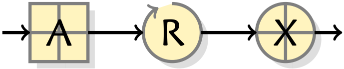
The ARX Toolkit

Gaëtan Leurent

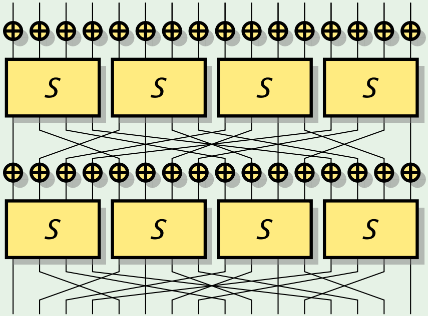
UCL Crypto Group

Asian Symmetric Key Workshop 2013

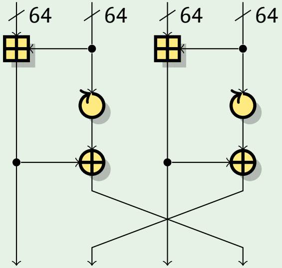


Symmetric key designs: two main categories

SmallPresent



Threefish



Symmetric key designs: two main categories

SPN with SBoxes

- S-Boxes and Linear Layers
- Important example: AES
- Few heavy rounds
- S S-Boxes
- Wire-crossing
- M MDS matrices

ARX designs

- Additions, Rotations, Xors (32/64-bit words)
- Inspired by MD/SHA
- Lots of light rounds
- ⊞ Addition
- Rotation
- ⊕ Xor

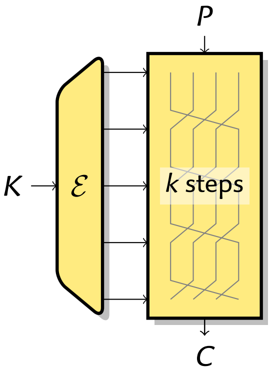
Addition, Rotation, Xor

ARX designs

Hash functions	Skein, BLAKE *(2 of the 5 SHA-3 finalists)*
Stream ciphers	Slasa20, ChaCha
Block ciphers	TEA, XTEA, HIGHT, Speck
PRF	SipHash

- Very efficient designs: Salsa20/12, BLAKE2, Simon/Speck
- Interaction between incompatible structures:
  - $\mathbb{F}_2$-linear: Rotation ⟲, Xor ⊕
  - $\mathbb{Z}_{2^n}$-linear: Addition ⊞

Differential cryptanalysis



[Biham & Shamir, Crypto 90]

- Take an input pair P, P'
  $C = E_K(P)$, $C' = E_K(P')$
- Look for Δ_P, Δ_C with large p:
  $$p = \Pr[\Delta_P \rightsquigarrow \Delta_C] = \Pr[C' = C + \Delta_C \mid P' = P + \Delta_P]$$
- Specify Δ_{X_i} at each step:
  $\Delta_P \rightsquigarrow \Delta_{X_1} \rightsquigarrow \Delta_{X_2} \rightsquigarrow \cdots \rightsquigarrow \Delta_C$
- $\Pr\left[\Delta_{X_0} \rightsquigarrow \Delta_{X_n}\right] \geq \prod_i \Pr\left[\Delta_{X_i} \rightsquigarrow \Delta_{X_{i+1}}\right]$

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]

- Tools are described in literature, but not all are public

ARXtools

1. Tool for S-systems (additions and xors)
   - Similar to [Mouha & *al.*, SAC 2010]
   - Completely automated
2. Representation of differential trails as sets of constraints, and analysis with S-systems
   - Similar to [De Cannière & Rechberger, Asiacrypt 2006]
   - Multi-bit constraints
   - Propagation of *necessary* constraints
3. Graphical tool for bit-twiddling with differential trails
4. Algorithm to build differential characteristics

ARXtools

1. Tool for S-systems (additions and xors)
   - Similar to [Mouha & *al.*, SAC 2010]
   - Completely automated

2. Representation of differential trails as sets of constraints, and analysis with S-systems
   - Similar to [De Cannière & Rechberger, Asiacrypt 2006]
   - Multi-bit constraints
   - Propagation of *necessary* constraints

3. Graphical tool for bit-twiddling with differential trails

4. Algorithm to build differential characteristics

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:
  $x \oplus \text{0x80000000} = x$

- 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

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

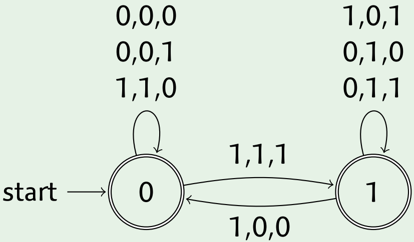
c	Δ	δ	x	c′
1	0	0	0	-
1	0	0	1	-
1	0	1	0	1
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	-
1	1	1	1	-

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

- States represent carries, transitions labeled with variables
- Automaton accepts solutions. 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*

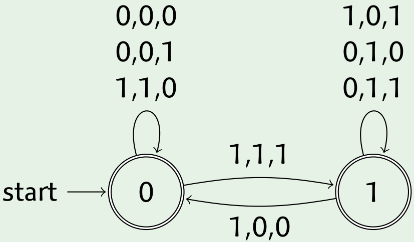


We use automata to study S-systems: [Mouha & *al.*, SAC 2010]

- States represent carries, transitions labeled with variables
- Automaton accepts solutions. 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*

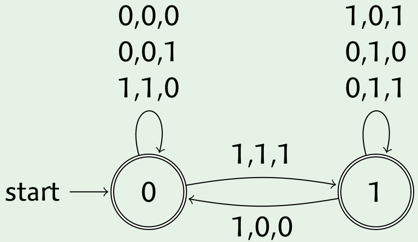


We use automata to study S-systems: [Mouha & *al.*, SAC 2010]

- States represent carries, transitions labeled with variables
- Automaton accepts solutions. Can count the number of solutions.

Decision Automata

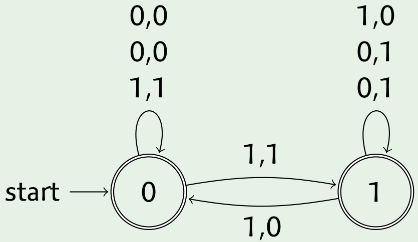
Carry transitions for $x \oplus \Delta = x \boxplus \delta$. *The edges are indexed by Δ, δ, x*



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

Decision Automata

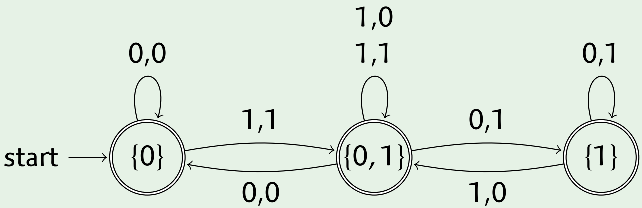
Decision automaton for $x \oplus \Delta = x \boxplus \delta$. *The edges are indexed by Δ, δ*



- 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*

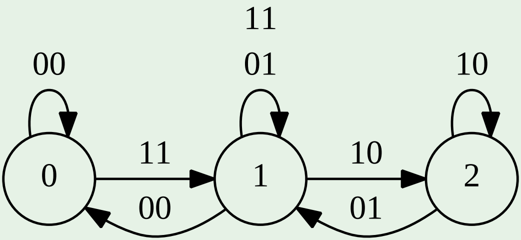


- Remove x from the transitions
- Can decide whether a given Δ, δ is compatible.
- Convert the non-deterministic automata to deterministic (optional).

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. Test compatibility, count solutions, or solve systems

ARXtools

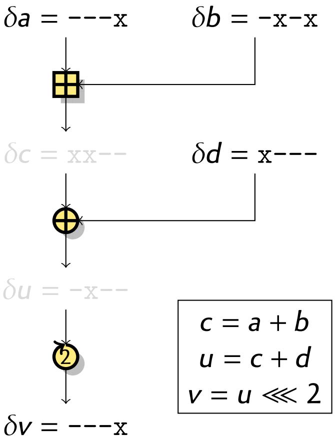
1. Tool for S-systems (additions and xors)
   - Similar to [Mouha & *al.*, SAC 2010]
   - Completely automated

2. Representation of differential trails as sets of constraints, and analysis with S-systems
   - Similar to [De Cannière & Rechberger, Asiacrypt 2006]
   - Multi-bit constraints
   - Propagation of *necessary* constraints

3. Graphical tool for bit-twiddling with differential trails

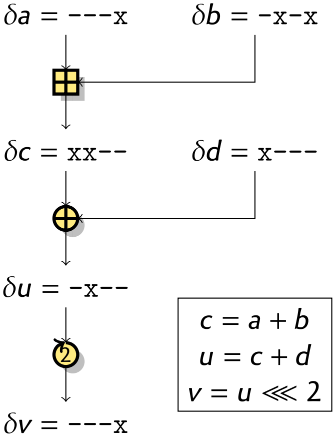
4. Algorithm to build differential characteristics

Differential Characteristic



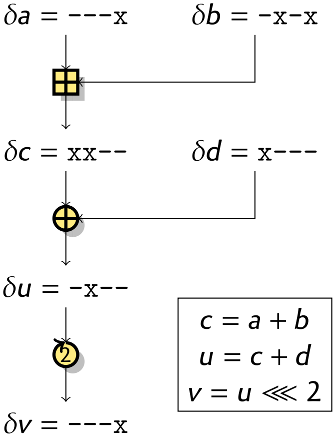
- Choose a difference operation: $\oplus$
- A differential only specifies the input and output difference
- A differential characteristic specifies the difference of each internal variable
- Compute probability for each operation

Differential Characteristic



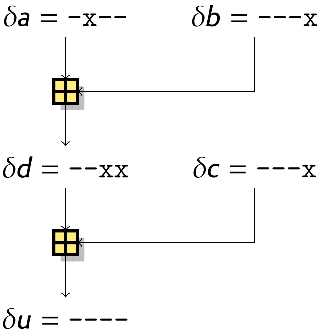
- Choose a difference operation: ⊕
- A differential only specifies the input and output difference
- A differential characteristic specifies the difference of each internal variable
- Compute probability for each operation

Differential Characteristic



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

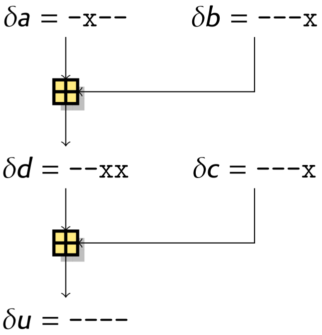
Problems with Xor-Characteristics



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

- Probability: $2^{-3} \cdot 2^{-2}$
- Obviously wrong if you consider modular differences
  - $\delta a \rightsquigarrow \pm 4$
  - $\delta b \rightsquigarrow \pm 1$
  - $\delta c \rightsquigarrow \pm 1$
- Consider signs
  [Chabaud & Joux, 1998]

Problems with Xor-Characteristics

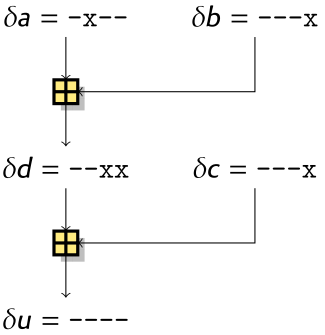


$$d = a + b$$
$$u = c + d$$

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

  [Chabaud & Joux, 1998]

Problems with Xor-Characteristics



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

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

  [Chabaud & Joux, 1998]

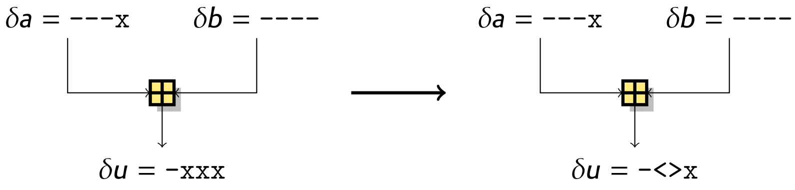
Signed difference

- 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]

	(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		-	✓	✓	✓

Multi-bit Constraints

- We study carry propagation



$$u = a + b$$ $$u = a + b$$

- 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
  - $> \equiv \{\texttt{n}^\texttt{n}, \texttt{u}^\texttt{u}\}$: $x'^{[i]} \neq x^{[i]} = x^{[i-1]}$
  - $< \equiv \{\texttt{n}^\texttt{u}, \texttt{u}^\texttt{n}\}$: $x'^{[i]} \neq x^{[i]} \neq x^{[i-1]}$

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

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})

See details

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)	0
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		

Multi-bit Constraints as S-systems

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

$$\begin{aligned} z &= x \odot y \\ f(x, x', x \boxplus x, \Delta_x) &= 0 \\ f(y, y', y \boxplus y, \Delta_y) &= 0 \\ f(z, z', z \boxplus z, \Delta_z) &= 0 \end{aligned}$$

   - Defines right pairs (x, y, z, x', y', z') for parameters $\Delta_x, \Delta_y, \Delta_z$
   - S-system

2. Build the automaton

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

Improved technique

Limitations of the initial technique

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

1. Build the system for a set of 32 base constraints
2. Take the union of the transitions

See details

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

Using the tools

```
# Macros definitions
M1: (X0==X5) && (X1==X6) && (X2==X7) && (X3==X8) && (X4==X9);
M2: X0+X0;
M0: M1(X0, X1, M2(X0), M2(X1), M2(M2(X0)), X2,X3,X4,X5,X6);

# Define operation ⊙: MD5 IF function
    M3: ((X0&X1) | ((X0^1)&X2));
# Describe the system: V0-V5 variables, P0-P19 parameters
    # Input constraints
    M0(V0, V1, P0, P1, P2, P3, P4);
    M0(V2, V3, P5, P6, P7, P8, P9);
    M0(V4, V5, P10,P11,P12,P13,P14);
    # Output constraints
    M0(M3(V0,V2,V4), M3(V1,V3,V5), P15,P16,P17,P18,P19);
```

Using the tools

```
$ build_fsm fun_if.system -s -b -v -o fun_if.fsm
Parsed expression: 20 params, 6 vars, 12 sums
512/511
Seems good
$ constraints_xx -s fun_if.fsm -w 8 -p --
   "--------" "-x------" "----x---" "--------"
System is compatible!
Propagate:
5 new constraints
New parameters:
-0--1---
-x------
----x---
--------
```

ARXtools

1. Tool for S-systems (additions and xors)
   - Similar to [Mouha & *al.*, SAC 2010]
   - Completely automated
2. Representation of differential trails as sets of constraints, and analysis with S-systems
   - Similar to [De Cannière & Rechberger, Asiacrypt 2006]
   - Multi-bit constraints
   - Propagation of *necessary* constraints
3. Graphical tool for bit-twiddling with differential trails
4. Algorithm to build differential characteristics

Verifying trails

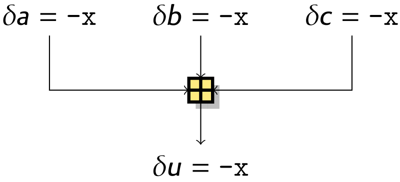
Problem

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

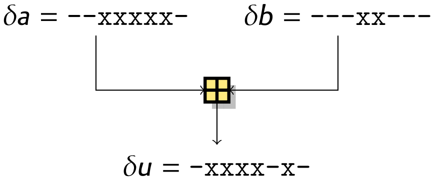
Incompatibility with additions

Some “natural” differentials do not work with additions:



$u = a + b + c$

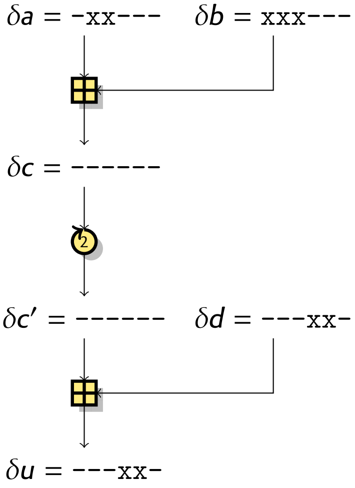
- Linearized trail



$u = a + b$

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

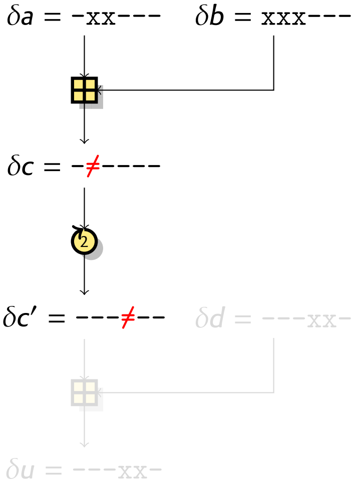
Carry incompatibility



- 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 by multi-bit constraints

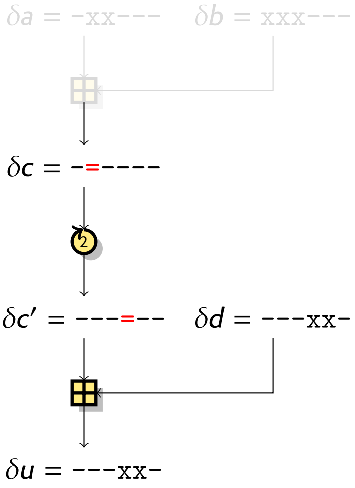
Carry incompatibility



- Each operation has a non-zero probability
- Trail seems valid with signed difference

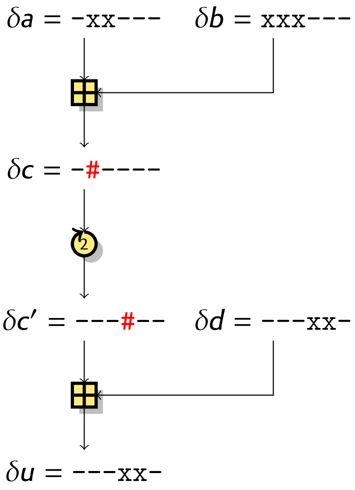
- Consider the 1st addition
  - Constraint: $c^{[4]} \neq c^{[5]}$
- Consider the 2nd addition
  - Constraint: $c'^{[2]} = c'^{[3]}$
- Incompatible!
  - Detected by multi-bit constraints

Carry incompatibility



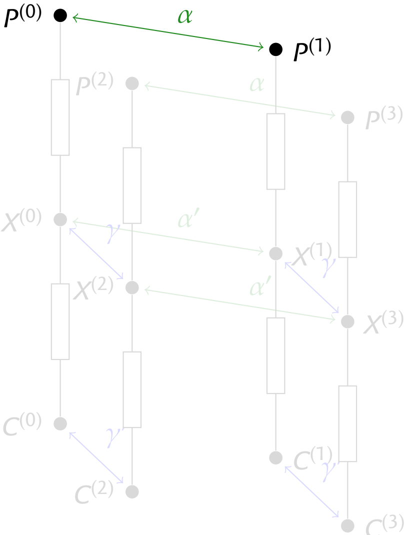
- Each operation has a non-zero probability
- Trail seems valid with signed difference
- Consider the 1st addition
  - Constraint: $c^{[4]} \neq c^{[5]}$
- Consider the 2nd addition
  - Constraint: $c'^{[2]} = c'^{[3]}$
- Incompatible!
  - Detected by multi-bit constraints

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 by multi-bit constraints

Incompatibilities in Boomerang Characteristics



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

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

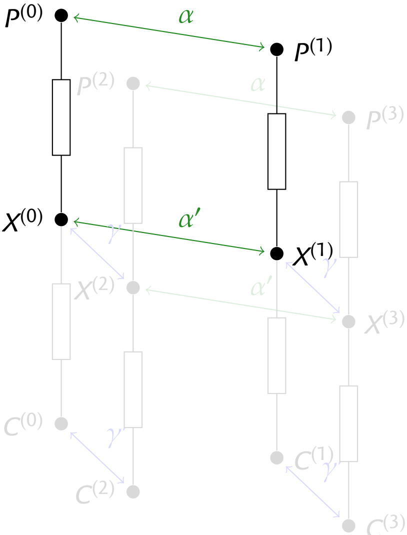
- Expect:

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

If independent: $C = 1/p_a^2 p_b^2$

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

Incompatibilities in Boomerang Characteristics



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

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

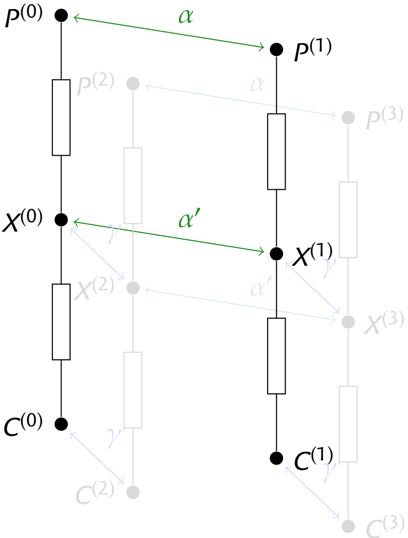
- Expect:

$$(X^{(0)}, X^{(1)}) \overset{f_a}{\longleftarrow} \alpha \quad (X^{(2)}, X^{(3)}) \overset{f_a}{\longleftarrow} \alpha$$
$$(X^{(0)}, X^{(2)}) \overset{f_b}{\longrightarrow} \gamma' \quad (X^{(1)}, X^{(3)}) \overset{f_b}{\longrightarrow} \gamma'$$

If independent: $C = 1/p_a^2 p_b^2$

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

Incompatibilities in Boomerang Characteristics



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

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

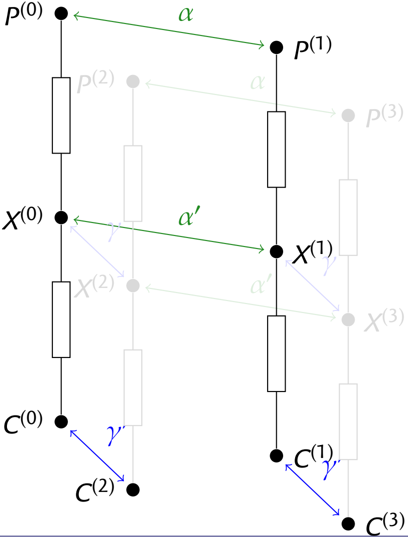
- Expect:

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

If independent: $C = 1/p_a^2 p_b^2$

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

Incompatibilities in Boomerang Characteristics



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

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

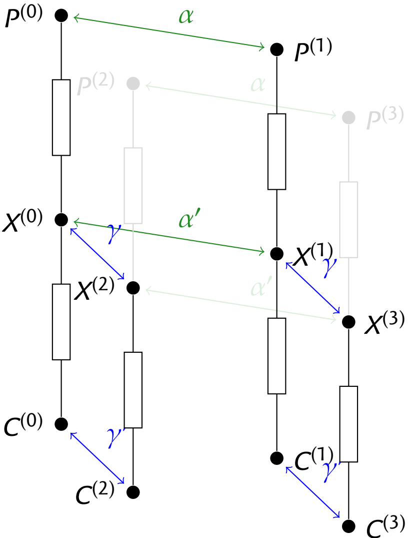
- Expect:

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

If independent: $C = 1/p_a^2 p_b^2$

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

Incompatibilities in Boomerang Characteristics



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

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

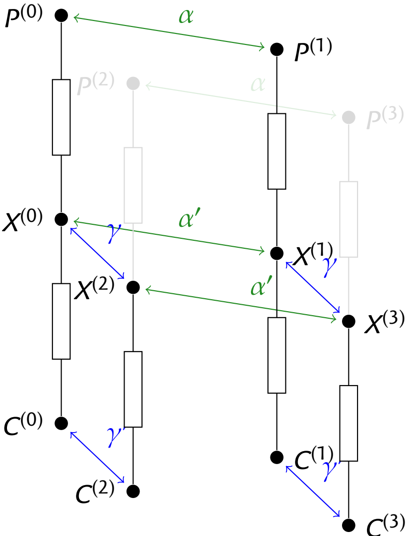
- Expect:

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

If independent: $C = 1/p_a^2 p_b^2$

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

Incompatibilities in Boomerang Characteristics



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

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

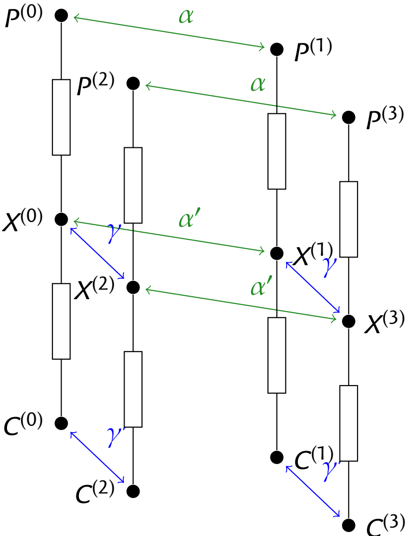
- Expect:

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

If independent: $C = 1/p_a^2 p_b^2$

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

Incompatibilities in Boomerang Characteristics



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

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

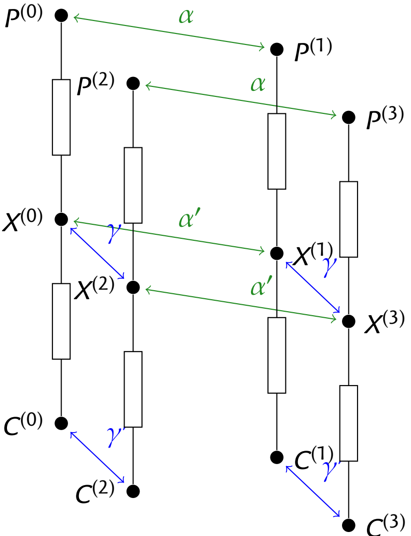
- Expect:

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

If independent: $C = 1/p_a^2 p_b^2$

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

Incompatibilities in Boomerang Characteristics



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

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

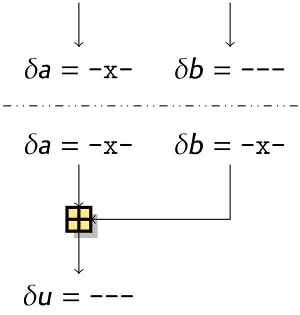
- Expect:

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

If independent: $C = 1/p_a^2 p_b^2$

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

Boomerang incompatibility



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

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

$$u = a + b$$

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

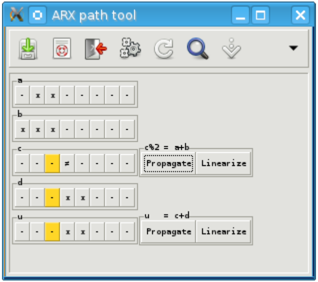
Graphical tool

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

arx.path

```
@conf wordsize = 6;

@vbox;
@state a         : -xx---
@state b         : xxx---
@state c%2 = a+b : ------
@state d         : ---xx-
@state u   = c+d : ---xx-
@end;
```



Verifying characteristics

Several proposed 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
  - 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 [Yu, Chen & Wang, FSE 2013]
  - Complex rebound-like handcrafted characteristic
  - ePrint version was not satisfiable
  - Fixed in final paper

ARXtools

1. Tool for S-systems (additions and xors)
   - Similar to [Mouha & *al.*, SAC 2010]
   - Completely automated

2. Representation of differential trails as sets of constraints, and analysis with S-systems
   - Similar to [De Cannière & Rechberger, Asiacrypt 2006]
   - Multi-bit constraints
   - Propagation of *necessary* constraints

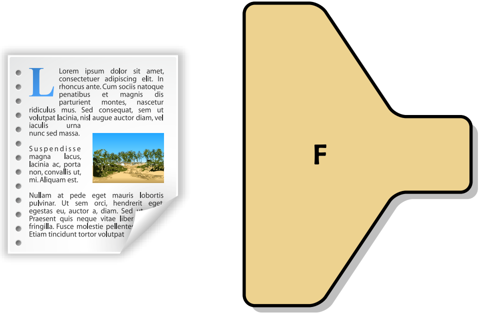
3. Graphical tool for bit-twiddling with differential trails

4. Algorithm to build differential characteristics

Hash Functions

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

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



0x1d66ca77ab361c6f

Security goals

Preimage attack

Given F and $\overline{H}$, find M s.t. $F(M) = \overline{H}$.
Ideal security: 2^n.

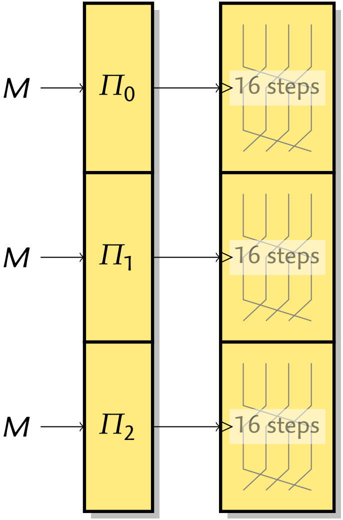
Second-preimage attack

Given F and M_1, find $M_2 \neq M_1$ s.t. $F(M_1) = F(M_2)$.
Ideal security: 2^n.

Collision attack

Given F, find $M_1 \neq M_2$ s.t. $F(M_1) = F(M_2)$.
Ideal security: $2^{n/2}$.

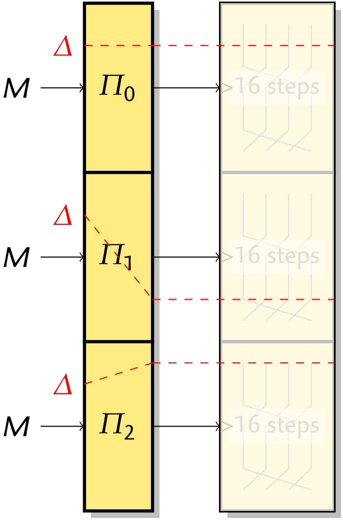
Differential collision attack



[Chabaud & Joux, Crypto 1998]
[Wang *& al*, Crypto & EC 2005]

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

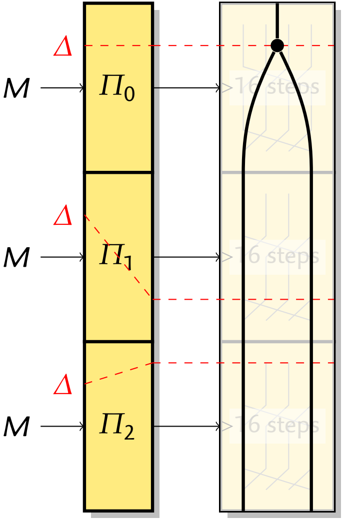
Differential collision attack



[Chabaud & Joux, Crypto 1998]
[Wang & al, Crypto & EC 2005]

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

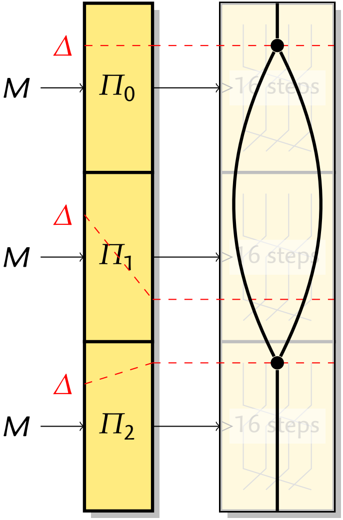
Differential collision attack



[Chabaud & Joux, Crypto 1998]
[Wang *& al*, Crypto & EC 2005]

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

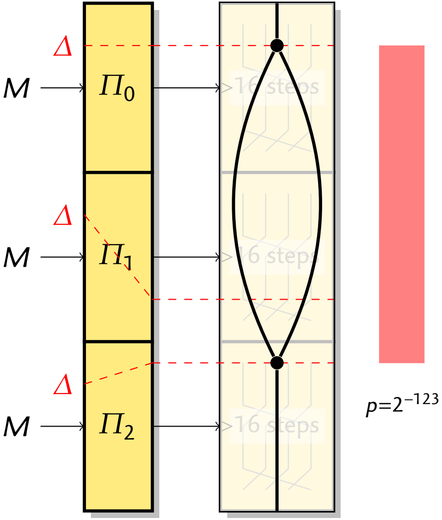
Differential collision attack



[Chabaud & Joux, CRYPTO 1998]
[Wang *& al*, CRYPTO & EC 2005]

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

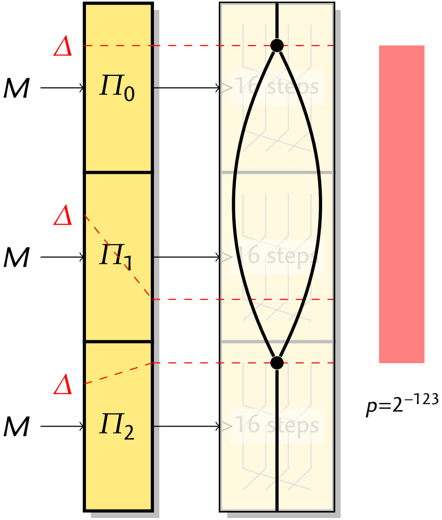
Differential collision attack



[Chabaud & Joux, Crypto 1998]
[Wang & *al*, Crypto & EC 2005]

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

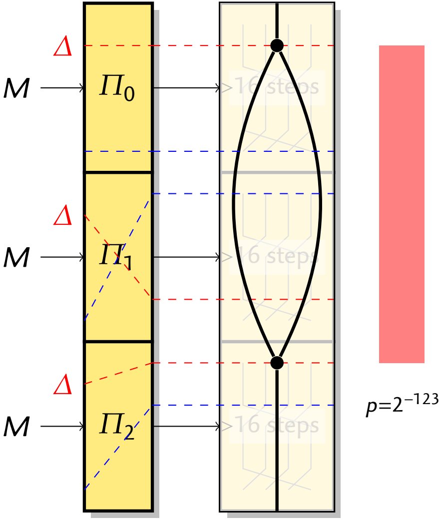
Differential collision attack



[Chabaud & Joux, Crypto 1998]
[Wang & *al*, Crypto & EC 2005]

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

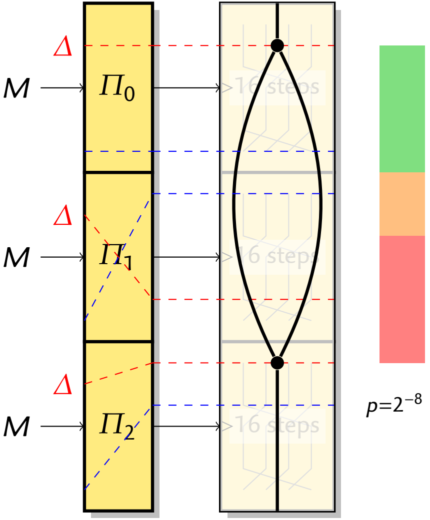
Differential collision attack



[Chabaud & Joux, CRYPTO 1998]
[Wang & *al*, CRYPTO & EC 2005]

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

Differential collision attack



[Chabaud & Joux, CRYPTO 1998]
[Wang & *al*, CRYPTO & EC 2005]

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

Background: Cryptanalysis of MD/SHA

- In 2005, devastating attacks on MD5/SHA-1 [Wang & *al.*]
  - Collisions attacks based on differential cryptanalysis
- Differential trails built by hand
  - Very technical attacks
  - Bit-twiddling
  - Problems with several attacks
- Later, automatic search
  - MD4, MD5, SHA-1, SHA-2, ... [SO06], [S+09], [DCR06], [MNS11]
  - Much easier to analyze similar design
  - Leads to powerful attacks using special characteristics
    - HMAC-MD4 key recovery [FLN]
    - Rogue MD5 certificate [Stevens & *al.*]
    - Attack against combiners [Mendel, Rechberger & Schläffer]

Differential attacks against ARX

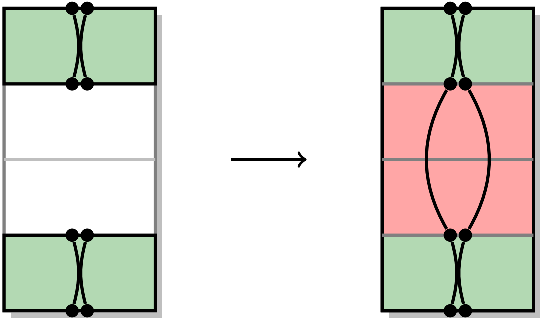
- Differential cryptanalysis of ARX designs requires bit-twiddling
  - As opposed to SBox based designs
- Building/verifying differential trails for ARX designs is hard
  - Problems with several attacks
  - Very few complex trails known (build by hand)
  - Hard to evaluate a design

Our contribution

- Build ARX trails automatically
- Pure ARX designs seem harder than MD/SHA
  - No absorbing Boolean functions
  - The only freedom is in the carry extensions

Building differential characteristics

- We target hash-function attacks
- We aim to connect two high-probability trails
- We will use degrees of freedom on the low probability section



Building differential characteristics

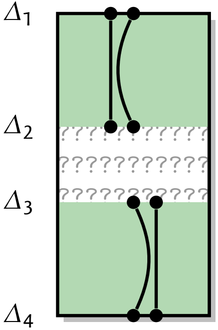
- We target hash-function attacks
- We aim to connect two high-probability trails
- We will use degrees of freedom on the low probability section

Using the algorithm

1. Set input/output difference, and key difference
   - Select simple high probability trails by hand
2. Algorithm finds intermediate difference
   - Complex trail in the middle
3. Find a pair of input values
   - Easy using degree of freedom

Algorithm

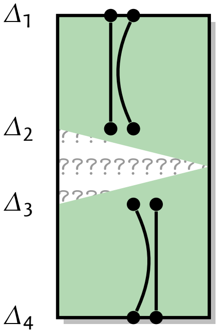
- Guess active bits in the middle and propagate
- Propagation will add necessary constraints (forced guess)



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

Algorithm

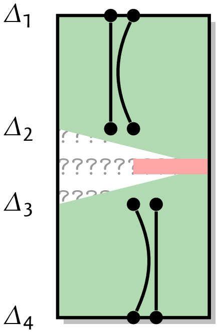
- Guess active bits in the middle and propagate
- Propagation will add necessary constraints (forced guess)



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

Algorithm

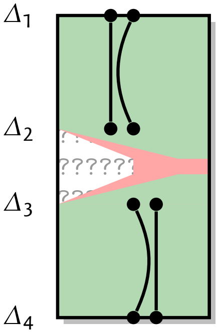
- Guess active bits in the middle and propagate
- Propagation will add necessary constraints (forced guess)



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

Algorithm

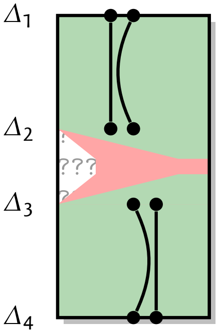
- Guess active bits in the middle and propagate
- Propagation will add necessary constraints (forced guess)



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

Algorithm

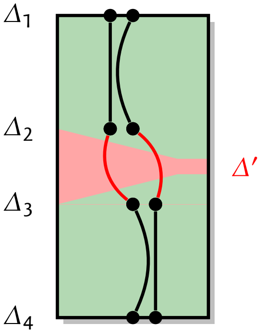
- Guess active bits in the middle and propagate
- Propagation will add necessary constraints (forced guess)



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
  - Simulated annealing

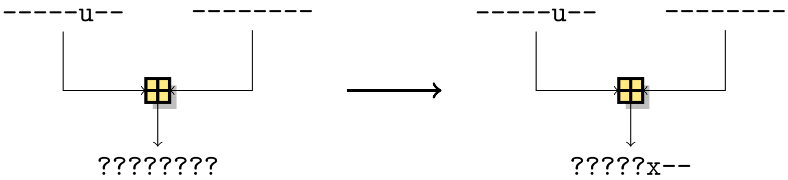
Main step: Propagation

- We want to propagate information:

- Input difference given
- Goal: infer output difference

Main step: Propagation

- We want to propagate information:



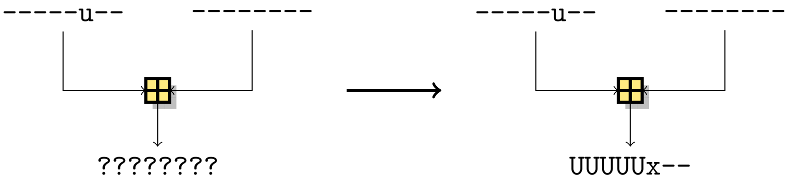
- Input difference given
- Goal: infer output difference

With single-bit constraints: [DCR06]

- We don't know if there is a carry
- Output bits can be active or inactive

Main step: Propagation

- We want to propagate information:



- Input difference given
- Goal: infer output difference

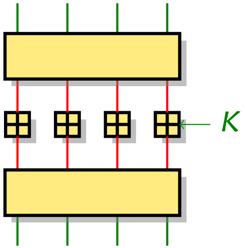
With multi-bit constraints: [L13]

- Carry bit can be active only if previous bit is active:
  - `x` if previous bit is `n`
  - `-` if previous bit is `-` or `u`

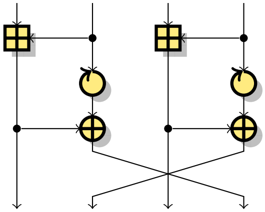
Degrees of freedom

- Without degree of freedom, connecting trails does not make sense
  - For a fixed permutation, one pair on average with a given input/output difference

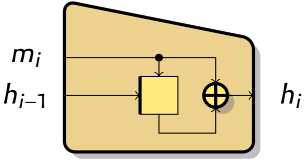
- Use key addition as the meeting point:



Skein



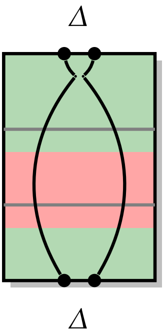
Threefish-256 round



MMO mode

- ARX design
  - 64-bit words
  - $\mathtt{MIX}_r(a,b) := ((a \boxplus b), (b \lll r) \oplus c)$
  - 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

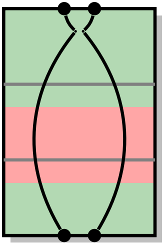


- Trails with no key difference
- Select a small difference Δ in the state
  - Build a trail $\Delta \to \Delta$
  - Collisions with the feed-forward
- Algorithm finds 12-round characteristics
- Practical attack

Limitations

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

Collision Attack

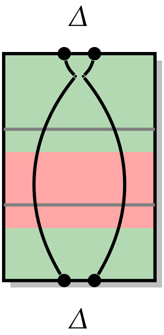


- Trails with no key difference
- Select a small difference Δ in the state
  - Build a trail $\Delta \to \Delta$
  - Collisions with the feed-forward
- Algorithm finds 12-round characteristics
- Practical attack

Limitations

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

Collision Attack

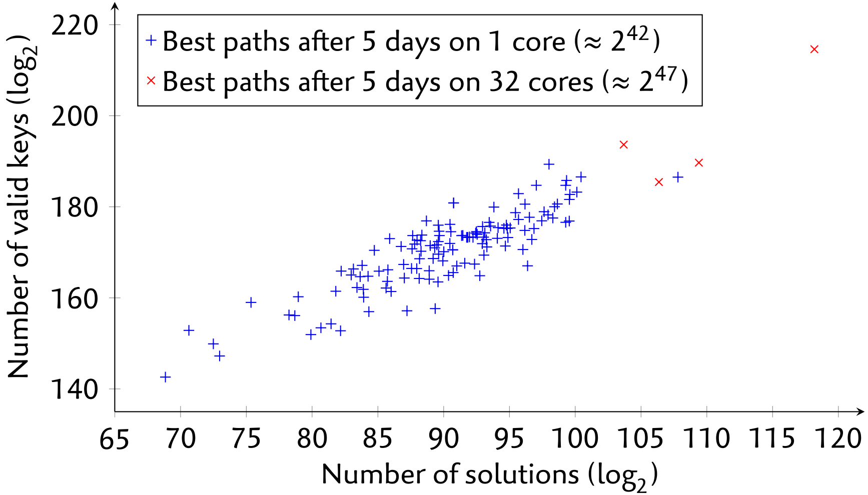


- Trails with no key difference
- Select a small difference Δ in the state
  - Build a trail $\Delta \rightarrow \Delta$
  - Collisions with the feed-forward
- Algorithm finds 12-round characteristics
- Practical attack

Best path

Valid keys	$2^{214.6}$	*Probability*	$2^{-124.9}$
Valid states	$2^{28.4}$	*Solutions*	$2^{118.1}$

Experiments



Differential path

	Constraints	Prob.	Example
k_0	------0[70]------[71]-[72]--[73]---0[74]-------[75]---![76]----[73]---[77]-[78]-----1[78][70]--[79]----		1026e28955c0ee6b
k_1	--		730713324ca92af6
k_2	------------------[71]--!=----0[70]------[7a]-[77]----[7b]---[75]--[7c]--------------		2c54640ad6894e20
k_3	--[7d]----[7e]----[7f]-----0-00[80]-=--00---[7e]--011-1-1[7e]--01--0-10-1--[74]--=[81]-0		3f264123afdb3740
k_4	-------[82]------------------------[7b]---------------------[83]---------		6b82cf48c9c7a7df
$e_{4,0}$	-----x-------------x--	2.0	baf8706e8d9d4741
$e_{4,1}$	-------------------x--	0.0	c68d20d1606e4b39
$e_{4,2}$	--	0.0	be982098c566415f
$e_{4,3}$	--	0.0	06b9774647dcb276
$e_{5,0}$	-----x--	1.0	8185913fee0b927a
$e_{5,1}$	--	0.0	b217d003bf34f56c
$e_{5,2}$	--	0.0	c55197df0d42f3d5
$e_{5,3}$	--	0.0	c9b1c9247cc5e3d9
$e_{6,0}$	-----x--	1.0	339d6143ad4087e6
$e_{6,1}$	--	0.0	3c900291c2f15c69
$e_{6,2}$	--	1.0	8f0361038a08d7ae
$e_{6,3}$	-----x--	0.0	6556403ead7b74a9
$e_{7,0}$	-----x--	1.0	702d63d57031e44f
$e_{7,1}$	-----x-----------------------n----1-----------------------------	0.0	8f2d082761c472fa
$e_{7,2}$	-----x--	1.0	f459a14237844c57
$e_{7,3}$	-----x--	0.0	38cc1b7b44afac4e
$v_{8,0}$	------------------------1----n----------------------------------	2.0	ff5a6bfcd1f65749
$v_{8,1}$	--------------------------------u-------------------------------	1.0	b8d0357a65b097cd
$v_{8,2}$	--	0.0	2d25bcbd7c33f8a5
$v_{8,3}$	x-----------------------u----u----------------------------------	2.0	1afb6f10e9780818

Differential path

	Constraints	Prob.	Example
k_0	------0[7/0]-------[7/1]-[7/2]--[7/3]---0[7/4]--------[7/5]---![7/6]----[7/3]---[7/7]-[7/8]-----1[7/8][7/0]--[7/9]----		1026e28955c0ee6b
k_1	--		730713324ca92af6
k_2	------------------[7/1]--!=----0[7/0]------[7/a]-[7/7]----[7/b]---[7/5]--[7/c]--------------		2c54640ad6894e20
k_3	--[7/d]----[7/e]----[7/f]-----0-00[8/0]-=--00---[7/e]--011-1-1[7/e]--01--0-10-1--[7/4]--=[8/1]-0		3f264123afdb3740
k_4	-------[8/2]-----------------------[7/b]----------------------[8/3]--------		6b82cf48c9c7a7df
$e_{8,0}$	-------------------------n--[7/8]-----------------------------	0.6	72617f2f1e9f823f
$e_{8,1}$	[8/4]--[8/5]---[8/6][7/2]--------------------[7/b]--u----1----[8/7]------=----0--1-011!-	0.3	e52499853c39e5ed
$e_{8,2}$	-[7/7]-----------------------------[7/4]------------------------------	0.0	6c4bfde12c0f2fe5
$e_{8,3}$	x=-=![8/1]-----1----=-------u-[7/1]--u-----![7/2]---=!---1[8/8]---!------1----0-	0.0	867e3e59b33faff8
$e_{9,0}$	---1-1-1-[7/a]-[8/9]0-[7/f]--[7/7]----0-101!-n0[8/a]u-011!-!-1[7/f]---0[8/b]---0[8/c][7/7]-[7/8]-[7/9]------	0.6	578618b45ad9682c
$e_{9,1}$	x-0101-010110101011---1-n10!-u1n=-0100-1-01---1!-!-==[7/5]-n-=!-n--[7/5]	1.0	94b563cbd3b2a36e
$e_{9,2}$	<==[7/c]---!=---1[8/d]---![8/3]-[7/b][7/5]-0u-[8/e]--u--11!=[8/1]------0![7/3][8/8]-![7/8]-----11[8/f][8/2]-11[8/2]--	2.2	f2ca3c3adf4edfdd
$e_{9,3}$	010[9/0]---n1---11-00110[8/2][8/2]-1!-[9/1]--n--n0000001=-0100=![8/f]-----01[9/2]0-111--	0.0	5dfe6b7f8113211f
$e_{10,0}$	n=-!-!=[8/1]-0-=!-1--[7/a]--1nuu1uuuuuuuu0101110100110000-0101n-0-1n--0	2.4	ec3b7c802e8c0b9a
$e_{10,1}$	n0-nunnu[7/5][8/1]-1=-110n-11nnnn01010nnu10100100111001111-1010[8/2]-0-u00-1	0.6	b67f5fab5273f523
$e_{10,2}$	u10nuuuu1100!=[7/d]--01-unnnn011!-nuu1!=[7/3]-----10[8/9]0-0000-0000[8/2]=-11[7/1]--	1.7	50c8a7ba606200fc
$e_{10,3}$	u10001001n1000u--1u-1uun-ununnunu1110---0-n11=-u1n1-111n-111u[8/2]--	1.4	44e0d9ad767eff76
$e_{11,0}$	-0nu-unu-01110[8/b]--n!-n100001010nn1000000nuuuuuuuuuuuuuuuu-unnn[8/2]-1		a2badc2b810000bd
$e_{11,1}$	nn-nnnn011n101u-0-u-nnn011unn-uu0u-010n1--110000u0n1nuuuu-000[8/2]-u		fef41ed80b703844
$e_{11,2}$	1001010nnu1010u1-0u00uu!-nn0u-nn1101011!-1n00001u0uuuuuu0=-1u0-0		95a98167d6e10072
$e_{11,3}$	0unu1n0u0nnuuu11-u10u00n-10nun0n0010nnuuunuunuunnnuunnnn0nnun0-1		2c6321552c49cf69
$v_{12,0}$	-0-u-uun-0nunnn-1-nnnun10000u-111u-01nu0--110000u0n1nuu[8/a]u-00u-un	4.7	a1aefb038c703901
$v_{12,1}$	11n011100nuuu1010n1u11u1n10n0--uu010nn11-1u0100n-nn0n11u-u-0nn-u	0.0	ee456dd42f49ee8e
$v_{12,2}$	11u0uunuu0001nuun0n0uu1unun1n-0nuuuuu01n-u1unu1u11u01111[9/3]n-n10-1	2.4	c20ca2bd032acfdb
$v_{12,3}$	-u-u-u1u--unnnn-n-0u001nu-00u--nu1-nu010--u001u-u-1-011[9/4]n-un1-01	0.0	aadec347728427d9

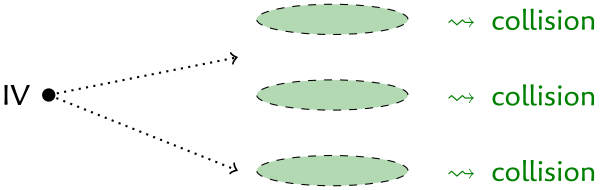
Differential path

	Constraints	Prob.	Example
k_0	`------0[70]------[71]-[72]--[73]---0[74]-------[75]---![76]----[73]---[77]-[78]-----1[78][70]--[79]----`		1026e28955c0ee6b
k_1	`----------------------------------------------------------------`		730713324ca92af6
k_2	`------------------[71]--!=----0[70]------[7a]-[77]----[7b]---[75]--[7c]--------------`		2c54640ad6894e20
k_3	`--[7d]----[7e]----[7f]-----0-00[80]-=--00---[7e]--011-1-1[7e]--01--0-10-1--[74]--=[81]-0`		3f264123afdb3740
k_4	`-------[82]------------------------[7b]---------------------[83]---------`		6b82cf48c9c7a7df
$e_{12,0}$	`NNVMVM[10][50]UUMNNNVU[11]x[12]x-nn----un---[12]x-0-[13]x-=-[82]---[14]NV[15]x-[16]UMVM[17]NVMN[10]x`	3.9	ce035f0e62f98721
$e_{12,1}$	`--n----1----n-----n-nnn-n--nunn-n--1nn-10un--nu-u0-uu-u-nn--nn-u`	7.0	2d6baef7df2525ce
$e_{12,2}$	`NVMV[11]x-[51]VUUUUUMN[52]UMVUMNVMNVMVUUU[18]UUMVUUMVMNVUM[10]x[19]NVUUU[1a][1b]MV[11]x----`	1.3	2d8f7205ccf277ba
$e_{12,3}$	`n-n-n-nn-0----u-n--u---1n---u---n---n0----u---u-u--1--nuu--u----`	13.9	bb05a5d0c8451646
$e_{13,0}$	`----n-nn0-n--nnnu-u--nnu-uuuu--uu-uu-un-0-u1--nu-0--n----n----n-`	22.1	fb6f0e06421eacef
$e_{13,1}$	`---u---n0---u---n---nn----0---n-1--u--n10-11---[88]0-1-u-01-u----u-`	1.4	4d45df9383713505
$e_{13,2}$	`x--0---un---0---0---0n----u---1-n0-n--0nu--n----n-u-1-nu-u----0-`	19.9	e89517d695378e00
$e_{13,3}$	`----0-00n-0--0101-1--u01-nu01--1000u-01-u-1u--01-u--0----u----0-`	1.0	10d2f9cf0b6d27b5
$e_{14,0}$	`-----u-0--n----u[77]--u-!--n-0-n---1n-[71]--!-[82]-u----n--1-[78]---------[7b]-`	8.6	48b4ed99c58fe1f4
$e_{14,1}$	`-----[72]---------n-----[95]--[72]---u---0u[7b]---!---n----u--1-----1-------`	0.4	9349b4563eb26ffa
$e_{14,2}$	`x------1[70]------0[72]----[76]----!-0---n0[82][78]--=---1----0--n-[77]---n-----[75]-`	6.3	f96811a5a0a4b5b5
$e_{15,3}$	`x----0----1----0---1----1---0---u0--------1----1--u-------------`	0.0	18e039c43cb7d6e7
$e_{15,0}$	`-----u----n--------u----n-------=--------------[72]--------1-------`	4.6	dbfea1f0044251ee
$e_{15,1}$	`----------u-------------u-------------------------------1-------`	1.0	a59eac713d6548a0
$e_{15,2}$	`----------0---------------------1[74]----------------------n-------`	2.0	12484b69dd5c8c9c
$e_{15,3}$	`-----0----1--------1----1-------------------------------u-------`	0.0	f0e1f8c7f90bf534
$v_{16,0}$	`-----u-------------u--------------------------------------------`	2.0	819d4e6141a79a8e
$v_{16,1}$	`-------------------x--------------------------------------------`	0.0	2254e2afca57992f
$v_{16,2}$	`----------------------------------------------------------------`	0.0	032a4431d66881d0
$v_{16,3}$	`-----0-------------0--------------------------------------------`	0.0	3248c046ed0e8e9a

Full Collision Attack

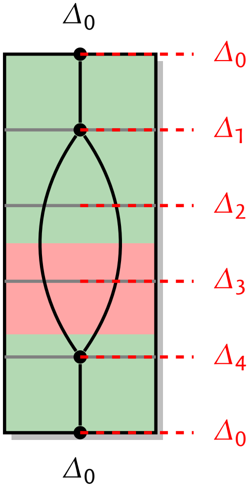
- We build a collision characteristic valid for 2^{106} keys for a cost of $\approx 2^{50}$

1. Build many characteristics (2^{50})
2. Use random message blocks to reach a valid CV for one path.



- Collision attack for 12-round Skein-256 with complexity $\approx 2^{100}$

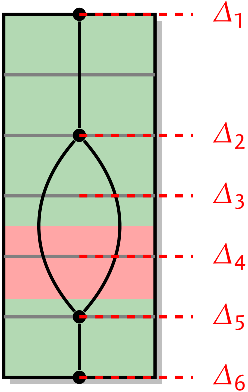
Free-start Collision Attack



- Trails with small key difference
- This allows inactive rounds
- The key schedule repeats after 5 block
  - Collisions with the feed-forward

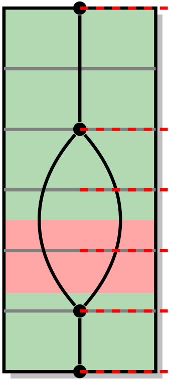
- Algorithm finds 20-round characteristics
- Practical attack

Free-tweak Free-start Near-collision Attack



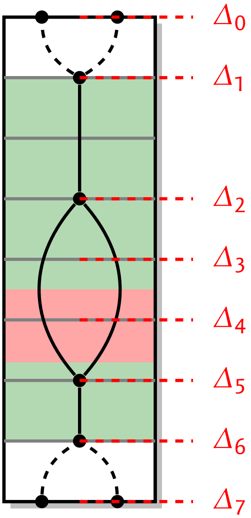
- Trails with small key difference and small tweak difference
- This allows one round with inactive subkeys
- Controlled characteristic for 24 rounds
  - 5 active bits in the output ($\Delta_1 \oplus \Delta_6$)
- Algorithm work for the middle rounds
- Practical attack
- Can be extended to partial-collisions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

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]	$22^\dagger$	$2^{253.8}$
		(biclique)	$37^\dagger$	$2^{255.7}$
Free-tweak partial-collision	12	[YCJW12]	32	2^{85}
Collision	0		12	$\approx 2^{100}$
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 (fewer 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

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

References

Analysis of Differential Attacks in ARX Constructions
Asiacrypt 2012

Construction of Differential Characteristics in ARX Designs
Application to Skein
Crypto 2013

Tools available

- Code and documentation available at:
  `http://www.di.ens.fr/~leurent/arxtools.html`

Extra slides

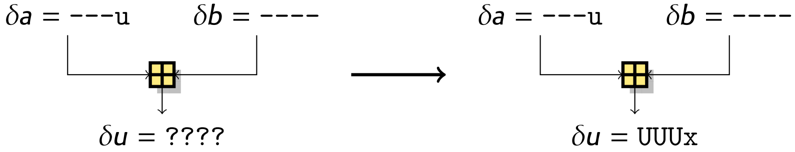
2-bit constraints

2.5-bit constraints

Propagation example

2-bit constraints

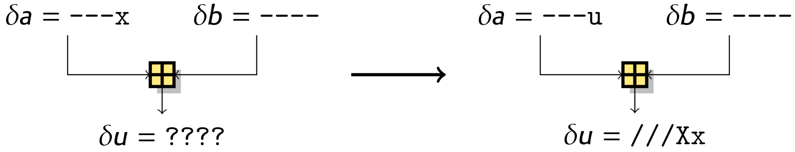
- 1.5-bit conditions extract information from carries when the xor difference is known
- What if the xor-sum is not known?



- Several possibilities:
  - δu = ---u
  - δu = --un
  - δu = -unn
  - δu = unnn
  - δu = 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 /

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	1
0^{03}	✓	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	
0^{0C}	-	✓	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	
0^{u3}	-	-	✓	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	
0^{uC}	-	-	-	✓	-	-	-	-	-	-	-	-	-	-	-	-	-	-	
0^{n3}	-	-	-	-	✓	-	-	-	-	-	-	-	-	-	-	-	-	-	
0^{nC}	-	-	-	-	-	✓	-	-	-	-	-	-	-	-	-	-	-	-	
0^{13}	-	-	-	-	-	-	✓	-	-	-	-	-	-	-	-	-	-	-	
0^{1C}	-	-	-	-	-	-	-	✓	-	-	-	-	-	-	-	-	-	-	

⋮

Back to the talk

Propagation example

Example System

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

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

Back to the talk

Propagation example

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

Propagation example

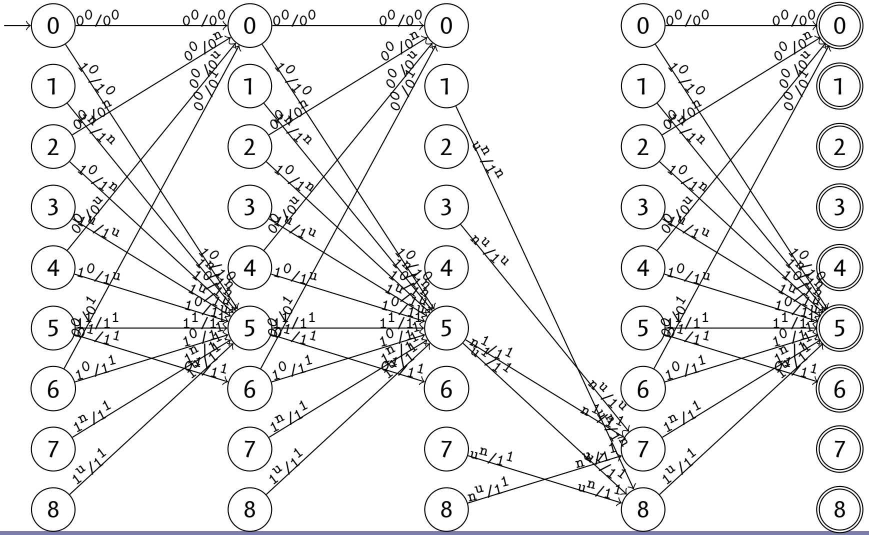
Example System

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

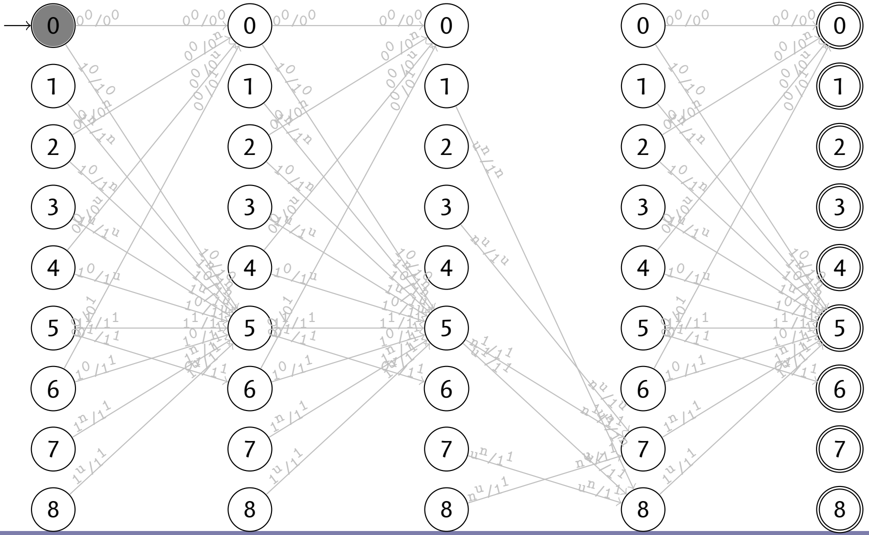
1. Compute transitions
2. Build a graph with initial constraints
   - Example: $\delta(a, a') = \texttt{-x--}, \delta(u, u') = \texttt{----}$
3. Identify paths
4. Paths give new constraints
   - Example: $\delta(a, a') = \texttt{1x1-}, \delta(u, u') = \texttt{111-}$

Back to the talk

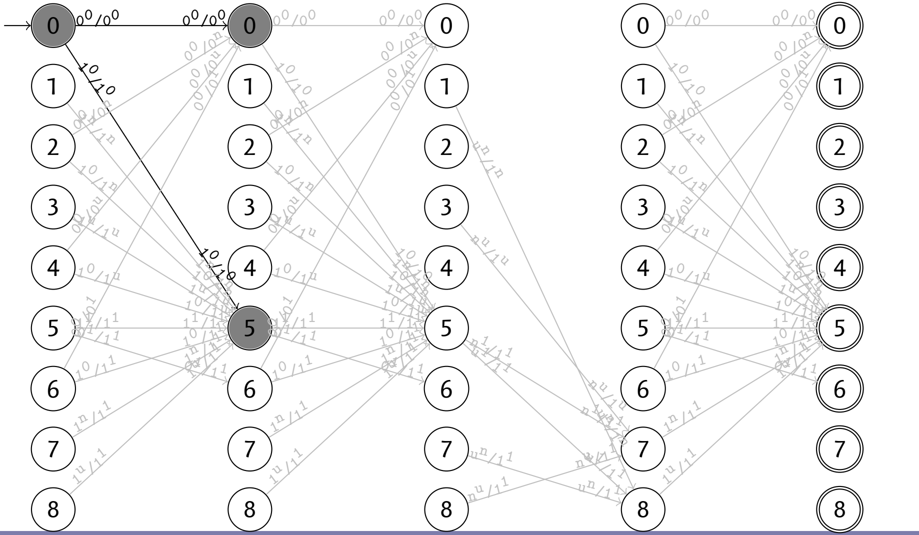
Propagation example



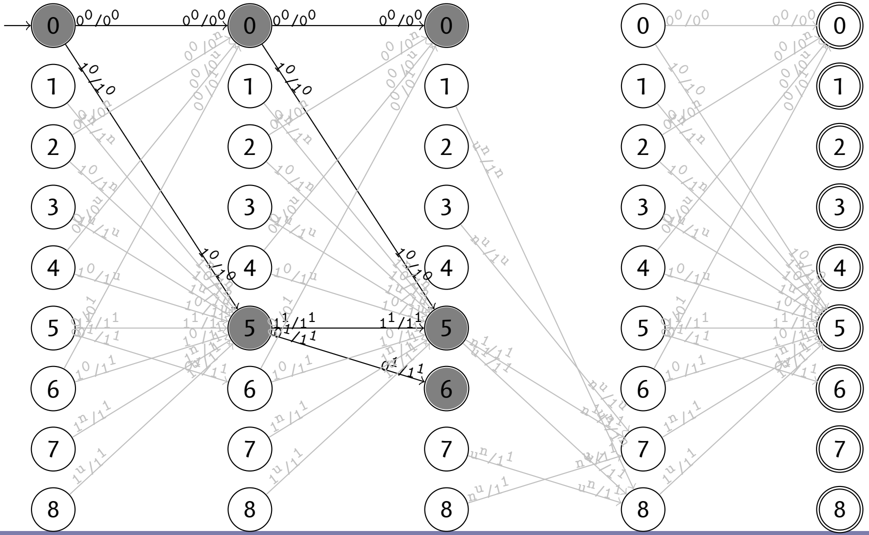
Propagation example



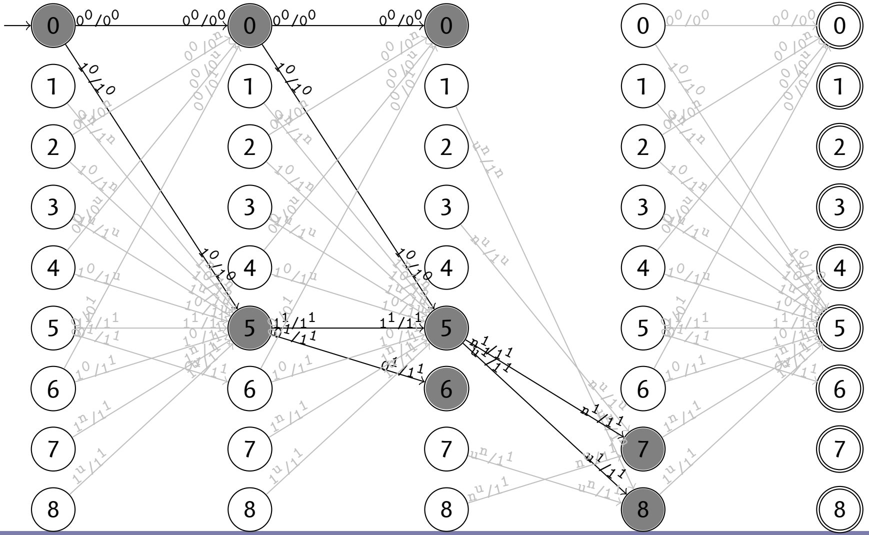
Propagation example



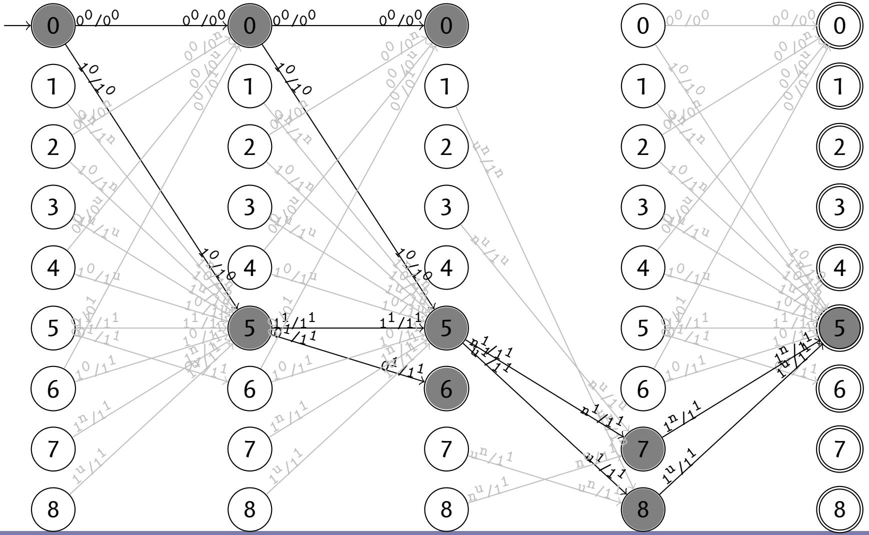
Propagation example



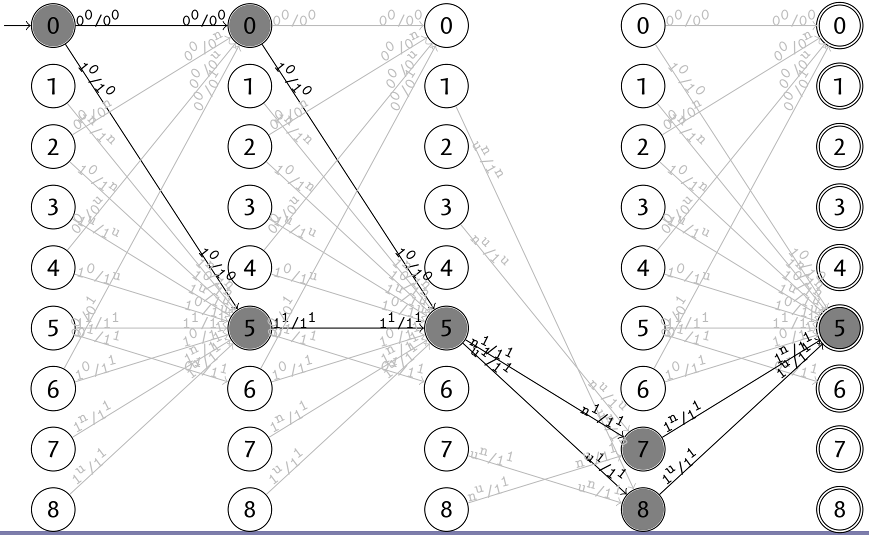
Propagation example



Propagation example



Propagation example



Propagation example

Example System

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

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