96
$e_{7,0}$	----------u-----0-----	2.0	436e19e8488fe19b
$e_{7,1}$	----------1---------0----------101[5]6[2]1---n----------[6]d	0.0	06a1773b59686055
$e_{7,2}$	------[6]2----------0---=-[6]----[6]-[6]----[6]----------1	0.0	949be51c7cc8dd7d
$e_{7,3}$	------n----------0---=-1----0-0011100!----------n-----u[6]e---1	0.0	e6ea92fe1c500c11
$v_{8,0}$	-100[6]8[6]---0----------[5]3----------n---------u----------	1.0	4a0f9123a1f841f0
$v_{8,1}$	-11001n111-----[6]e----u-----u[6]4---1[6]1-----n[5]2----------n---![7]0[5]n1---0	1.3	67d6740b7ff27b70
$v_{8,2}$	-11110n1![5]8---0!==-1000000110105[5]34----0[5]----------[6]1n---10u[5]b---0	1.0	7b86781a9918e98e
$v_{8,3}$	-0010011--n----------0101001111[6]1-u---------u----------	0.4	1367f176a75936cb

Differential path

	Constraints	Prob.	Example
k_0	-001!==5501-52----------00010010552-52----5----5----------		97c787b0252f1bef
k_1	1001101117-5-----00111001110111101100110101!58-----59----------		9ba673bd9a918263
k_2	-101!=!---5a----------1b5---------5a---------5c----------		59f24b2909ae5223
k_3	-001011000-----5d----5c-----1b5---1e-----5f1-----------6---0655---0		963151773356523a
k_4	-1011000!==5a---1111-0101100010010e5----075----------65---1!56---1		d873f5892cba83b7
$e_{8,0}$	10100100000000011101110001001100101010111 0n001101001u10000010011		a401dc4caba69413
$e_{8,1}$	111111n0000001111100u10110u00010101100n1010010001100n10110n01010		fe07c582b348cdaa
$e_{8,2}$	010100n1111110100110110110100011110001011101001101 10n10101u00101		53fa6da3c5d36d45
$e_{8,3}$	1010101100n0111101111001001001101100110010u010000101u01010111100		ab2f7926cc8852bc
$e_{9,0}$	101000n0000010011010u00111u01111010111n011n011110110000110n11101		a209a1cf5eef61bd
$e_{9,1}$	100001n000u011110010101001u000101100u00011n0011101101011001011 10		860f2a42c0e76b2e
$e_{9,2}$	111111n100n010011110011011001010100100 1001u1101111000000000u00001		ff29e6ca925bc001
$e_{9,3}$	010100110110n0010000n101u0u1110101101101 10u0u10111011110u0n11100		53690d1d6d85de3c
$e_{10,0}$	0010nu0000u11000110un1000u01001000unn1n11n0101101100110011n01011		2818cc121fd6cceb
$e_{10,1}$	0u1010n0uunnu1000010uuun1nnnnnu1n1uuunuuu11n0101u0n0101n000000n		2a3421fdc53a9581
$e_{10,2}$	010100n0nuunu010111nuunn1nn0unnnnnnnnnnnnn10u00110011110u0111101		5292f3e7ffe19e3d
$e_{10,3}$	1001nu1011n1000010nun1u01n10000010nnn0111n111010n10000n010u11101		9af0ace0bbfac29d
$e_{11,0}$	un0nuunu0n00n10unnn0nn10uu0un1n11n100n0nu00nu001unnu00nuu1n01nuu		524cee0fe511626c
$e_{11,1}$	0u01u1n101uuu0u1001111unun u1uu10010un01101110u0unuuuuuuuu11uu001		17413d524b708061
$e_{11,2}$	1110n1u1n0uuu01110n000uunnu0100010nnn01111unnnuuu1nuuuuun1u11010		ed83a0c8bbdc60da
$e_{11,3}$	1un01100101u1nn00111u011unuu01u100100n0n10uuu10uu111nuuuu1n11100		acae73452584787c
$v_{12,0}$	u11010un1000nnn00un0n0n1unnu001uuu1100uu1uuuuuu1nnnu00nu1100n10n		698e2b623081e2cd
$v_{12,1}$	u010nun0nunn1101000nnun1nu0nnuu00n110nuu10101110101n00011n110010		2abd1b9874aeb1f2
$v_{12,2}$	10u1n01uuu11001uuuu1u1u0uu0u11u11n100u0nun10000u1101nuu101010110		9a32140de160d956
$v_{12,3}$	nu000u011u10nuu1100u0uun0unu10n1un0111nunu01000n1110111011101111		81a9812b5e91eeef

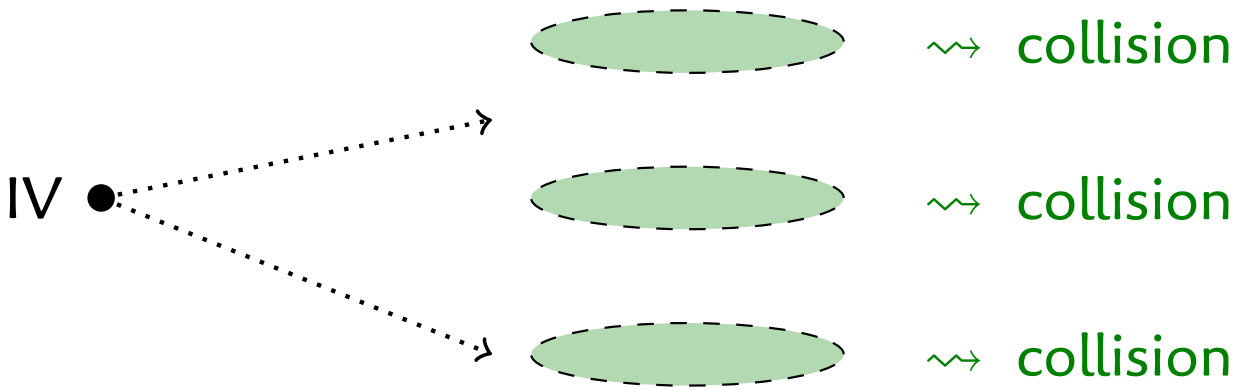
Differential path



Full Collision Attack

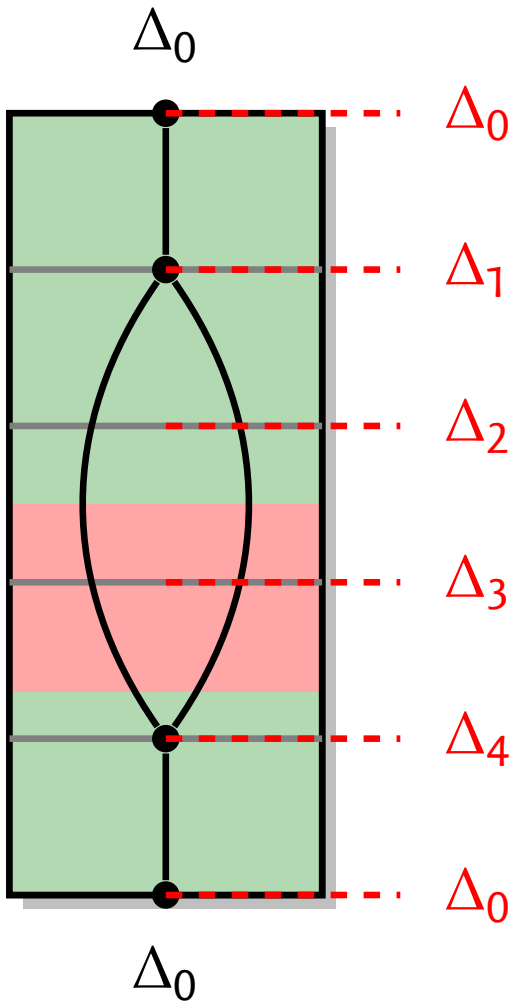
- We build a collision characteristic valid for 2^{70} keys for a cost of $\approx 2^{40}$

1. Build many characteristics
2. Use random message blocks to reach a valid CV for one path.



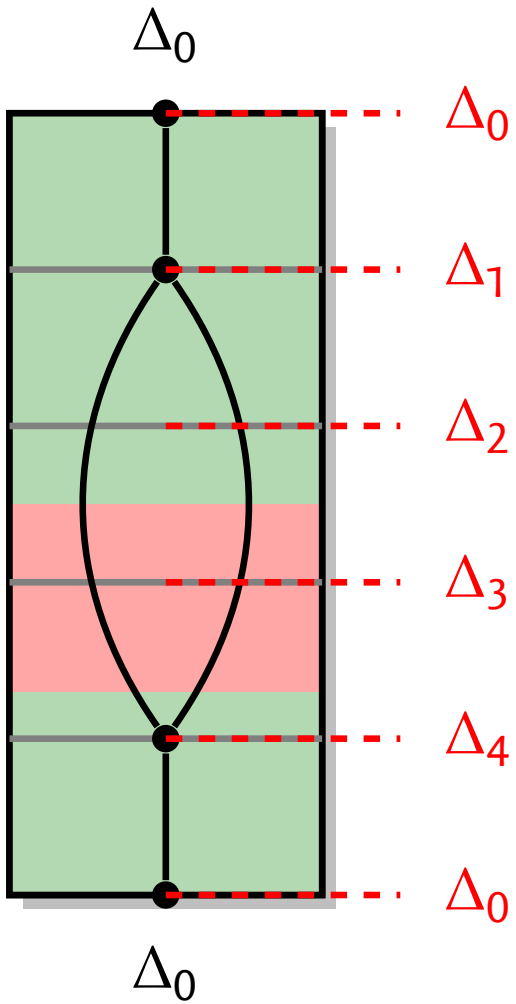
- Collision attack for 12-round Skein-256 with complexity $\approx 2^{114}$

Free-start Collision Attack



- Trails with small key difference
- This allows inactive rounds
- The key schedule repeats after 5 block
  - Collisions with the feed-forward
- Algorithm gives 20-round characteristics
- Practical attack

Free-start Collision Attack



- Trails with small key difference
- This allows inactive rounds
- The key schedule repeats after 5 block
  - Collisions with the feed-forward
- Algorithm gives 20-round characteristics
- Practical attack

Differential path

	Constraints	Prob.	Example
k_0	-1-1-11110016 5 a9-----111 6 a--1111!--5 d -010-1001001-0 6 1----111!--01!==-		5f977cfdd64d2f57
k_1	x0-1010110000!-1100100011001001 10-000-1000101!=- 7 1----011 5 3--110!=-		35839193022be6f4
k_2	x0-0-101-=-0-------010 6 a--0010-----000-=! 7 4---0-5 1-----010---00 6 4---		05e168930700458f
k_3	-11011110!-0!== 5 8---1010101111-1110001-110 6 8--11111-01101101116 4--0		6f47d57f8b6f9b78
k_4	00011011!=-0001101001011010 11-00111100!== 7 1--0101000011010111 5 b--0		1b634b58f1f50d76
$e_{4,0}$	x----------	0.0	dd5113862e4682f2
$e_{4,1}$	x----------	0.0	976b12a915df1438
$e_{4,2}$	----------	0.0	2682e7ab2b50853e
$e_{4,3}$	----------	0.0	c21caeeb08ac00af
$e_{5,0}$	----------	0.0	74bc262f4425972a
$e_{5,1}$	----------	0.0	f9c797c9b7c5d83b
$e_{5,2}$	----------	0.0	e89f969633fc85ed
$e_{5,3}$	----------n----------	0.0	26979807350b410f
$e_{6,0}$	----------	1.0	6e83bdf8fbeb6f65
$e_{6,1}$	----------75 4d----11-----u----76 7e-----x------ 5 5-----!----------	0.0	76b75dcddd173495
$e_{6,2}$	----------!6 c------6 8--1----------n----------	1.3	0f372e9d6907c6fc
$e_{6,3}$	---------==5 9---10-------!---7 3-----7 8----5 a--!=6 4---=----------10 6 1--	0.7	188d43891e190294
$e_{7,0}$	---5 4----------66 41----0u-----1----706 90e----------5 -----=5 6----	2.5	e53b1bc6d902a3fa
$e_{7,1}$	--==7-766 a bd3---7 8-------!!6 c---!=---------01 0 1-7 c-6 2-------!!6 5---1000---	0.0	c583f4662226eac0
$e_{7,2}$	0010 6 a-=55 cd---01000-11001000!5 2---!-000011n0 5 e--0005---0100110 6 4---0	0.3	27c472268720c990
$e_{7,3}$	10110-00=---00!==-000110101n---1-n10111u01-10n1 6 1---111110010--00	0.0	b0e1c6b1ee76ff28
$v_{8,0}$	-0-0-018 6 8-6 a-5 a-------!==5 0--01u1-----==5 0 -655 83b---5 e--------==5 a---==6 c---	2.6	aabf102cfb298eba
$v_{8,1}$	-u11011u11-10u0 6 1---00111111n0-001n000-01! 7 1--0n!==-001110000u 6 4--1	0.1	36d0c7f0c5760e09
$v_{8,2}$	1101100010-0011000111000110n!-000n110101! 5 2--0n111100100010116 4--0	0.0	d8a638d87597c8b8
$v_{8,3}$	-=-!-00u10011 5 b-----110!--11u11--5-===-1010101-1 5 b----101 5 3--011!=-	0.4	8899faec3eaa7adc

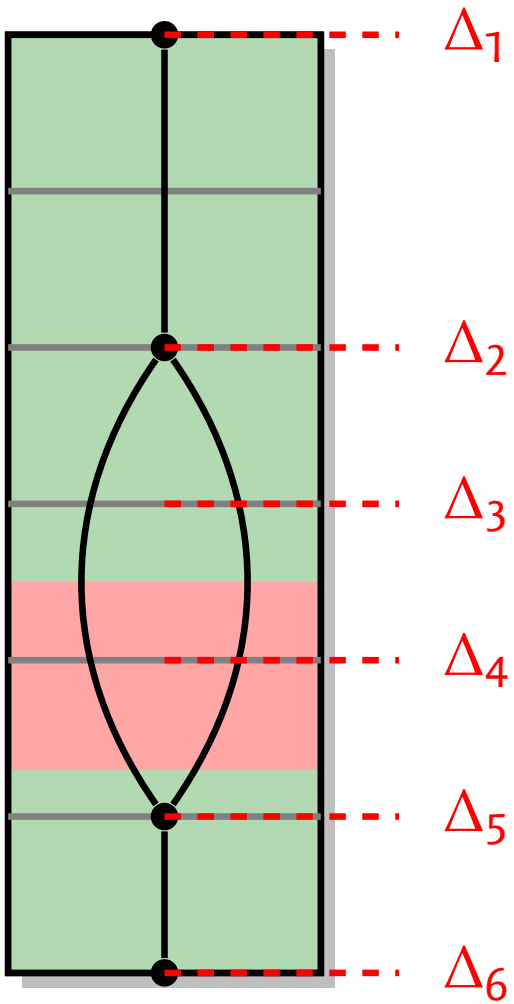
Differential path

	Constraints	Prob.	Example
k_0	-1-1-1111001$^{65}_{a9}$-----111$^{6}_{a}$--1111!--$^{5}_{d}$-010-1001001-0$^{6}_{1}$----111!--01!==-		5f977cfdd64d2f57
k_1	x0-1010110000!-1100100011001001l0-000-1000101!=$^{7}_{1}$----011$^{5}_{3}$--110!=-		35839193022be6f4
k_2	x0-0-101-=-0-------010$^{6}_{a}$--0010-----000-=!$^{7}_{4}$---0-$^{5}_{1}$-----010---00$^{6}_{4}$---		05e168930700458f
k_3	-11011110!-0!==$^{5}_{8}$---1010101111-1110001-110$^{6}_{8}$--11111-0110110111$^{6}_{4}$--0		6f47d57f8b6f9b78
k_4	00011011!=-0001101001011010l1-00111100!==$^{7}_{1}$--0101000011010111$^{5}_{b}$--0		1b634b58f1f50d76
$e_{8,0}$	n0110000101000000111100011u0000000000100010100111010100010010001		b0a078c00229d449
$e_{8,1}$	1u10011u00011u0010011101011n00000n01000011100n0110101001100u0001		a6189d7050e5a981
$e_{8,2}$	1111010000001001100001000l1n00010n10011110001n0011010110001010110		f4098431678cd62e
$e_{8,3}$	1110100u0011000101110111111u1010000101001111011110101010001101l1		e83177ea14f7aa35
$e_{9,0}$	0n01011u10111u0100010110001100000n01001100001n1101111101110u1010		56b91630530f7dca
$e_{9,1}$	1010101n110n0000111010001110110l1n01011010110u010110100u01010010		abd0e8ecd6b16852
$e_{9,2}$	1101110u0011101011111100000110110n11110010000n0010000000001100011		dc3afc1b7c848063
$e_{9,3}$	0n11000n11100101000u00100000100n0u1110010110111l000n01000n0u110u		71e50209396f144c
$e_{10,0}$	uu000010nuuu1u0111111111000l110nu0101001110000001110011u00un1100		0289ff1d29c0e61c
$e_{10,1}$	nnu101n0n1nnnn0uuunn010000n00uuu101001n1n00uuu01010u10101000unnn		d6fc3420a7814a87
$e_{10,2}$	0n00111000unnnnnnnnn111000100nuu10110101111nuu11100n010010n0nnnn		4e1ffe24b5f394af
$e_{10,3}$	nu0u0111nunu0u11010u001u0001001nn010011100001n011000110n0uuu1010		87a34213a70d8d0a
$e_{11,0}$	11u1nuun10uuu11uuunn0011001nn10nn10nuuunu10uuu100unn000u101u0unn		d986333dd14230a3
$e_{11,1}$	n1un1u00u1uu1n1u0n0u101unu1nnn1n1nnn1n1uunu0u01n00nn0010uu0111nu		d84e4abffe43321e
$e_{11,2}$	n1un0101n1uu0u1101u0000u0011nu0uu101110nuuuuu00100nu000n1u1110un		d5c340385d0121b9
$e_{11,3}$	11uunu01n1unu101n1nnuun1100n100un00nu0101un0nu0n01u0nnnunun1nuun		c9d5f39892a94eb9
$v_{12,0}$	nu1n00u11nu1unu0u1n111un1nnnn1un110u1n1n1000un0n01n00un01n0u0001		b1d47dfdcf8562c1
$v_{12,1}$	nn00n010nu1nu0001nn00100n110n00nn10nunun00010100uu000011011u000u		cab0e4e9d5140360
$v_{12,2}$	nu011n1n1uunnu0nu01nuu1n1nu1uuu01110n111101u1unu011100u001n100n0		9f9933d0efaa7072
$v_{12,3}$	1u111u00uu011n01u01un010uu0u0u1uu0u0u11111n0u0nn00nuuuu100u11un0		b81d2a0207e3211a

Differential path

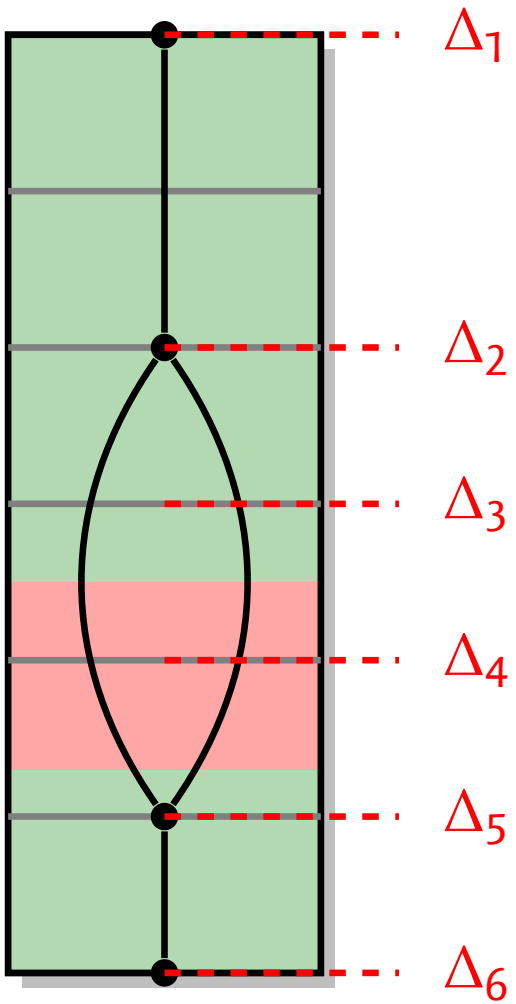
	Constraints	Prob.	Example
k_0	-1-1-11110016559-----1116a--1111!--5d-010-1001001-061----111!--01!==-		5f977cfdd64d2f57
k_1	x0-1010110000!-1100100011001001l0-000-1000101!=71----0115--110!=-		35839193022be6f4
k_2	x0-0-101-=-0-------0106a--0010-----000-=!74---0-15-----010---0041---		05e168930700458f
k_3	-11011110!-0!==58---1010101111-1110001-1108--11111-01101101114--0		6f47d57f8b6f9b78
k_4	00011011!=-0001101001011010l1-00111100!==71--0101000011010111b5--0		1b634b58f1f50d76
$e_{12,0}$	xunu00u!==-nn1482V2x10unn0nnnn-un01un4xnu1b6--5926>=-1nnnn000nn41--1	0.0	211c537d5af4fe39
$e_{12,1}$	nnnuu1nuuu-nunuuuunnuuuuunuuu-nunnuuu1nn0e5--1001uu01000011unb5--u	0.0	e6143042c70910d6
$e_{12,2}$	22<-1-n1n0u15b7414U2x--nua427x-nuu1g8x-2gxuuu-unnnnnna2x24bbNN13xnnnb222xu0e612>-	0.5	ff30b0cec5f79fc9
$e_{12,3}$	-n-unnunnunuuu-ununnnunnnuunununu-uun-nuuuuun1nn----nuu0--u100u-	0.3	eda0bb950a0f0811
$e_{13,0}$	xuuu0nnnuunnuuu61---000111nuuuuuuu0100-unnnnnnnnnuu-uunnnnuuuunnnn	7.1	073083c021fe0f0f
$e_{13,1}$	xnnn1uuunnuunnn5d---11100010000000-00101101000-nnnnunuuuunnnnuuuu	5.4	f8cf7c400b47d0f0
$e_{13,2}$	x110-1001101u00----unn0--n100u--n-unuuuuuuuuu-101010011111011010	11.2	ecd16c63d006a7da
$e_{13,3}$	-000-0101011n11----1000nnnnuuuunnuu---000011u0100-10011101101111	0.0	82be91e18c32276f
$e_{14,0}$	xuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuu-1!=-0101000-!==-01111111111111	3.6	000000002d45dfff
$e_{14,1}$	-00----0100100011110011---000=--0d5---100100=-1==-11-111-1100010	0.0	8691e6867e4e3762
$e_{14,2}$	x116-111100unnnnnnnnnnn--1000!--u65!---000011!-0!=-00111101001001	4.4	6f8ffe455c38cf49
$e_{15,3}$	-11101000011110000-111n000110011164----1001010-!==-0-11010010111-	0.3	f43c3e33f255dd2e
$e_{15,0}$	-00----0100100011110011---000=--n5566---11100771-16=8-01-111-1100001	0.6	8691e686ab941761
$e_{15,1}$	-1!61-1110014----10-0100--==-7d---u74----010--11-1=6-!-00110111-==-	0.6	ef30a90e0533a378
$e_{15,2}$	x1!62-0111100110000-1110--111!=--ub5----1010005d-1!56-!-11000111!==-	0.7	63cc3c794e8eac77
$e_{15,3}$	-00----01----0111010000---011---n-59---11!---56-62!6-!--====-0-111-0	0.0	0c8ba11cb26d2fbc
$v_{16,0}$	-50-----11!5c-----!=-!==----5c----------0!a-----e85--0--01!-1!56-00-	0.0	75c28f94b0c7bad9
$v_{16,1}$	-------73------56!----------6a----------------77e--64--!-----01---	0.0	c23af22a0c707d2f
$v_{16,2}$	x5603----086----1111!-1116---0345---------a1c-----38e--6c--!60--0-1053--	0.0	7057dd9600fbdc33
$v_{16,3}$	x------01----------=!----56----------1=7f----------01---!-----	0.0	70f12cec5ff713d7

Free-tweak Free-start Near-collision Attack



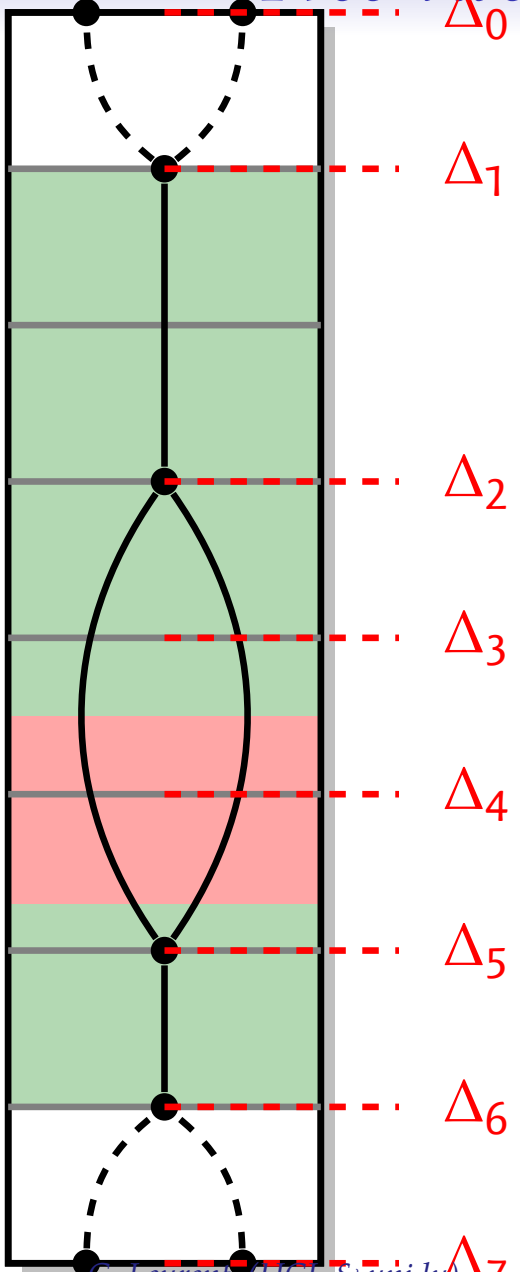
- Trails with small key difference and small tweak difference
- This allows one round with inactive subkeys
- Controlled characteristic for 24 rounds
  - 5 active bits in the output ($\Delta_1 \oplus \Delta_6$)
- Algorithm work for the middle rounds
- Practical attack
- Can be extended to partial-collisions for 32 rounds

Free-tweak Free-start Near-collision Attack



- Trails with small key difference and small tweak difference
- This allows one round with inactive subkeys
- Controlled characteristic for 24 rounds
  - 5 active bits in the output ($\Delta_1 \oplus \Delta_6$)
- Algorithm work for the middle rounds
- Practical attack
- Can be extended to partial-collisions for 32 rounds

Free-tweak Free-start Near-collision Attack



- Trails with small key difference and small tweak difference
- This allows one round with inactive subkeys
- Controlled characteristic for 24 rounds
  - 5 active bits in the output ($\Delta_1 \oplus \Delta_6$)
- Algorithm work for the middle rounds
- Practical attack
- Can be extended to partial-collisions for 32 rounds

Differential path

	Constraints	Prob.	Example
k_0	-1101101101100100010110011100011000 0!===6 2-0100000000000010001!-10		edb22ce30810011a
k_1	0000100000010100001011101001000001000100 5 d-1100000000010101001010		08142e9044b0054a
k_2	-0!6 0-5 2-5 4--5 2------01111016 1-0!-111!===5 6-6 8---08 0--11--011111!6 2-10-11		1e06bd5779535f97
k_3	>====6 0-5 c--6 3----11110011110000101111 00-0111000101--11100006 1-1!-10		82a5e785e5c5b836
k_4	x-!5 8------5 e------10000116 1-1=-011!==--------08 0--10--1110014 4--1--11		62d4437b79caf9d3
t_0	u0000000000000000000000000000000000000000000000000000000000000000		0000000000000000
t_1	0000000000000000000000000000000000000000000000000000000000000000		0000000000000000
t_2	u0000000000000000000000000000000000000000000000000000000000000000		0000000000000000
$e_{12,0}$	x----------	0.0	87ad0104ef83d5a7
$e_{12,1}$	----------	0.0	da552a21bd36c3ad
$e_{12,2}$	----------	0.0	d9fd1666ce2e3e47
$e_{12,3}$	----------	0.0	980fe1445c75f41a
$e_{13,0}$	x----------	0.0	62022b26acba9954
$e_{13,1}$	----------	0.0	cae71f9e1abbf0e9
$e_{13,2}$	----------	0.0	720cf7ab2aa43261
$e_{13,3}$	x----------!6 2----------8 2-------	0.0	217846a1f70e3300
$e_{14,0}$	x----------5 6----------	0.0	2ce94ac4c7768a3d
$e_{14,1}$	x----------111!5 6-------=----------0010a366--u5 8--01------	0.0	17ef213dc2826776
$e_{14,2}$	0 2------5 1--00!5 a---!-1176 1a65 1----------8 0---5 d-1!5 2----6 b--5 c----------	0.0	93853e4d21b26561
$e_{14,3}$	n------0110100---0-110==6 2-------0---=---10010 6 5---0--=----------	0.0	d0d3387d00910c93
$e_{15,0}$	----------101!5 6----5 8--!5 8----------11u0a366--15 5--11	1.0	44d86c0289f8f1b3
$e_{15,1}$	!=5 d-------0---=---0100116 5---0--01----------n------1000101---1-010	0.0	7b18528906b7453a
$e_{15,2}$	01100!=6 c--0110000111011011 00!-!=-0100010 1-0000110111000111110100	0.0	645876ca224371f4
$e_{15,3}$	100111u5 b--0001111101000010 00!-10-1111110111100101111n00000n01110	0.0	9c87d0867ef2f82e
$v_{16,0}$	--=6 c------1------0111110 5-d-8 3-0-011!=7 1-------n7--00--11u1133 56--0--01	0.5	bff0be8b90b036ed
$v_{16,1}$	-11111n0000100101011n11101n11110001111u1 5-d-1100011011n01010n0!-00	0.0	7e12bf7e3db1baa4
$v_{16,2}$	000000u0111000000100011101010000101000 0!6 2-1101100110n01000n00010	0.0	00e04750a1366a22
$v_{16,3}$	-013-5-5--u------1111011 5-d-11-00111101 6-6---n5--=6--10u100!5 6-0!-00	0.9	b947fbb1eba86464

Differential path

	Constraints	Prob.	Example
k_0	-1101101101100100010110011100011000 0!===6-2-0100000000000010001!-10		edb22ce30810011a
k_1	0000100000010100001011101001000001000100 5-d-1100000000010101001010		08142e9044b0054a
k_2	-0!6-5-5--5------01111016-0!-111!===5-6---08--11--011111!6-2-10-11		1e06bd5779535f97
k_3	>====6-5--6----11110011110000101111100-0111000101--11100006-1-1!-10		82a5e785e5c5b836
k_4	x-!8------e------10000116-1=-011!==-------08--10--1110014--1--11		62d4437b79caf9d3
t_0	u0000000000000000000000000000000000000000000000000000000000000000		0000000000000000
t_1	0000000000000000000000000000000000000000000000000000000000000000		0000000000000000
t_2	u0000000000000000000000000000000000000000000000000000000000000000		0000000000000000
$e_{16,0}$	u0100010110001010000001000000111000010100 1n110110011u00011000000		22c502070a7b30c0
$e_{16,1}$	011010n1110001001110n10001n00001010001u1110000011011n01110n11110		6bc4ec6145c1bbbe
$e_{16,2}$	u00010u011110100011101011110000011100101111001100110n11101n01100		08f475e0e5e66f6c
$e_{16,3}$	1101011101u011101011100100001001011001011n110111100u01111111111		d74eb90964fbc3ff
$e_{17,0}$	n00011n0100010011110n11001n01000010100u000n1110011101100001n11110		8e89ee68503cec7e
$e_{17,1}$	u10110u101u0101001001010000u100011000n00100u1110111100100000100101		594a4a11891de425
$e_{17,2}$	n11000u001u0001100101110111010100100101011n0001000110011011n01011		e0432eea4ae2336b
$e_{17,3}$	n01101011001u0011011n111u0u1100000111111011u1u01101110110n0u01111		b591bf183ed3768f
$e_{18,0}$	1110011111u101000011n0000111100111011n0u1010110101101000010n00011		e7d43879d95ad0a3
$e_{18,1}$	1u0010nu10n1n111110un101u1nn110u1u1110011100n0000u0u1111u001011n		8abfcd7cb9c80f17
$e_{18,2}$	100101u111u1u1001110n110u0u0001010001001101 1u10110101001n1111010		95d4ee0289b5a9fa
$e_{18,3}$	1010010110u0u00110n0n1u0110111010111n00001000010u10000u101n11101		a581acdd7842417d
$e_{19,0}$	un1100nu100nu100000001unn1nn011u1u0nuun100nuu01unnu1nnnnn0n110nu		729405f69322dfba
$e_{19,1}$	01111uu10uun01n1nn1u0nnnu10uununnu0u00uuun0nnu1nu0110nnuu0u0n111		7917e745805b360f
$e_{19,2}$	00111un1unun0110nuun1unun11uuuuuuuuuu0unnnnn0nnnn1101unnu1n10111		3b569ae001f7eb77
$e_{19,3}$	n10un1uun1uun00011100uunn11nu00n000nnuu0n11uu1111uuuu0uuu1u1nnuu		ccc8e1f118e7805c
$v_{20,0}$	nn1un0nn1u1un0n1nn1un1u1001nnnu000un00nnun1nnnnuuuun0n011100n00n		ebabed3c137e15c9
$v_{20,1}$	0u01unu011101nnn0n1nu1n1un00n0u01uu0uu1n1n0uuunnunun010111nnu0u0		14ef774883c355f0
$v_{20,2}$	u00unu0u0001n11101n1nnuu11unu00n000nn0n01n0nnnnn011010nn110nu01n		081f7cd11adf6bd3
$v_{20,3}$	110u1u0n01010111100u0u10nnuu01n000u01n0000uu1nuuu1101010u0010u11u		c957058c1818d426

Differential path

	Constraints	Prob.	Example
k_0	-1101101101100100010110011100011000 0!===6-2-0100000000000010001!-10		edb22ce30810011a
k_1	0000100000010100001011101001000001000100 5-d-1100000000010101001010		08142e9044b0054a
k_2	-0!6-5-5--5------011110161-0!-111!===5-6-8---08--11--011111!62-10-11		1e06bd5779535f97
k_3	>====6-5--6----11110011110000101111000-0111000101--11100006-1!-10		82a5e785e5c5b836
k_4	x-!58------5e------10000116-1=-011!========08--10--1110014--1--11		62d4437b79caf9d3
t_0	u0000000000000000000000000000000000000000000000000000000000000000		0000000000000000
t_1	0000000000000000000000000000000000000000000000000000000000000000		0000000000000000
t_2	u0000000000000000000000000000000000000000000000000000000000000000		0000000000000000
$e_{20,0}$	xn011uunununnnn1u0uunnun0000nn1n100un10nn5d-uunnnuuuun0n1011114b<-1n	0.0	d95e1a1f1b8e16e3
$e_{20,1}$	uuunnnunuuuuuunnnunuu10nnn01n0uunnuun0uu16-nnuunnunun101100n1nun0	0.0	1d03a5d8c8735b3a
$e_{20,2}$	-01c24N4dUUc22x----0n1n0414<62-1u-uuued4ef21xa4U2xnf2N33>--0010nn23463332>-nu	0.8	26263a289432cb6a
$e_{20,3}$	u10u1u-n--73----unnnunnu1uuunuuunnn11n-unnnu111nu--00110u!56-0u-un	0.3	4bfced11fdde8c61
$e_{21,0}$	xnnn0nnuunnuuuu11unnnnnnnnn1unnnn1nuunuuuuuuuuuunnnnuunuuuun1-un	1.6	f661bff7e401721d
$e_{21,1}$	xuuu1uunnuunnnn000111111111110000000u-0-------uunuuunnunnnnunuuu	5.6	899e3ff805e88de8
$e_{21,2}$	-nn1-01000nuu01--01001n!62-1n-unu10u10-n---08--01010101n111001011	13.3	7223273a921157cb
$e_{21,3}$	-nu001111111u005d-10110010100uuunnuu100u000111011011110101010-10	0.0	47f15941903b7556
$e_{22,0}$	xnnnnnnnnnnnnnnnnnnnnnnnnnnunnnnn110!-0-------100000000000000-01	0.0	7fffffefe9ea0005
$e_{22,1}$	-!55--11110000=62-1001100161-n1-11110010176677--ca--011-10-00101!==-10	0.0	af80997f9519a95e
$e_{22,2}$	x!==-01000010!62-nuuuuuu061-11-10000100da55---68--01--0011001nu100!==-01	1.1	ba14807c224ccd21
$e_{23,3}$	-10111001000010111-111u1100010000110!-1-------00000000u-0-------	0.0	5c85dd886614017f
$e_{23,0}$	-!55--11110000=62-1001100161-10-111u1115d---------111-10-00101!=62-11	0.0	2f80996f7f03a963
$e_{23,1}$	-111-100100000--1-------88-==-100n-0----------11111011=-62-------	0.0	7483db04d7b7efd7
$e_{23,2}$	x001-1101001105d-01-111!=58-==-100100!56---------00110011!-61-------	0.0	169a5e048860cea0
$e_{23,3}$	-!---1-!58--11=5-100=5-d-1162-00-05-d-10-061----------84----11-11-=61-10	0.0	553e9b0a8157cfc6
$v_{24,0}$	-5577b--10000006611--5e---------161-!==60-62----------111-053-552e---------	0.0	a404747456bb993a
$v_{24,1}$	x787-------------------78-------61----------------------------------	0.0	ea8f36c95c86056c
$v_{24,2}$	x66---!-15d--156e--==-=5d-a1--04--=60-00-5d----------------56---------	0.0	6bd8f90f09b89e66
$v_{24,3}$	x---------------6----------------5------------0---------------	0.0	73b39ba32238423e

Summary of Results: Skein-256

Extra Degrees of freedom		Note	Rounds	Time
Collision	0	[KRS2012]	4	2^{96}
		(biclique)	8	2^{120}
			9	2^{124}
			12	$2^{126.5}$
Free-start collision	8	[LiIS12]	22†	$2^{253.8}$
		(biclique)	37†	$2^{255.7}$
~~Free-tweak partial-collision~~	12	[YCJW11]	32	2^{105}
Collision	0		12	$\approx 2^{114}$
Semi-free-start collision	4		12	$\approx 2^{40}$
Free-start collision	8		20	$\approx 2^{40}$
Free-tweak near-collision	10		24	$\approx 2^{40}$
Free-tweak partial-collision	10		32	$\approx 2^{119}$

† Skein-512 attacks (less rounds expected for Skein-256)

Our results

1. New constraints
   - Multi-bit constraints
     - Better targeted to pure ARX designs
   - Boomerang constraints

2. Tools for analysis of differential characteristics
   - Publicly available
   - Code and documentation available at:
     `http://www.di.ens.fr/~leurent/arxtools.html`
     `http://www.cryptolux.org/ARXtools`

3. Problems found in several proposed attacks
   - Incompatible trails seem to appear quite naturally

4. Algorithm to build differential characteristics
   - Attack on Skein-256 in various settings

Thanks

With the support of:

- FNR Luxembourg

Fonds National de la Recherche Luxembourg

COFUNDED BY MARIE CURIE ACTIONS

- ERC project CRASH

erc European Research Council
Established by the European Commission
Supporting top researchers from anywhere in the world

Outline

Extra slides

1.5-bit 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	$x' = 1$	-	✓	-	✓	✓	-	✓	-
=	$\equiv \{0^0, 1^1\}$	✓	✓	-	-	-	-	-	-
!	$\equiv \{0^1, 1^0\}$	-	-	✓	✓	-	-	-	-
>	$\equiv \{n^n, u^u\}$	-	-	-	-	✓	✓	-	-
<	$\equiv \{n^u, u^n\}$	-	-	-	-	-	-	✓	✓

Comparison

Simple situations with a modular difference of ±1:

Diff, carry	1-bit cstr.	1.5-bit cstr.	2-bit cstr.	2.5-bit cstr.
+1, k-bit (2^{n-k})	-unnn (2^{n-k})	-unnn (2^{n-k})	-unnn (2^{n-k})	-unnn (2^{n-k})
±1, k-bit (2^{n-k+1})	-xxxx (2^{n})	->><<x (2^{n-k+1})	->><<x (2^{n-k+1})	->><<x (2^{n-k+1})
+1, any (2^{n})	????x (2^{2n-1})	????x (2^{2n-1})	UUUUx (2^{n})	UUUUx (2^{n})
±1, any (2^{n+1})	????x (2^{2n-1})	????x (2^{2n-1})	XXXXx $(2^{n} \times n)$	///Xx (2^{n+1})

Back to the talk

Base constraints

- Each constraint specifies one value for $(x^{[i]}, x'^{[i]}, x^{[i-1]}, x'^{[i-1]}, x^{[i-2]})$

$(x, x', 2x, 2x', 4x)$:

	00000	00001	00010	00011	00100	00101	00110	00111	01000	01001	01010	01011	01100	01101	01110	01111	10000	10001
$_0$03	✓	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
$_0$0C	-	✓	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
$_0$u3	-	-	✓	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
$_0$uC	-	-	-	✓	-	-	-	-	-	-	-	-	-	-	-	-	-	-
$_0$n3	-	-	-	-	✓	-	-	-	-	-	-	-	-	-	-	-	-	-
$_0$nC	-	-	-	-	-	✓	-	-	-	-	-	-	-	-	-	-	-	-
$_0$13	-	-	-	-	-	-	✓	-	-	-	-	-	-	-	-	-	-	-
$_0$1C	-	-	-	-	-	-	-	✓	-	-	-	-	-	-	-	-	-	-
$_u$03	-	-	-	-	-	-	-	-	✓	-	-	-	-	-	-	-	-	-
⋮																		

Back to the talk

Propagation example

Example System

$$u = a \vee (a \boxplus a) \qquad u' = a' \vee (a' \boxplus a')$$
$$\delta(a, a') = \mathrm{A} \qquad \delta(u, u') = \mathrm{U}$$

1. Compute transitions
2. Build a graph with initial constraints
   - Example: $\delta(a, a') = \texttt{-x--}, \delta(u, u') = \texttt{----}$
3. Identify paths
4. Paths give new constraints
   - Example: $\delta(a, a') = \texttt{1x1-}, \delta(u, u') = \texttt{111-}$

Back to the talk

Propagation example

State	Transitions
0:	$0 \xrightarrow{0^0/0^0} 0\,0 \xrightarrow{n^0/n^0} 1\,0 \xrightarrow{u^0/u^0} 3\,0 \xrightarrow{1^0/1^0} 5$
1:	$1 \xrightarrow{n^n/n^n} 1\,1 \xrightarrow{0^n/n^n} 2\,1 \xrightarrow{1^n/1^n} 5\,1 \xrightarrow{u^n/1^n} 8$
2:	$2 \xrightarrow{0^0/0^n} 0\,2 \xrightarrow{n^0/n^n} 1\,2 \xrightarrow{u^0/u^n} 3\,2 \xrightarrow{1^0/1^n} 5$
3:	$3 \xrightarrow{u^u/u^u} 3\,3 \xrightarrow{0^u/u^u} 4\,3 \xrightarrow{1^u/1^u} 5\,3 \xrightarrow{n^u/1^u} 7$
4:	$4 \xrightarrow{0^0/0^u} 0\,4 \xrightarrow{n^0/n^u} 1\,4 \xrightarrow{u^0/u^u} 3\,4 \xrightarrow{1^0/1^u} 5$
5:	$5 \xrightarrow{1^1/1^1} 5\,5 \xrightarrow{0^1/1^1} 6\,5 \xrightarrow{n^1/1^1} 7\,5 \xrightarrow{u^1/1^1} 8$
6:	$6 \xrightarrow{0^0/0^1} 0\,6 \xrightarrow{n^0/n^1} 1\,6 \xrightarrow{u^0/u^1} 3\,6 \xrightarrow{1^0/1^1} 5$
7:	$7 \xrightarrow{n^n/n^1} 1\,7 \xrightarrow{0^n/n^1} 2\,7 \xrightarrow{1^n/1^1} 5\,7 \xrightarrow{u^n/1^1} 8$
8:	$8 \xrightarrow{u^u/u^1} 3\,8 \xrightarrow{0^u/u^1} 4\,8 \xrightarrow{1^u/1^1} 5\,8 \xrightarrow{n^u/1^1} 7$

Propagation example

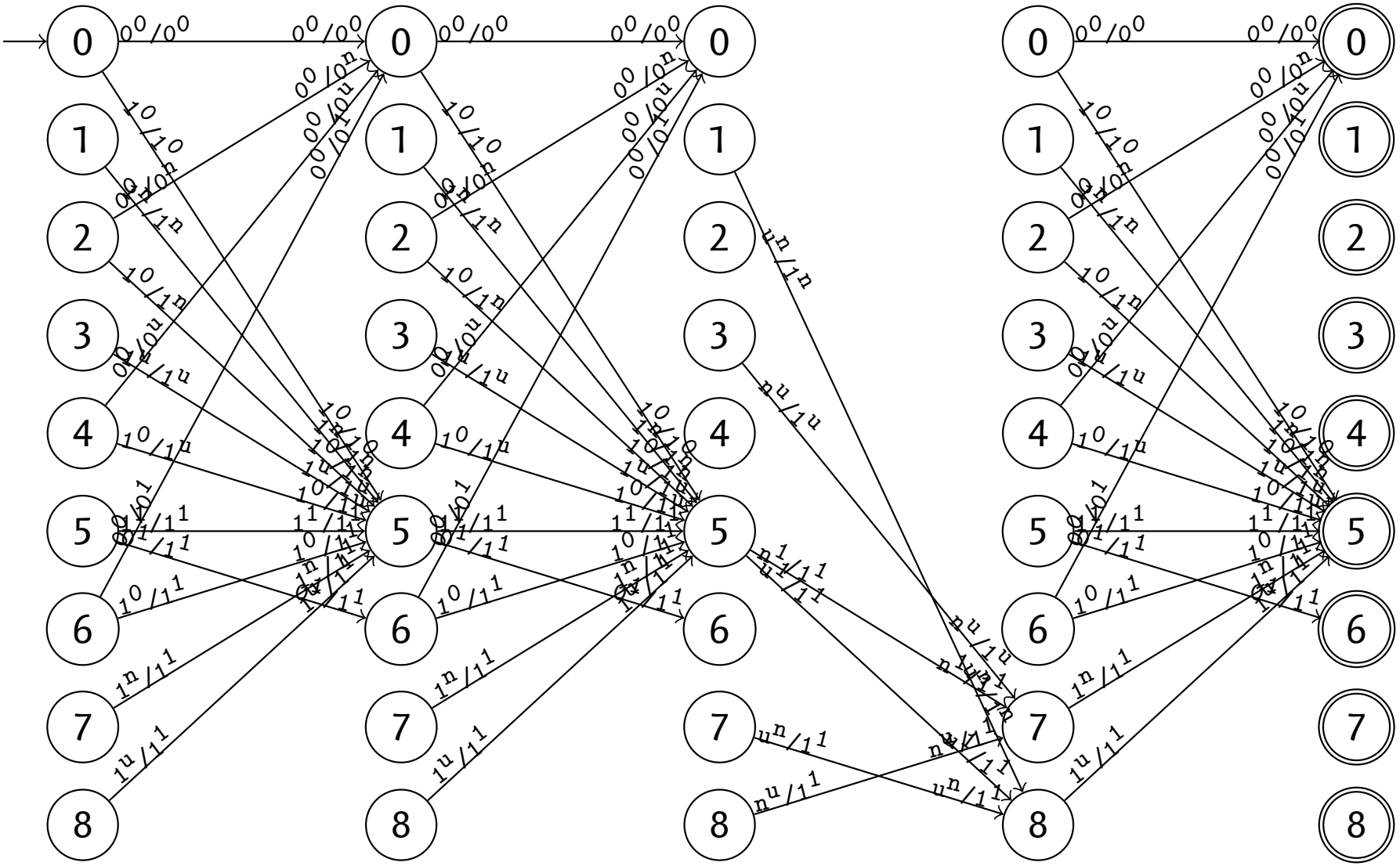
Example System

$$u = a \vee (a \boxplus a) \qquad u' = a' \vee (a' \boxplus a')$$
$$\delta(a, a') = \mathrm{A} \qquad \delta(u, u') = \mathrm{U}$$

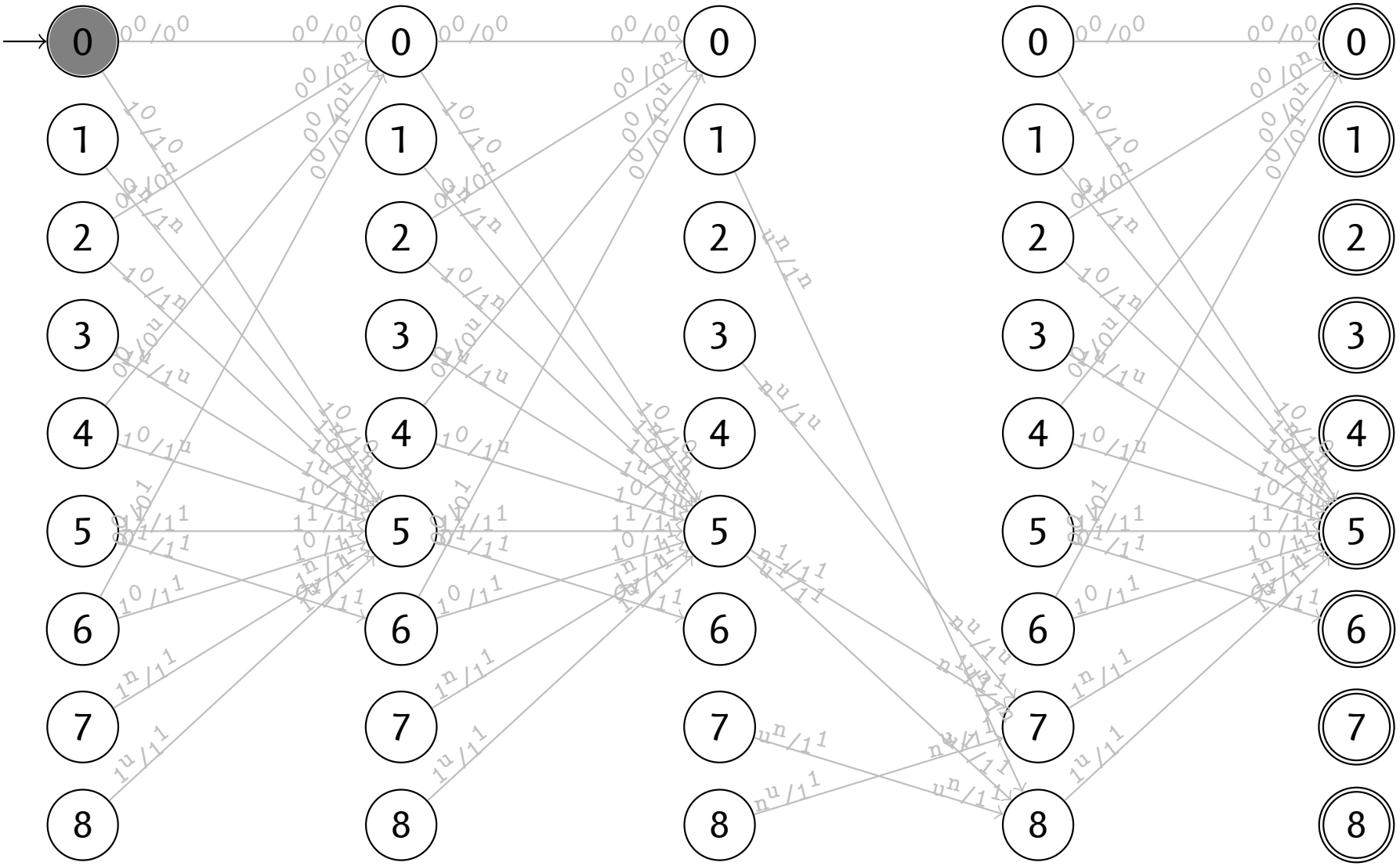
1. Compute transitions
2. Build a graph with initial constraints
   - Example: $\delta(a, a') = \texttt{-x--}, \delta(u, u') = \texttt{----}$
3. Identify paths
4. Paths give new constraints
   - Example: $\delta(a, a') = \texttt{1x1-}, \delta(u, u') = \texttt{111-}$

Back to the talk

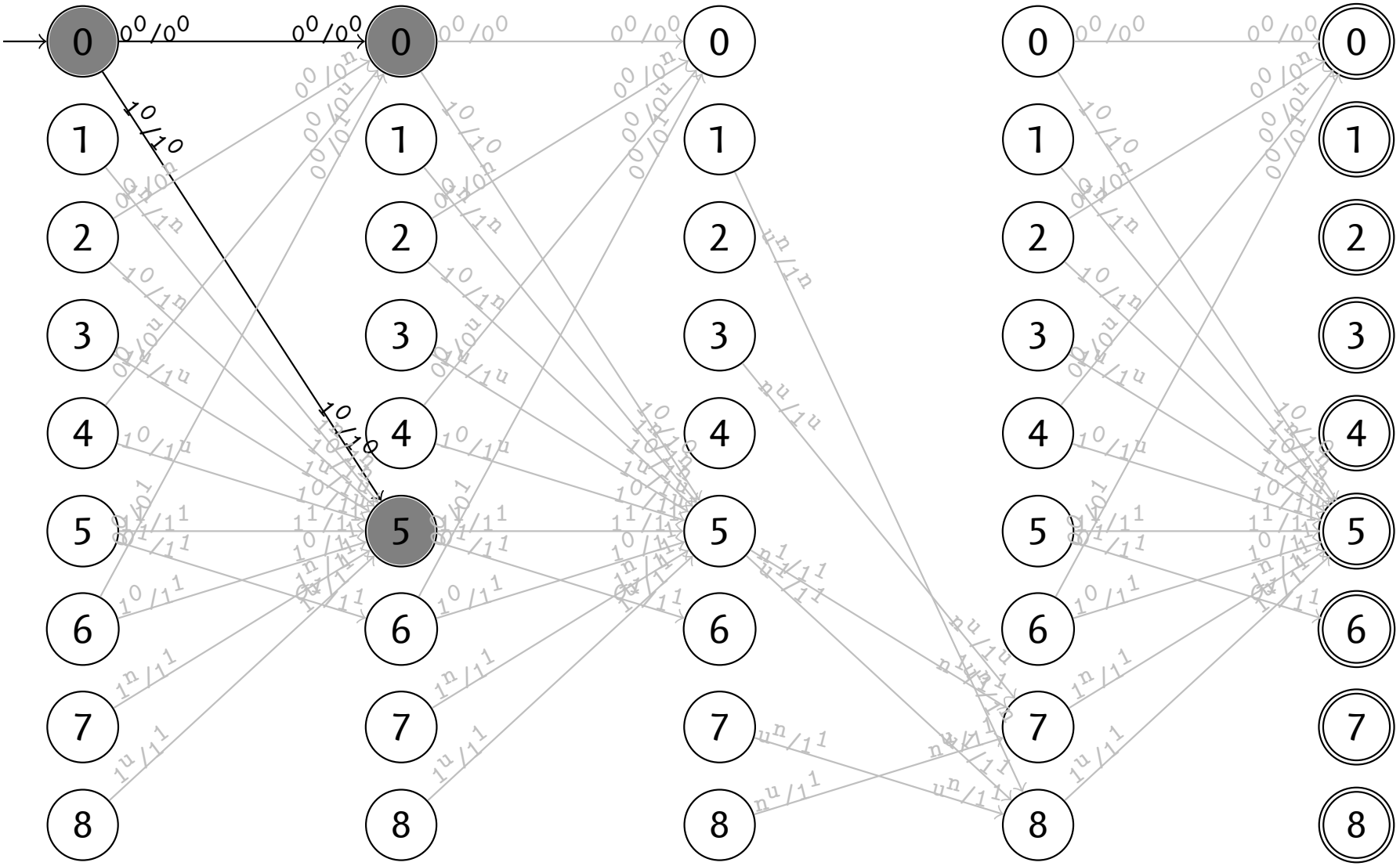
Propagation example



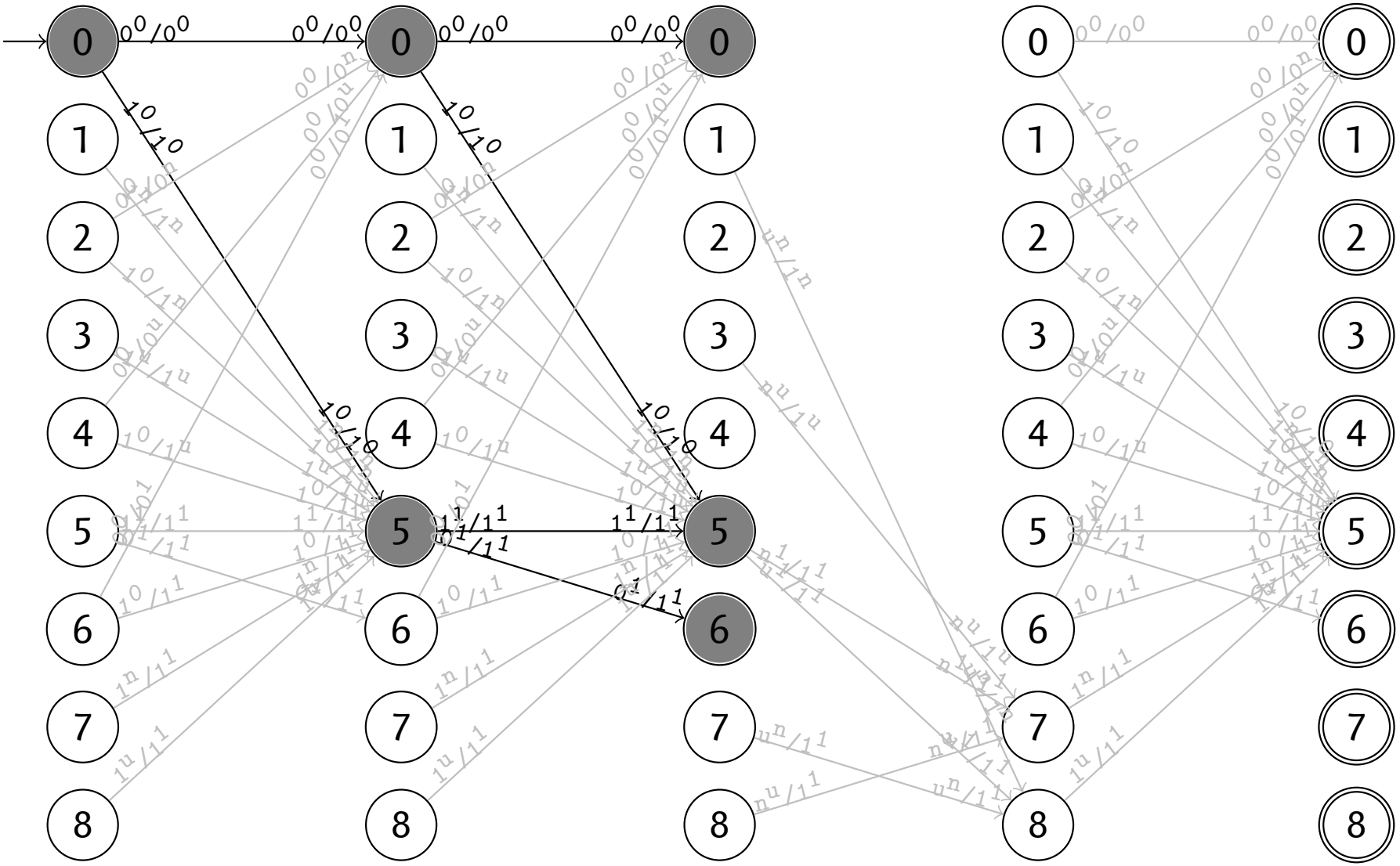
Propagation example



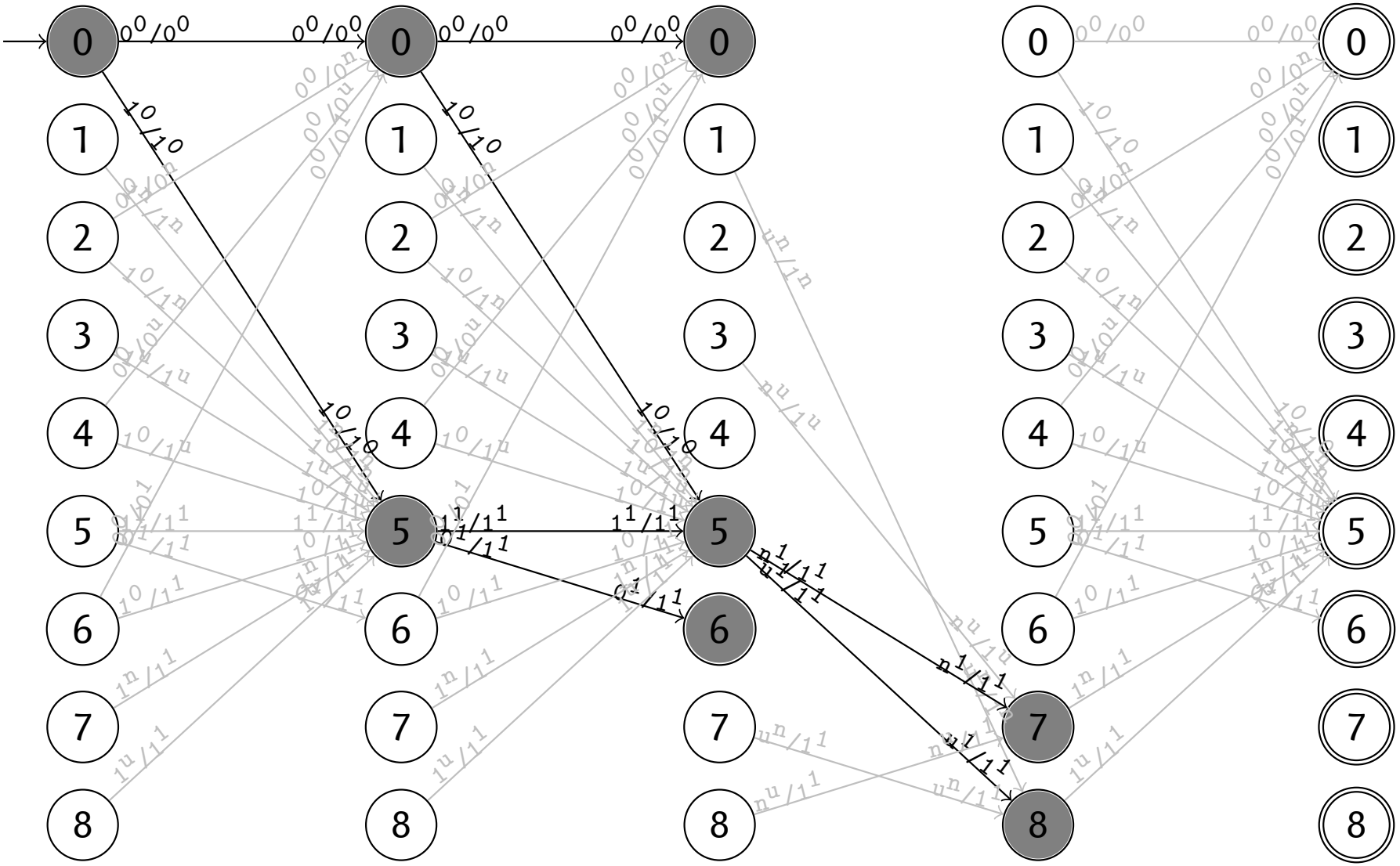
Propagation example



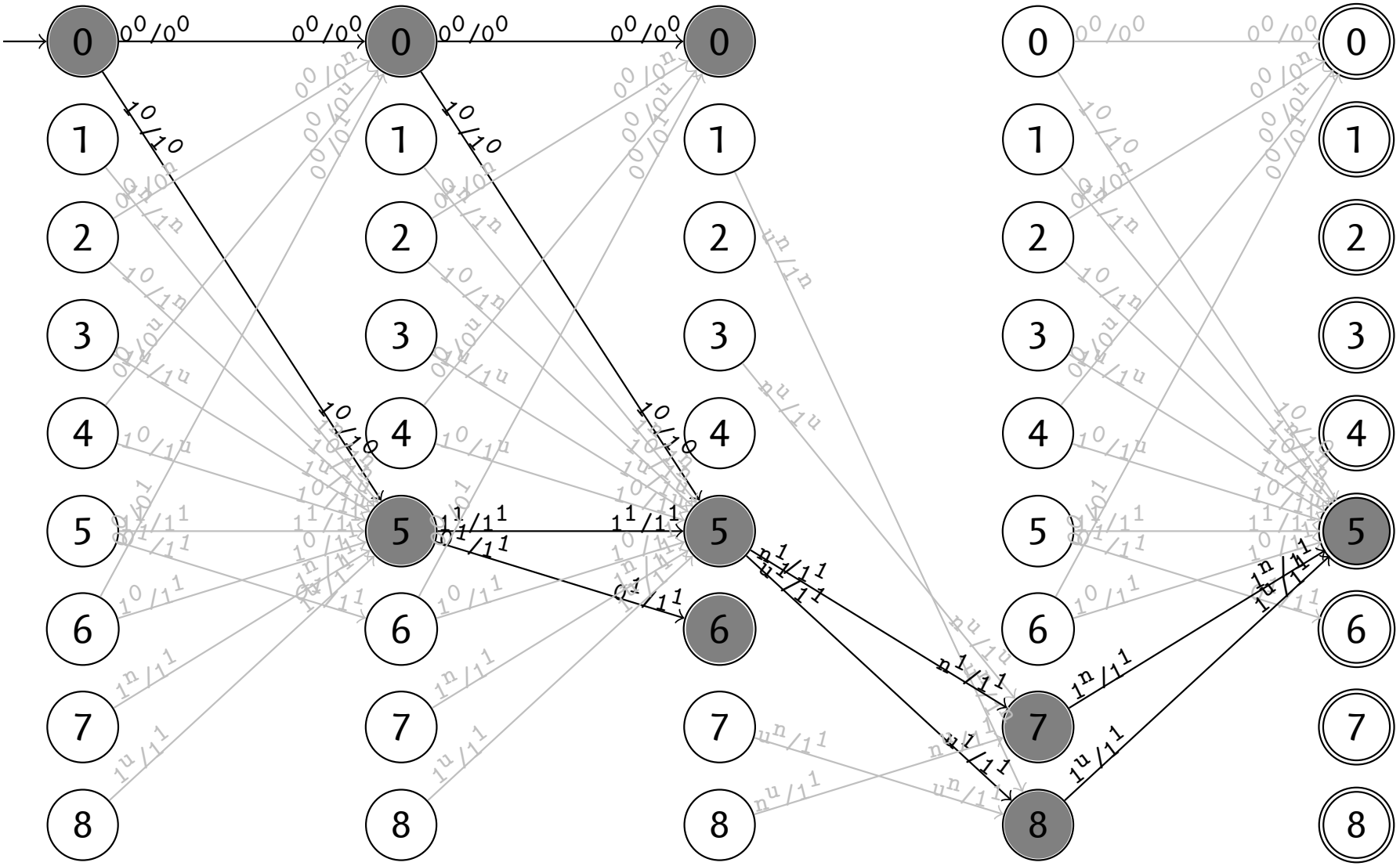
Propagation example



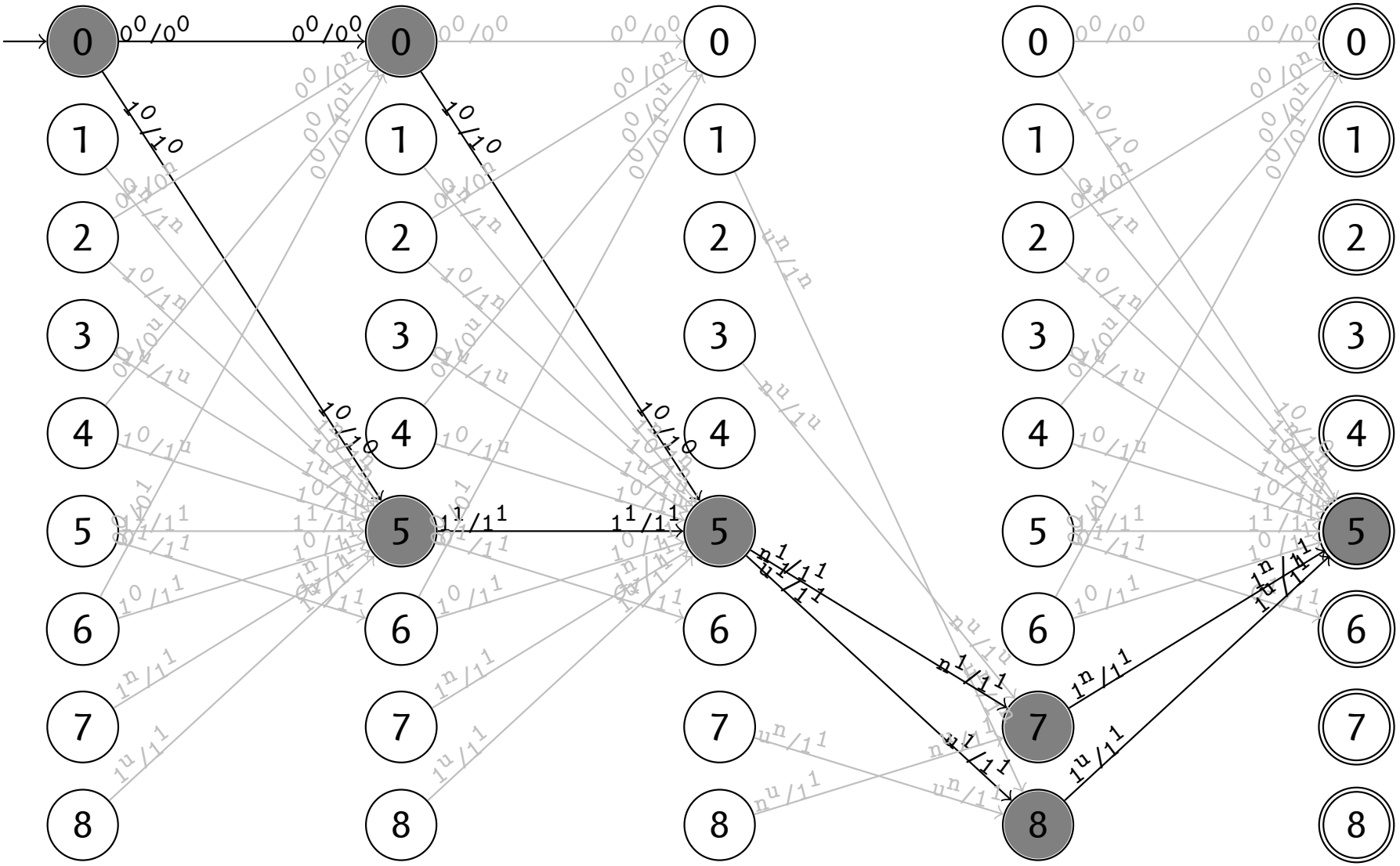
Propagation example



Propagation example



Propagation example



Propagation example

Example System

$$u = a \vee (a \boxplus a) \qquad u' = a' \vee (a' \boxplus a')$$
$$\delta(a, a') = \mathrm{A} \qquad \delta(u, u') = \mathrm{U}$$

1. Compute transitions
2. Build a graph with initial constraints
   - Example: $\delta(a, a') = \texttt{-x--}, \delta(u, u') = \texttt{----}$
3. Identify paths
4. Paths give new constraints
   - Example: $\delta(a, a') = \texttt{1x1-}, \delta(u, u') = \texttt{111-}$

Back to the talk