PRACTICAL COLLISIONS FOR WIDEA-8

Kerem Varıcı
KU Leuven / COSIC

Session ID: CRYP-R31
Session Classification: Advanced

OUTLINE

- Introduction
- Collisions for the WIDEA-8
- Results and Remarks

OUTLINE

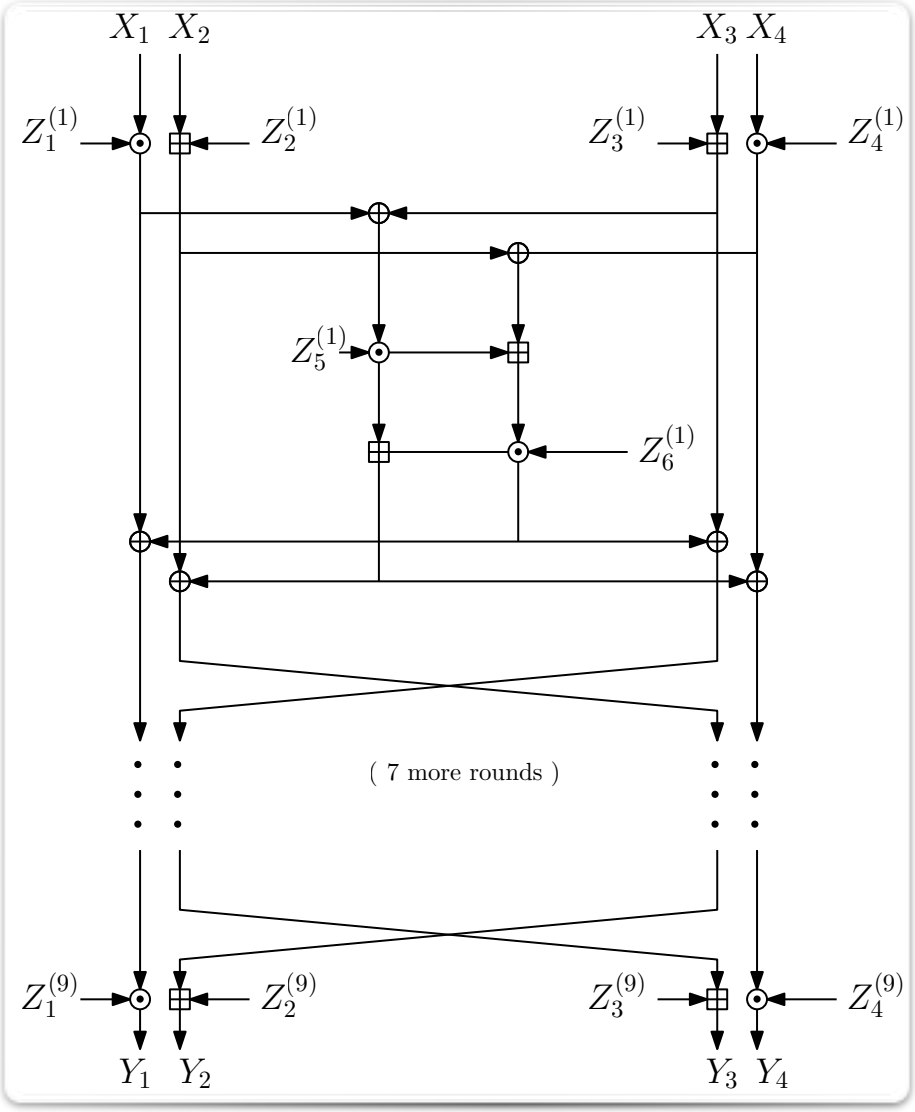
- Introduction
- Collisions for the WIDEA-8
- Results and Remarks

Introduction (WIDEA)

- Widea – a family of block ciphers
- Inspired by IDEA block cipher
- Designed by Junod and Macchetti
- Presented in FSE'09
- Suggested to use as compression function for hash function in Davies – Mayer mode
  - Competitive efficiency with most of the hash functions

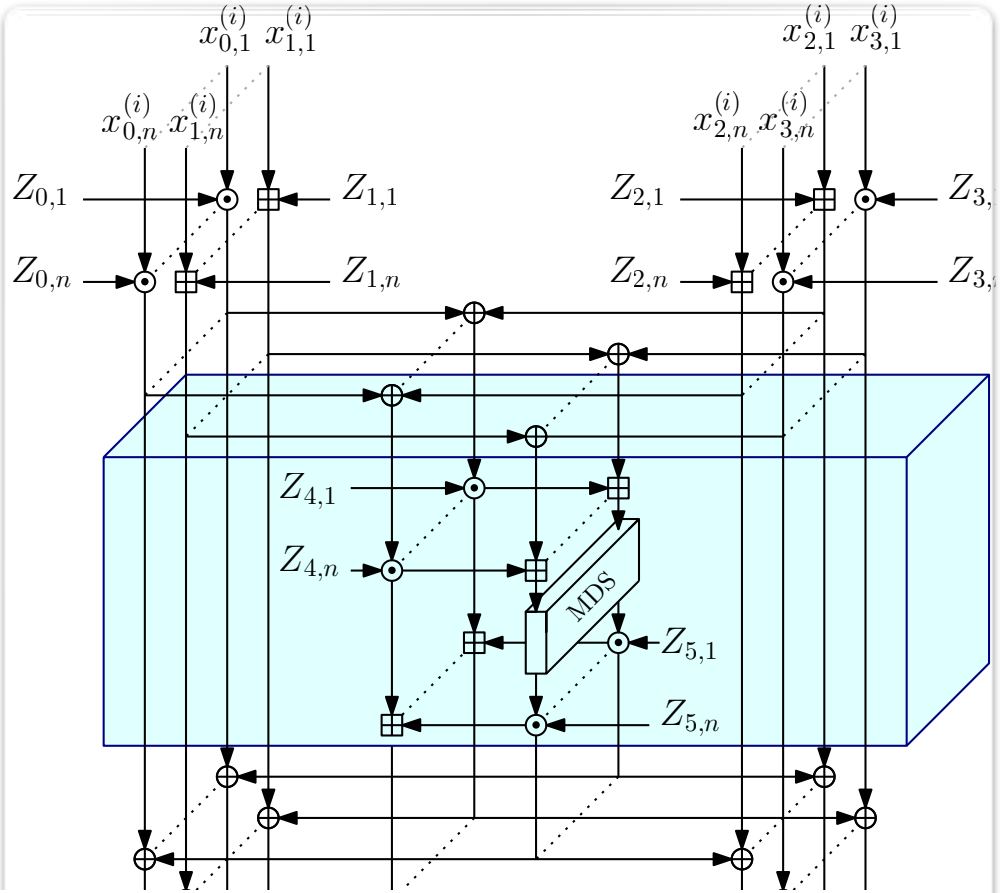
IDEA

- Designed by Lai and Massey in 1991
- 64-bit block size
- 128-bit key size
- 8.5 rounds
- Widely implemented
- 20+ security analysis
- Secure except weak keys



WIDEA-n

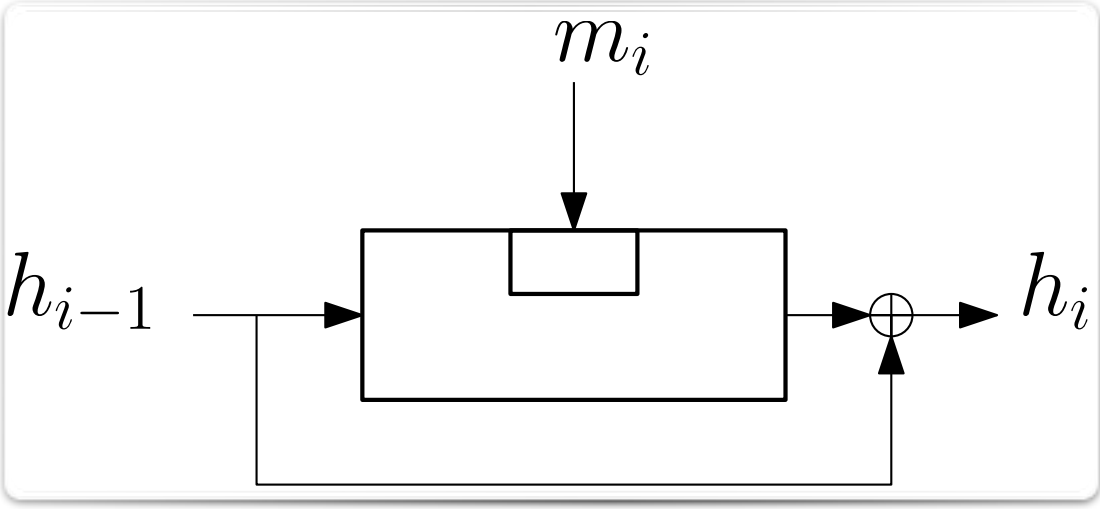
- Composed of n parallel applications of IDEA connected with an MDS matrix.
- 64×n bits block size
- 64×2n key size
- 8.5 rounds



$$
\begin{aligned}
Z_i &= K_i && 0 \le i \le 7 \\
Z_i &= ((((Z_{i-1} \oplus Z_{i-8}) \overset{16}{\boxplus} Z_{i-5}) \overset{16}{\lll} 5) \lll 24) \oplus C_{\frac{i}{8}-1} && 8 \le i \le 51, \quad 8 | i \\
Z_i &= ((((Z_{i-1} \oplus Z_{i-8}) \overset{16}{\boxplus} Z_{i-5}) \overset{16}{\lll} 5) \lll 24) && 8 \le i \le 51, \quad 8 \nmid i
\end{aligned}
$$

Davies-Meyer Mode

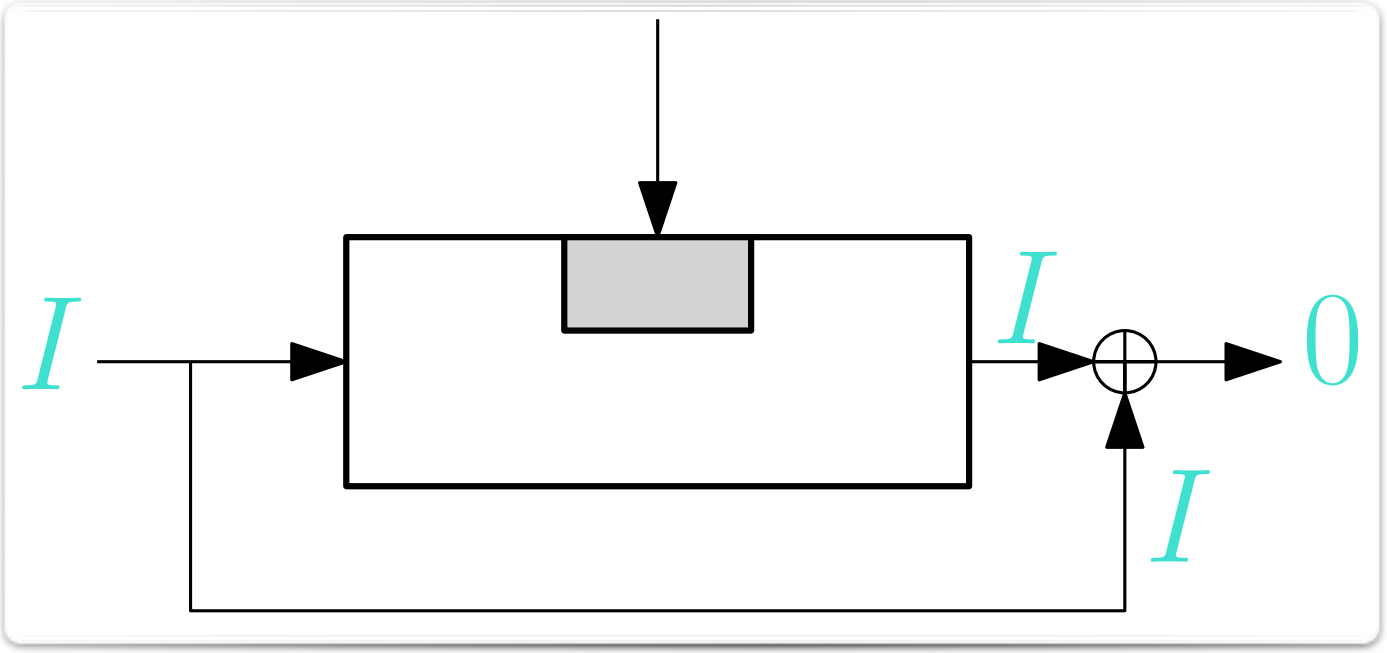
- Proposed to construct hash functions from block ciphers
- Most widely used hash functions, including MD5, SHA-1 and SHA-2 use this construction



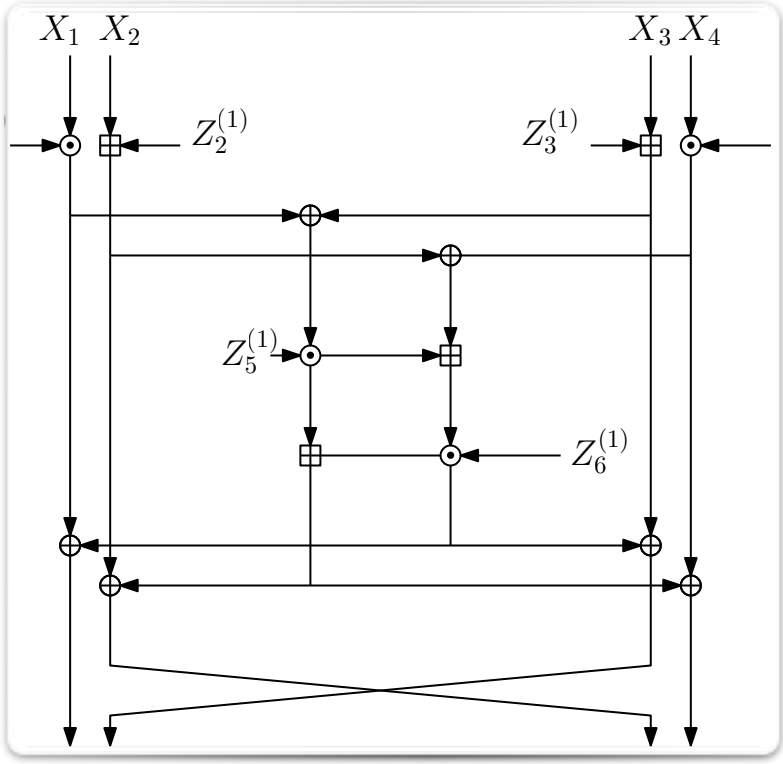
OUTLINE

- Introduction
- Collisions for the WIDEA-8
  - Attack strategy
  - Application of theory to practice
  - Extending the attack to full WIDEA-8
- Results and Remarks

Attack strategy – I

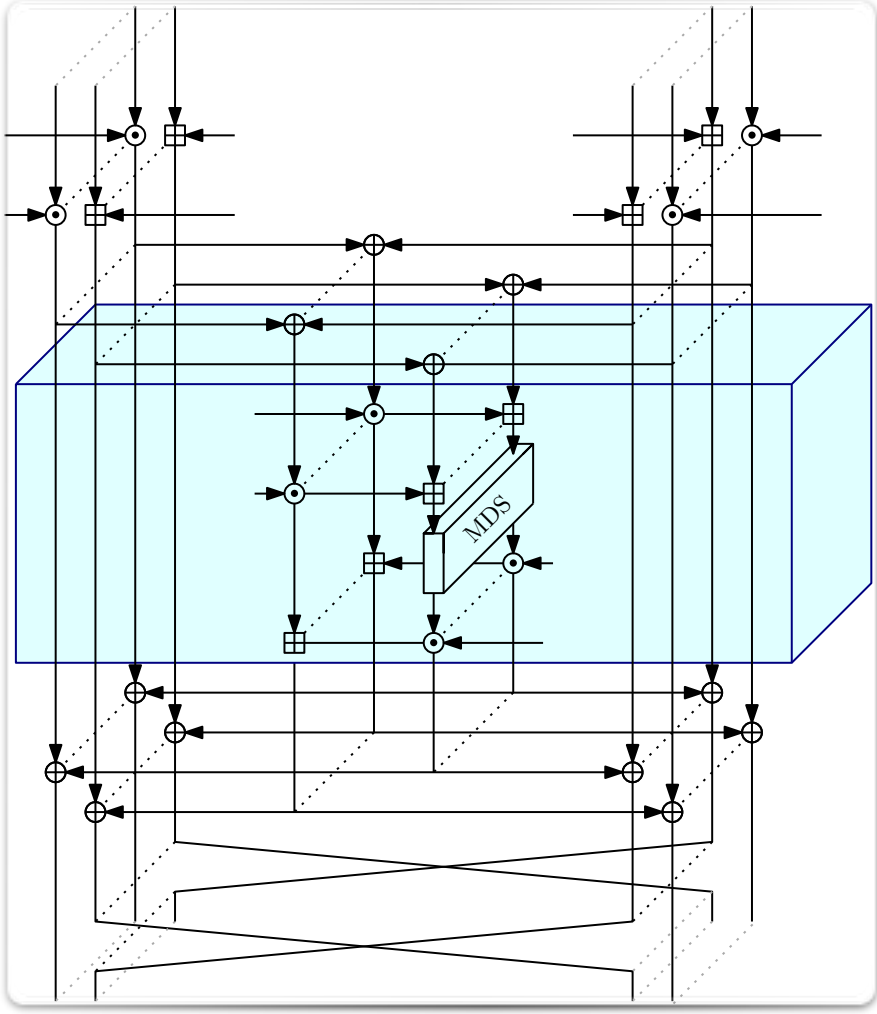


Attack strategy – II



Characteristic	Z_1	Z_4	Z_5	Z_6
(0,0,0,Δ) ⇒ (Δ,Δ,Δ,0)	–	±1	–	±1
(0,0,Δ,0) ⇒ (Δ,0,0,0)	–	–	±1	±1
(0,0,Δ,Δ) ⇒ (0,Δ,Δ,0)	–	±1	±1	–
(0,Δ,0,0) ⇒ (Δ,Δ,0,Δ)	–	–	–	±1
(0,Δ,0,Δ) ⇒ (0,0,Δ,Δ)	–	±1	–	–
(0,Δ,Δ,0) ⇒ (0,Δ,0,Δ)	–	–	±1	–
(0,Δ,Δ,Δ) ⇒ (Δ,0,Δ,Δ)	–	±1	±1	±1
(Δ,0,0,0) ⇒ (0,Δ,0,0)	±1	–	±1	±1
(Δ,0,0,Δ) ⇒ (Δ,0,Δ,0)	±1	±1	±1	–
(Δ,0,Δ,0) ⇒ (Δ,Δ,0,0)	±1	–	–	–
(Δ,0,Δ,Δ) ⇒ (0,0,Δ,0)	±1	±1	–	±1
(Δ,Δ,0,0) ⇒ (Δ,0,0,Δ)	±1	–	±1	–
(Δ,Δ,0,Δ) ⇒ (0,Δ,Δ,Δ)	±1	±1	±1	±1
(Δ,Δ,Δ,0) ⇒ (0,0,0,Δ)	±1	–	–	±1
(Δ,Δ,Δ,Δ) ⇒ (Δ,Δ,Δ,Δ)	±1	±1	–	–

Attack strategy – III



Observation 1

The parallel instances of IDEA are only connected by the MDS matrix in the MA-box and hence if we can find a characteristic for one lane where the MA-box is never active, the attack is reduced of attacking only one lane instead of all eight.

Observation 2

Given any eight consecutive subkeys $\{Z_{i+1}, Z_{i+2}, \ldots, Z_{i+8}\}$ *it is possible to construct the whole set of subkeys.*

Theory to practice

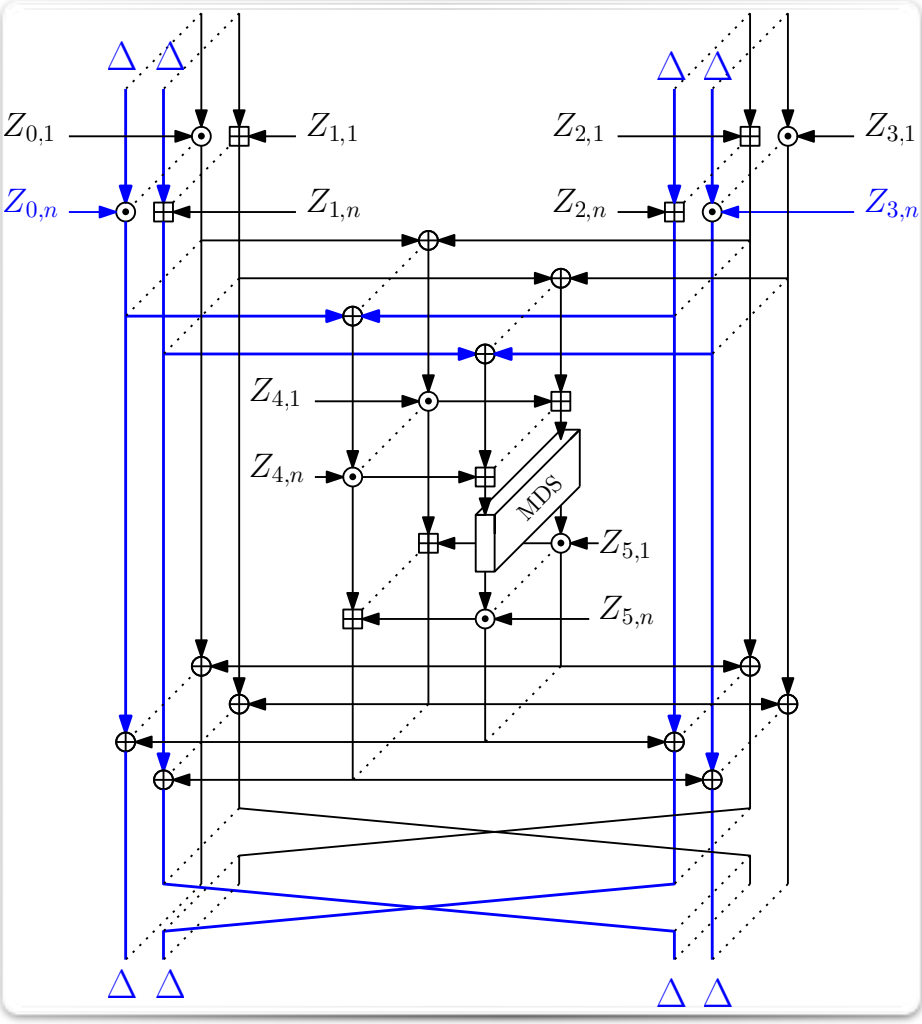


Table 1. Subkeys for WIDEA-8 when $Z_{25} = Z_{26} = \ldots = Z_{32}$

i	$z_{i,1}$	$z_{i,2}$	$z_{i,3}$	$z_{i,4}$	$z_{i,5}$	$z_{i,6}$	$z_{i,7}$	$z_{i,8}$
0	0000	E7FD	1444	6810	8B79	2822	47C8	0200
3	0000	E7FE	06F8	0000	0000	0000	0000	0000
6	0000	0001	F2E9	AFF7	0600	0000	0000	0000
9	0000	E7FF	FC58	0000	0000	0000	0000	0000
12	0000	F001	0520	0000	0000	0000	0000	0000
15	0000	0FFF	FAE0	0000	0000	0000	0000	0000
18	0000	F001	0520	0000	0000	0000	0000	0000
21	0000	0FFF	FAE0	0000	0000	0000	0000	0000
24	0000	F001	0520	0000	0000	0000	0000	0000
27	0000	0000	0000	0000	0000	0000	0000	0000
30	0000	0000	0000	0000	0000	0000	0000	0000
33	0000	0000	0000	0000	0000	0000	0000	0000
36	0000	0000	0000	0000	0000	0000	0000	0000
39	0000	0000	0000	0000	0000	0000	0000	0000
42	0000	0000	0000	0000	0015	E080	0B00	0000
45	4891	8264	0000	0000	0000	0000	00AF	5C00

Extending the attack to full WIDEA-8

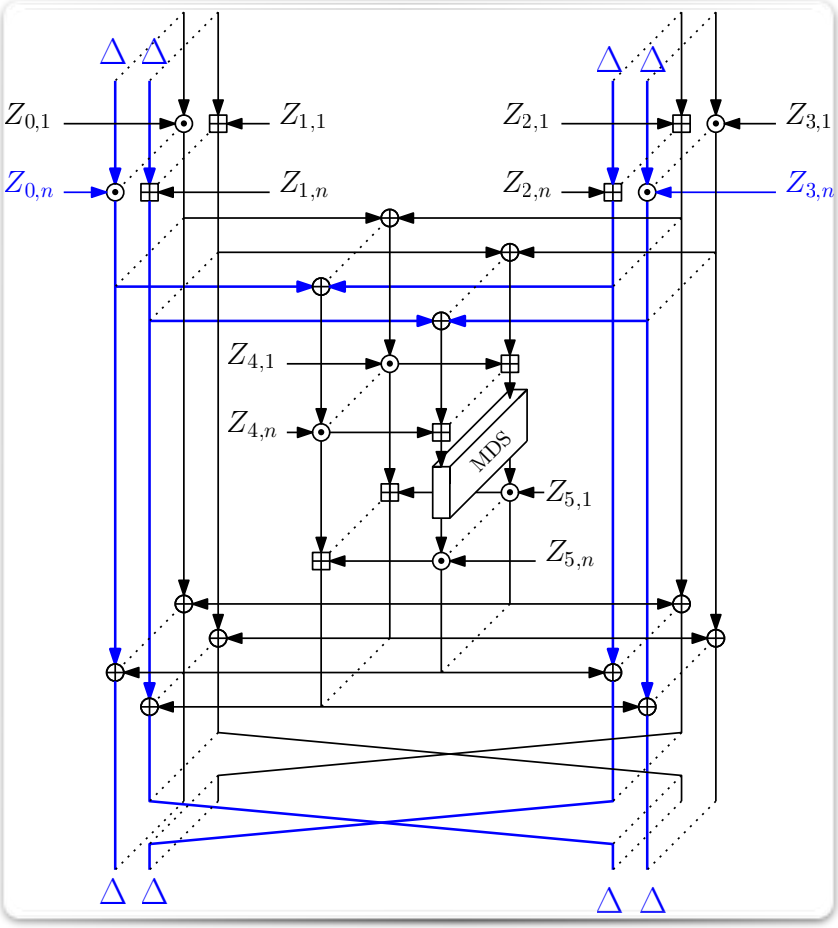


Table 4. Subkeys for WIDEA-8 when $Z_{33} = Z_{34} = \ldots = Z_{40}$

i	$z_{i,1}$	$z_{i,2}$	$z_{i,3}$	$z_{i,4}$	$z_{i,5}$	$z_{i,6}$	$z_{i,7}$	$z_{i,8}$
0	3209	680D	AB9C	470D	6357	300A	C7C8	42B4
3	0000	BFFB	22E2	E13A	8FBC	B209	0800	0000
6	0000	2806	E120	46FD	F980	0000	0000	0000
9	0000	67FF	9C35	7EB3	3108	31C0	0400	0000
12	0000	F001	5517	790A	1080	0000	0000	0000
15	0000	27FA	E93A	9F2E	F600	0000	0000	0000
18	0000	D806	CECC	48A1	0B80	0000	0000	0000
21	0000	0FFF	72E5	CF97	FB00	0000	0000	0000
24	0000	F001	4521	1838	0680	0000	0000	0000
27	0000	0000	0000	0000	0000	0000	0000	0000
30	0000	0000	0000	0000	0000	0000	0000	0000
33	0000	0000	0000	0000	0000	0000	0000	0000
36	0000	0000	0000	0000	0000	0000	0000	0000
39	0000	0000	0000	0000	0000	0000	0000	0000
42	0000	0000	0000	0000	0000	0000	0000	0000
45	0000	0000	0000	0000	0000	0000	0000	0000
48	F7BA	0000	0000	0000	0000	0000	0000	0000
51	0000	0000	0003	C018	1C60	0100	0000	0000

Extending the attack to full WIDEA-8

$z_{0,8} = \texttt{0x42B4}$, $z_{2,8} = \texttt{0x7e49}$, $z_{8,8} = \texttt{0x2600}$ and $z_{49,8} = \texttt{0x5E00}$

$$(x_{2,8}^{(1)} \overset{16}{\boxplus} z_{2,8}) \oplus ((x_{2,8}^{(1)} \oplus \Delta) \overset{16}{\boxplus} z_{2,8}) = \Delta \quad (1)$$

$$(x_{8,8}^{(2)} \overset{16}{\boxplus} z_{5,8}) \oplus ((x_{8,8}^{(2)} \oplus \Delta) \overset{16}{\boxplus} z_{8,8}) = \Delta \quad (2)$$

$$(x_{49,8}^{(8)} \overset{16}{\boxplus} z_{49,8}) \oplus ((x_{49,8}^{(8)} \oplus \Delta) \overset{16}{\boxplus} z_{49,8}) = \Delta \quad (3)$$

$$(x_{0,8}^{(1)} \odot \texttt{0x42B4}) \oplus ((x_{0,8}^{(1)} \oplus \Delta) \odot \texttt{0x42B4}) = \Delta \quad (4)$$

$$(x_{i,8}^{(j)} \odot \texttt{0x0000}) \oplus ((x_{i,8}^{(j)} \oplus \Delta) \odot \texttt{0x0000}) = \Delta \quad (5)$$

Collision Example

Table 4. A collision example for full WIDEA-8 in hexadecimal

State							
2C7A	0866	9F38	C148	3FB1	7BDA	0232	9054
E56C	8780	3E0D	96F3	6D1D	F028	907A	CA77
DDB6	AC09	77E4	D4C5	6715	E3CA	165A	3396
A835	DACB	CA5D	CC01	5270	F382	D7D7	7873
State'							
2C7A	0866	9F38	C148	3FB1	7BDA	0232	C874
E56C	8780	3E0D	96F3	6D1D	F028	907A	9257
DDB6	AC09	77E4	D4C5	6715	E3CA	165A	6BB6
A835	DACB	CA5D	CC01	5270	F382	D7D7	2053
$M_1 = M_2$							
3209	680D	AB9C	470D	6357	300A	C7C8	42B4
0000	5801	97F4	D0DA	0371	04E1	F400	0000
0000	7FF8	5C75	B946	131E	6335	CCF1	7E49
0000	BFFB	22E2	E13A	8FBC	B209	0800	0000
0000	4FF7	753E	2805	3E23	80E2	0C00	0000
0000	E7F9	7FC3	1818	DE12	EF37	C8F4	C1FF
0000	2806	E120	46FD	F980	0000	0000	0000
0000	4008	8FE9	8005	8A98	FF6E	F800	0000
WIDEA-8(State , M_1) $\oplus$ M_1 = WIDEA-8(State', M_2) $\oplus$ M_2							
2C06	6743	87F8	775D	8AB8	5957	226C	4F0F
626F	934B	949F	7195	333A	997A	0D1E	9A32
3D2C	3435	3861	E7CB	2198	8074	94DA	2C26
2544	AD24	4881	E8DC	2344	015F	B015	6D81

Results

target	rounds	time	attack type
comp. function	7	1	free-start collision
comp. function	8.5 (full)	1	free-start near-collision
comp. function	8.5 (full)	$2^{13.53}$	free-start collision

ANY QUESTIONS ?

Finding Collisions for Round-Reduced SM3

Florian Mendel, Tomislav Nad, Martin Schläffer

Institute for Applied Information Processing and Communications (IAIK)
Graz University of Technology
Inffeldgasse 16a, A-8010 Graz, Austria

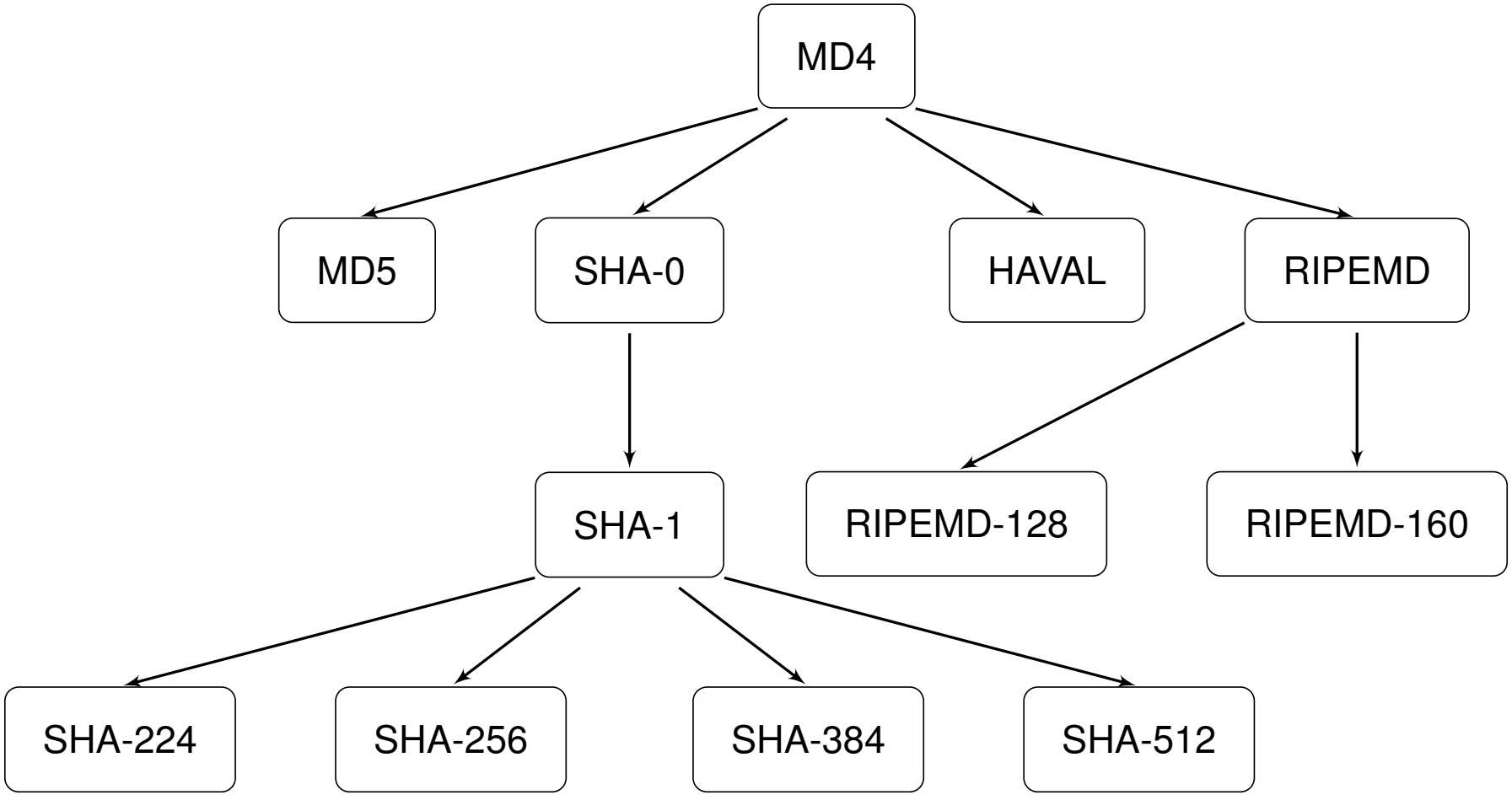
martin.schlaeffer@iaik.tugraz.at

CT-RSA 2013

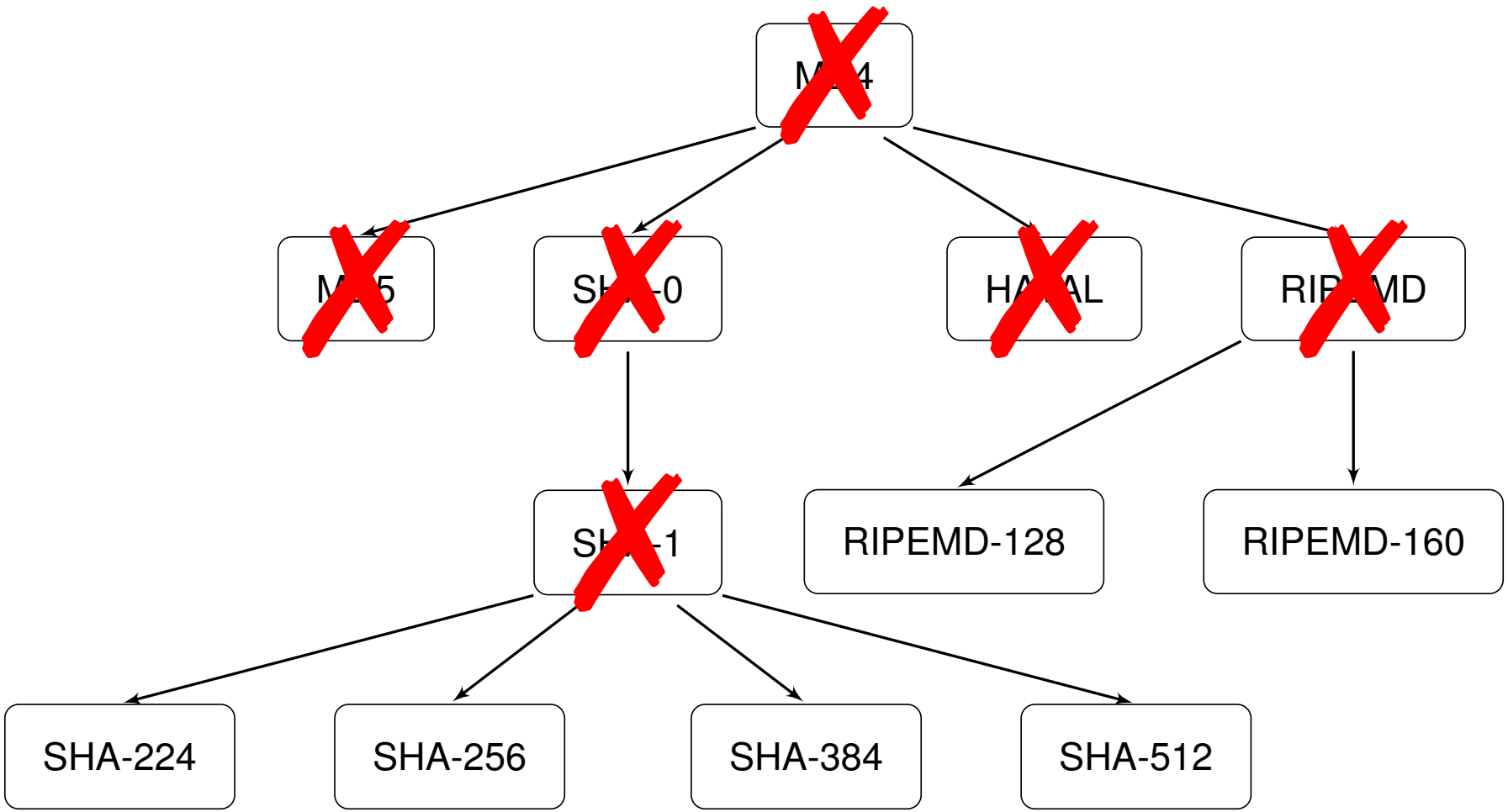
Outline

Outline

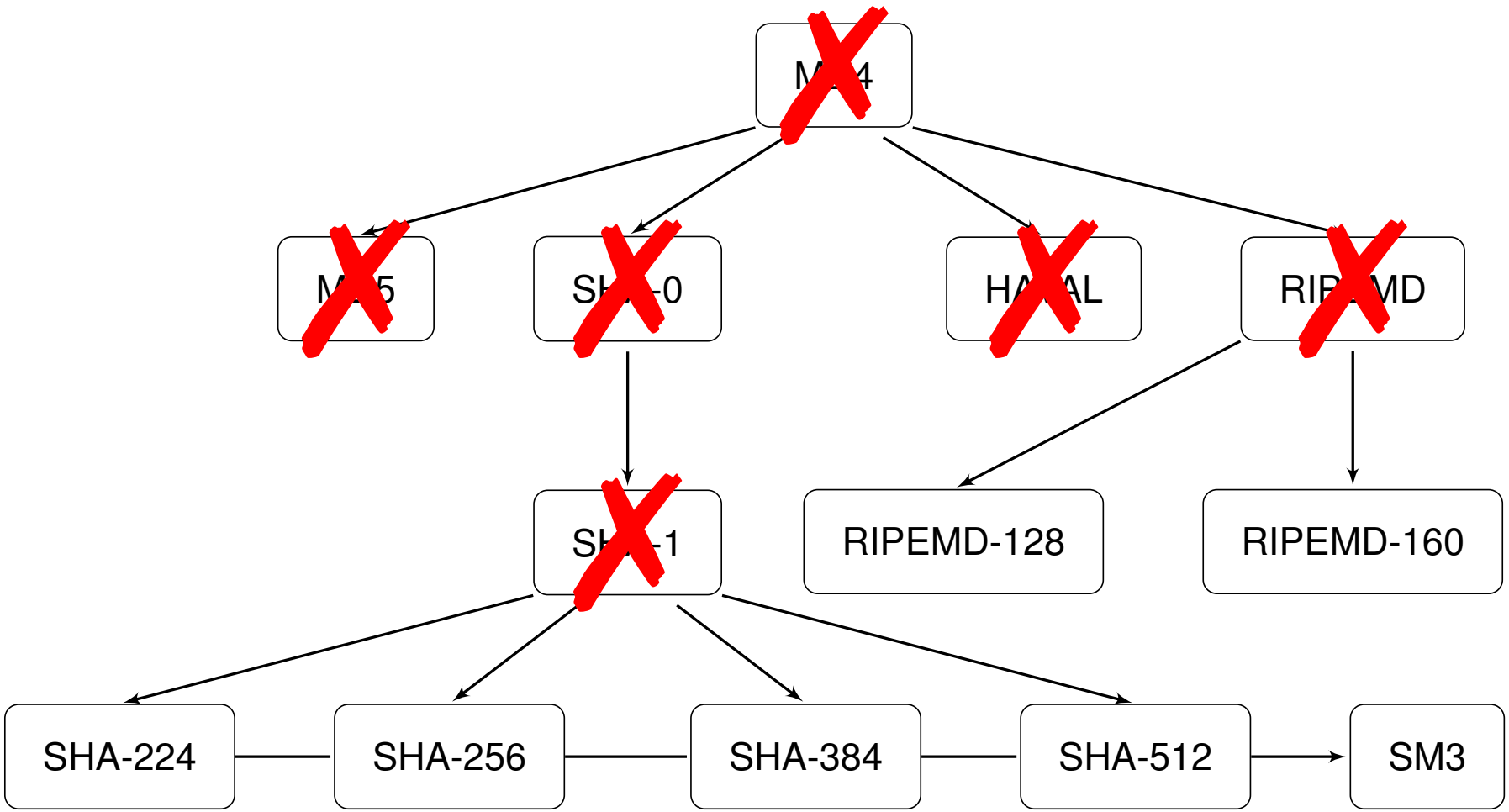
The MD4 Family of Hash Functions



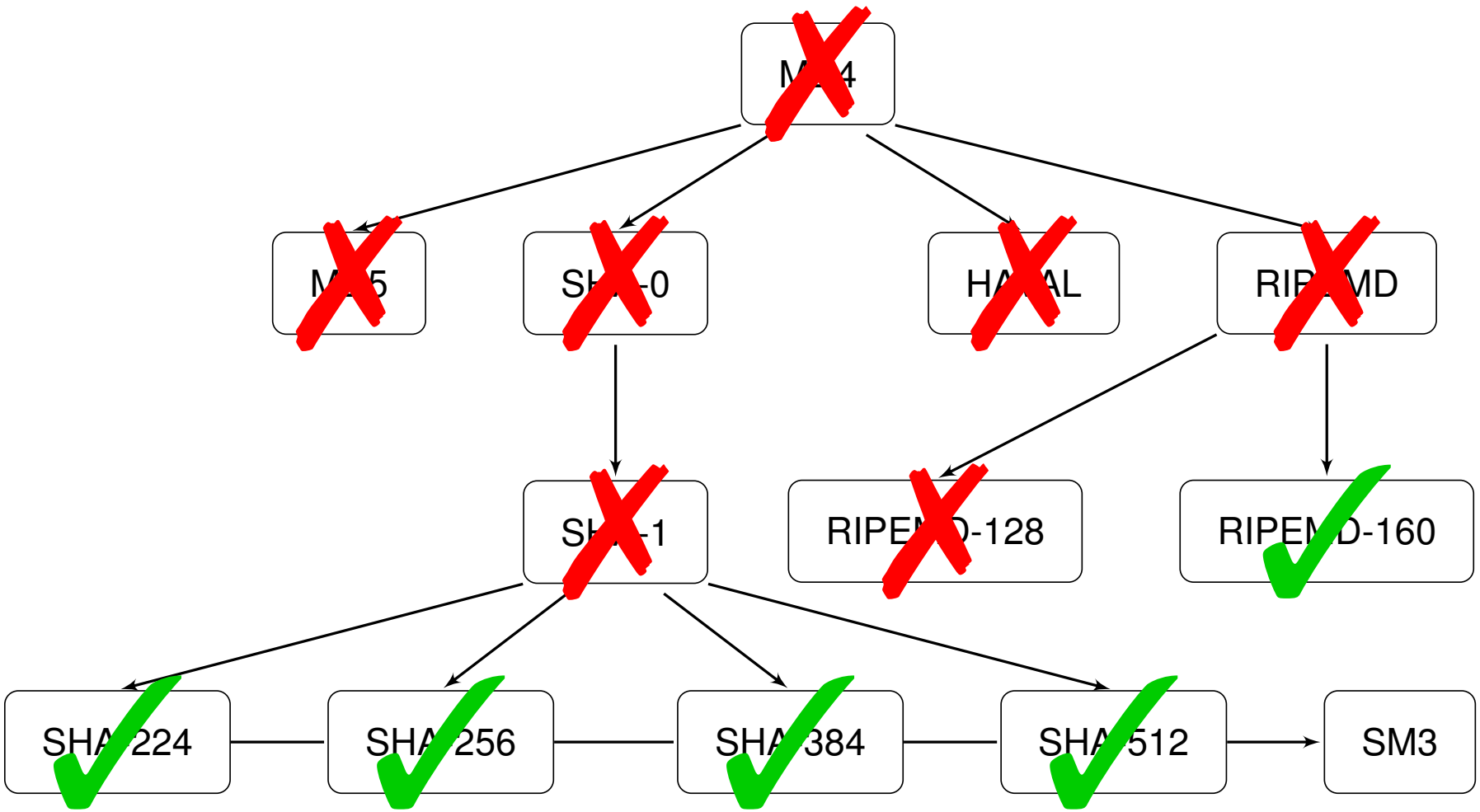
The MD4 Family of Hash Functions



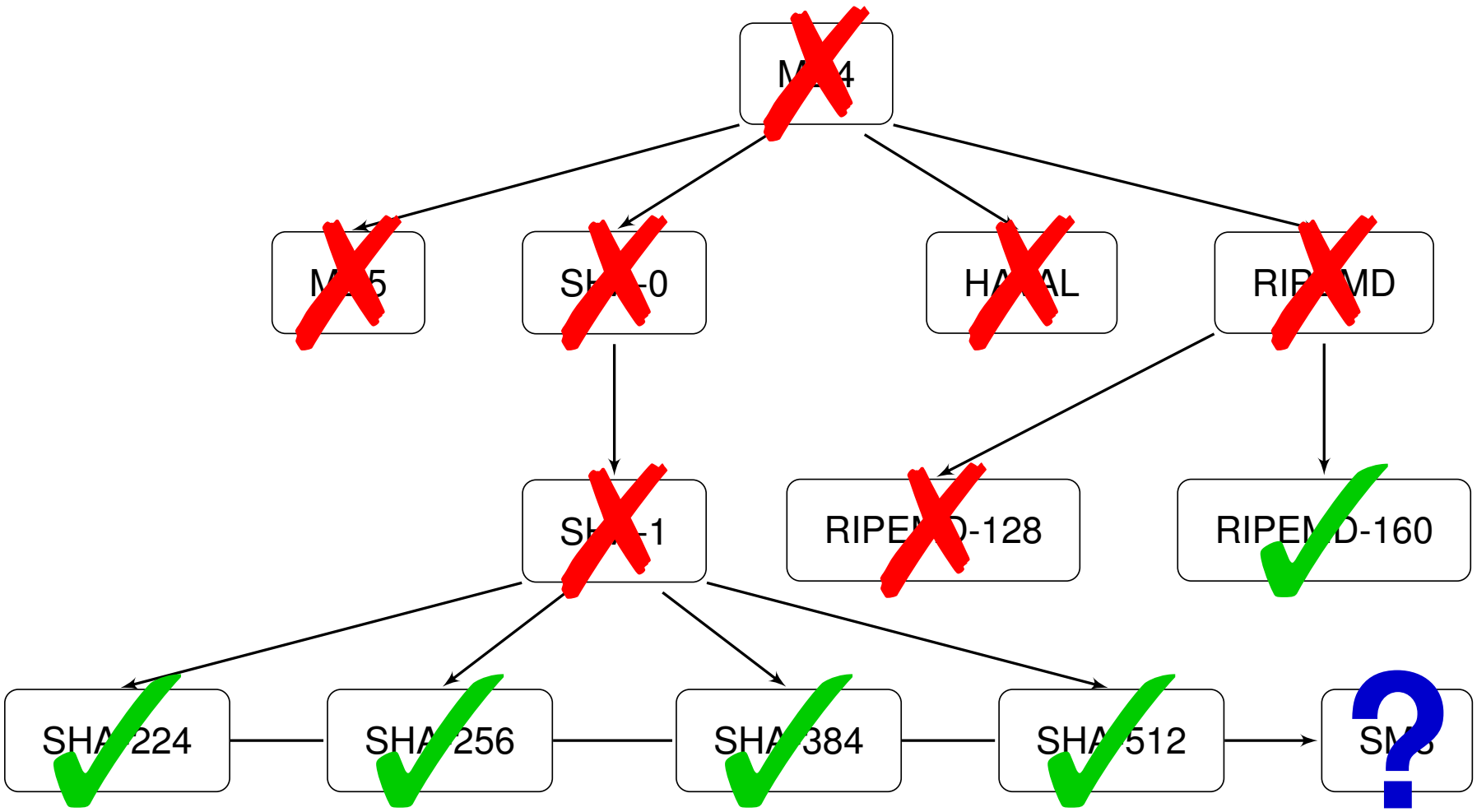
The MD4 Family of Hash Functions



The MD4 Family of Hash Functions

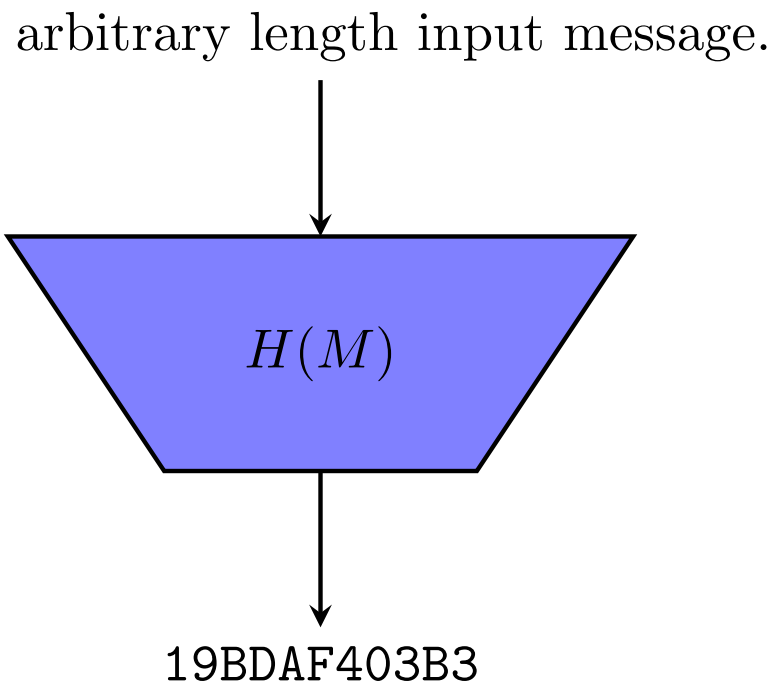


The MD4 Family of Hash Functions



Outline

Hash Functions

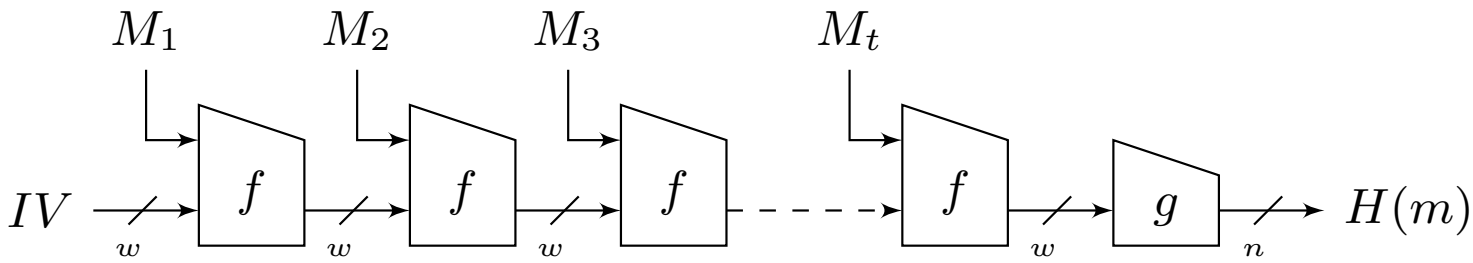


Protect short hash value instead of long text.

Main Security Requirements

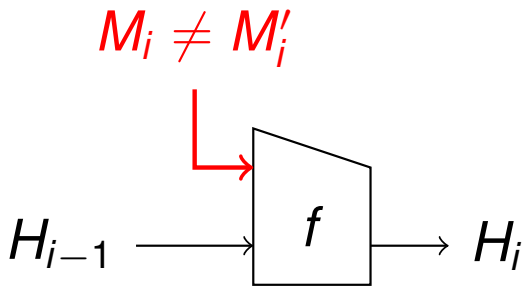
- Preimage Resistance
  - given $h(m)$ find m
  - generic complexity: 2^n
- Second-Preimage Resistance
  - given $m, h(m)$ find m' with $m \neq m'$ and $h(m) = h(m')$
  - generic complexity: 2^n
- Collision Resistance
  - find m, m' with $m \neq m'$ and $h(m) = h(m')$
  - generic complexity: $2^{n/2}$
- No non-random properties

Iterated Hash Function Construction



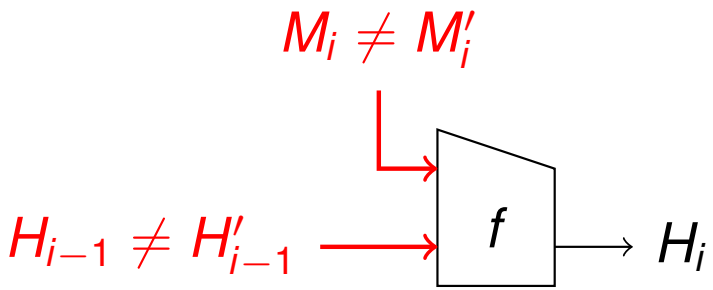
- Most hash functions use some kind of iteration
  - compression function f
  - output transformation g
  - chaining value size $w \geq n$
- Strength depends on f, g, w
  - smaller w needs stronger f
- Analyze building blocks
  - collision resistance of f
  - non-random properties of f

Collisions on the Compression Function



- semi-free-start collision:
  - attacker can choose value of chaining input H_{i-1}
  - $f(M_i, H_{i-1}) = f(M_i', H_{i-1}),\ M_i \neq M_i'$

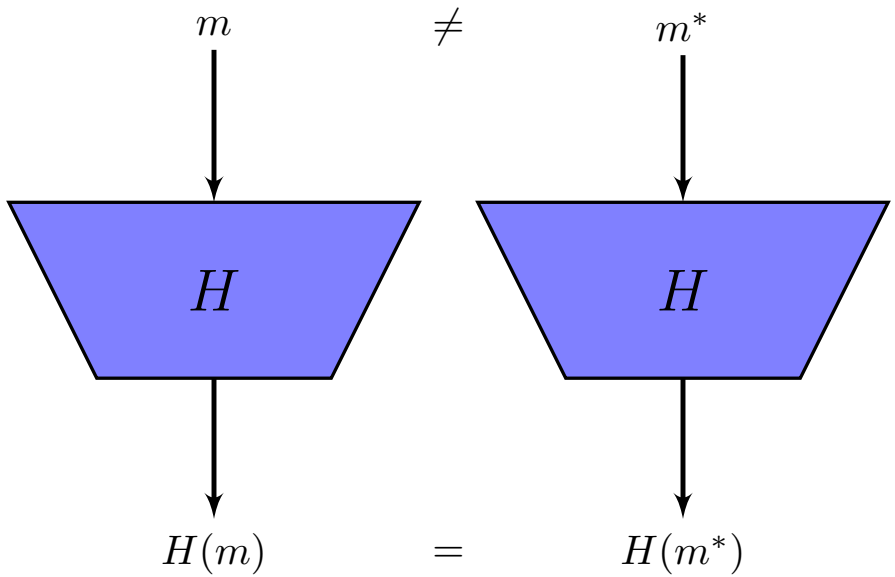
Collisions on the Compression Function



- semi-free-start collision:
  - attacker can choose value of chaining input H_{i-1}
  - $f(M_i, H_{i-1}) = f(M'_i, H_{i-1}),\ M_i \neq M'_i$
- free-start collision:
  - attacker can choose difference and value of chaining input H_{i-1}
  - $f(M_i, H_{i-1}) = f(M'_i, H'_{i-1}),\ M_i \neq M'_i, H_{i-1} \neq H'_{i-1}$

Outline

Collision Attacks

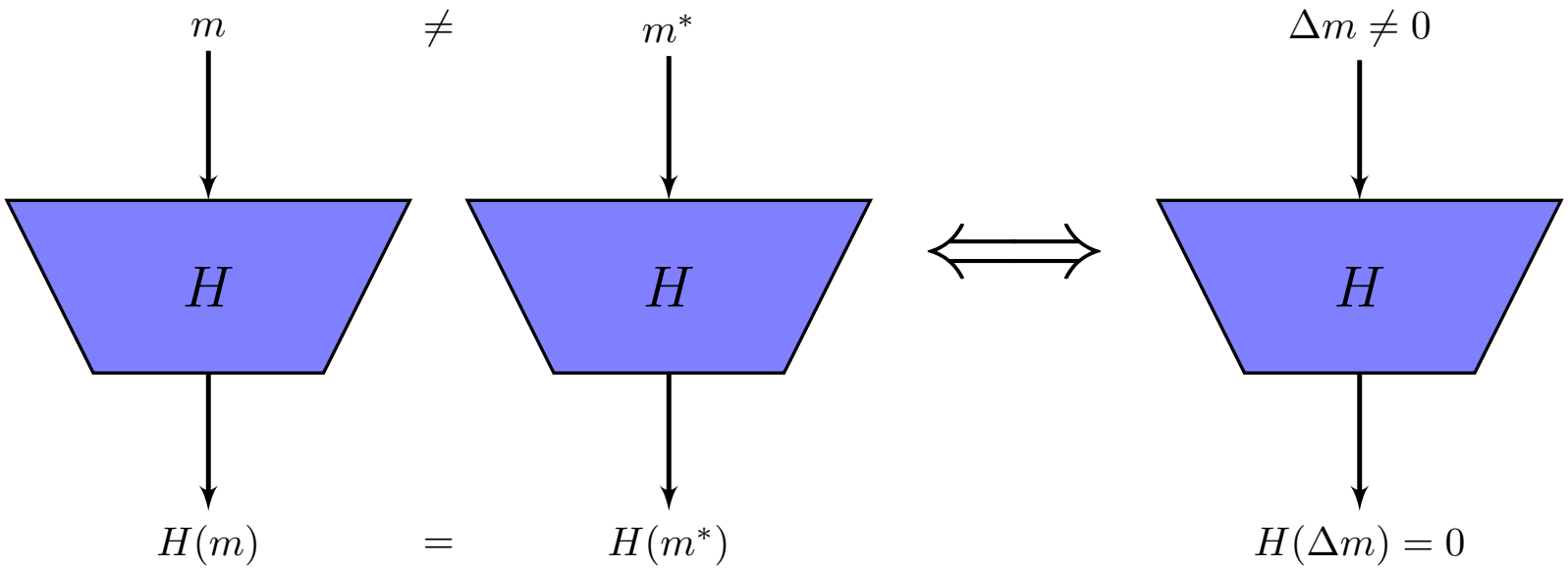


- Find two different messages which result in the same hash value:

$$m \neq m^* \text{ with } H(m) = H(m^*)$$

- Birthday effect applies: $2^{n/2}$

Collision Attacks (Differential View)



- Find two different messages which result in the same hash

$$m, \Delta m \text{ with } \Delta m \neq 0 \text{ and } H(\Delta m) = 0$$

- Usually XOR differences are used:

  $\Delta m = m \oplus m^*$ and $H(\Delta m) = H(m) \oplus H(m^*)$

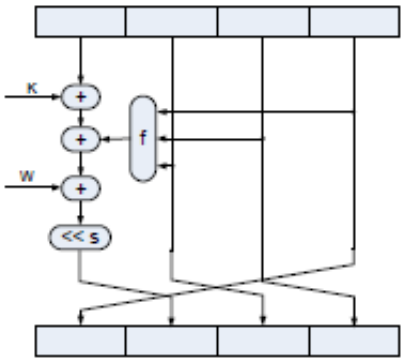
Differential Characteristic

- How to find m, Δm?
- Find differential characteristic (trail, path)
  - using (semi-) automatic tools
  - determines Δm
  - holds with high probability P
  - try $1/P$ random messages
- How to find m more efficiently?
  - we can choose m according to characteristic
  - invert equations in first steps of hash function
  - called message modification

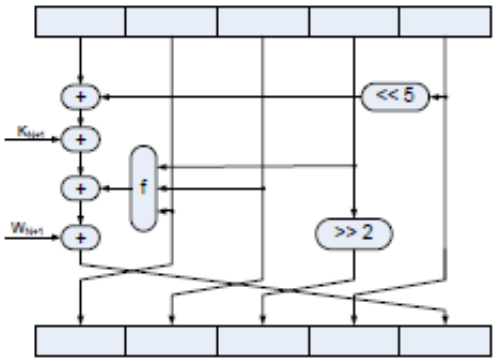
Outline

Design Complexity of the MD4 Family

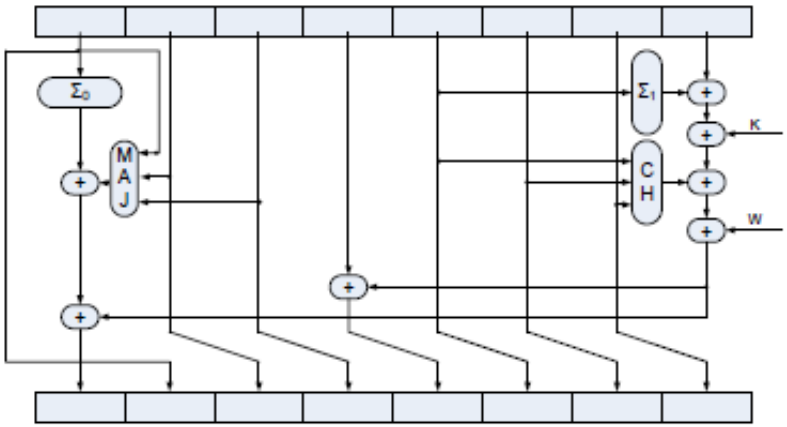
MD4



SHA/SHA-1



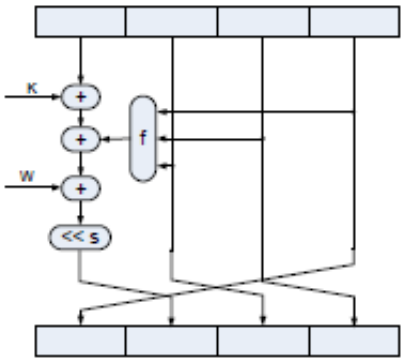
SHA-2 members



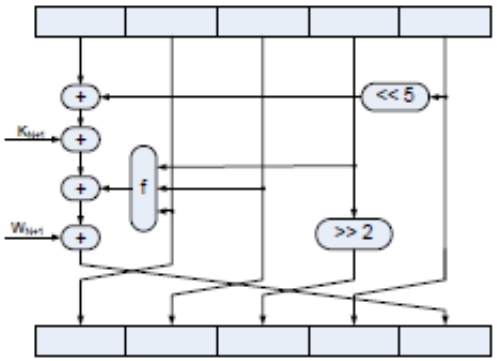
Design Complexity

Design Complexity of the MD4 Family

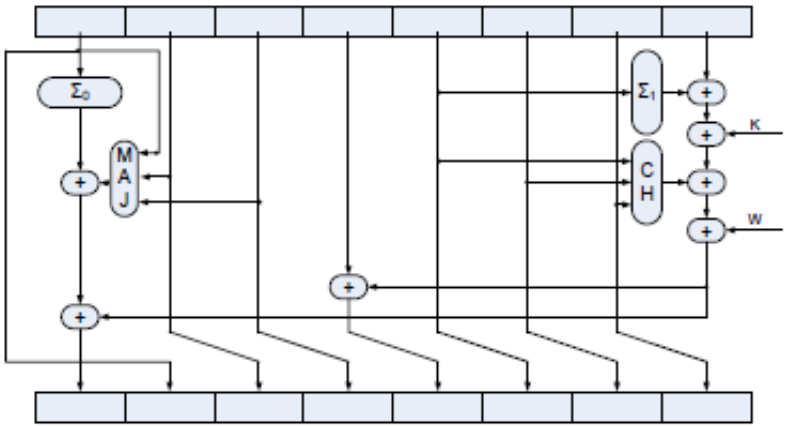
MD4



SHA/SHA-1



SHA-2 members



SM3: too complex to draw :-(

The Hash Function SM3

- Chinese hash function standard [Chi]
  (Chinese Commercial Cryptography Administration Office)
- Designed by Wang et al. after her attacks on SHA-1
- Based on SHA-2
  - more complicated mixing functions
  - additional expanded message words
- Only few cryptanalysis results published:

component	attack	steps	complexity	reference
hash function	preimage	35	2^{249}	[ZWW+11]
compression function	free-start collision	30	$2^{117.1}$	[KSWY12]

Description of SM3

message expansion:

$$W_i = \begin{cases} M_i & 0 \le i < 16 \\ \sigma_0(W_{i-16} \oplus W_{i-9} \oplus W_{i-3} \lll 15) \oplus W_{i-13} \lll 7 \oplus W_{i-6} & 16 \le i < 68 \end{cases}$$

state update transformation:

$$\begin{aligned} T_1 &= (A_{i-1} \lll 12 + E_{i-1} + K_i) \lll 7 \\ T_2 &= H_{i-1} + f_0(E_{i-1}, F_{i-1}, G_{i-1}) + T_1 + W_i \\ A_i &= D_{i-1} + f_1(A_{i-1}, B_{i-1}, C_{i-1}) + (T_1 \oplus A_{i-1} \lll 12) + (W_i \oplus W_{i+4}) \\ E_i &= \Sigma_0(T_2) \\ B_i &= A_{i-1} \\ C_i &= B_{i-1} \lll 9 \\ D_i &= C_{i-1} \\ F_i &= E_{i-1} \\ G_i &= F_{i-1} \lll 19 \\ H_i &= G_{i-1} \end{aligned}$$

(linear) Boolean functions:

$$\begin{aligned} f_{XOR}(X, Y, Z) &= X \oplus Y \oplus Z \\ f_{IF}(X, Y, Z) &= XY \oplus XZ \oplus Z \\ f_{MAJ}(X, Y, Z) &= XY \oplus YZ \oplus XZ \end{aligned}$$

$$\begin{aligned} \sigma_0(X) &= X \oplus (X \lll 15) \oplus (X \lll 23) \\ \Sigma_0(X) &= X \oplus (X \lll 9) \oplus (X \lll 17) \end{aligned}$$

Description of SM3

simplified equations and stored variables:

$$
\begin{aligned}
S_i &= W_{i-16} \oplus W_{i-9} \oplus W_{i-3} \lll 15 \\
P_i &= S_i \oplus S_i \lll 15 \oplus S_i \lll 23 \\
W_i &= P_i \oplus W_{i-13} \lll 7 \oplus W_{i-6} \\
W'_i &= W_i \oplus W_{i+4} \\
L_i &= A_{i-1} \lll 12 + E_{i-1} + K_i \lll 7 \\
F_i &= f_0(A_{i-1}, A_{i-2}, A_{i-3} \lll 9) \\
A_i &= F_i + W'_i + A_{i-4} \lll 9 + (L_i \lll 7 \oplus A_{i-1} \lll 12) \\
G_i &= f_1(E_{i-1}, E_{i-2}, E_{i-3} \lll 19) \\
R_i &= E_{i-4} \lll 19 + L_i \lll 7 + W_i + G_i \\
E_i &= R_i \oplus R_i \lll 9 \oplus R_i \lll 17
\end{aligned}
$$

Description of SM3

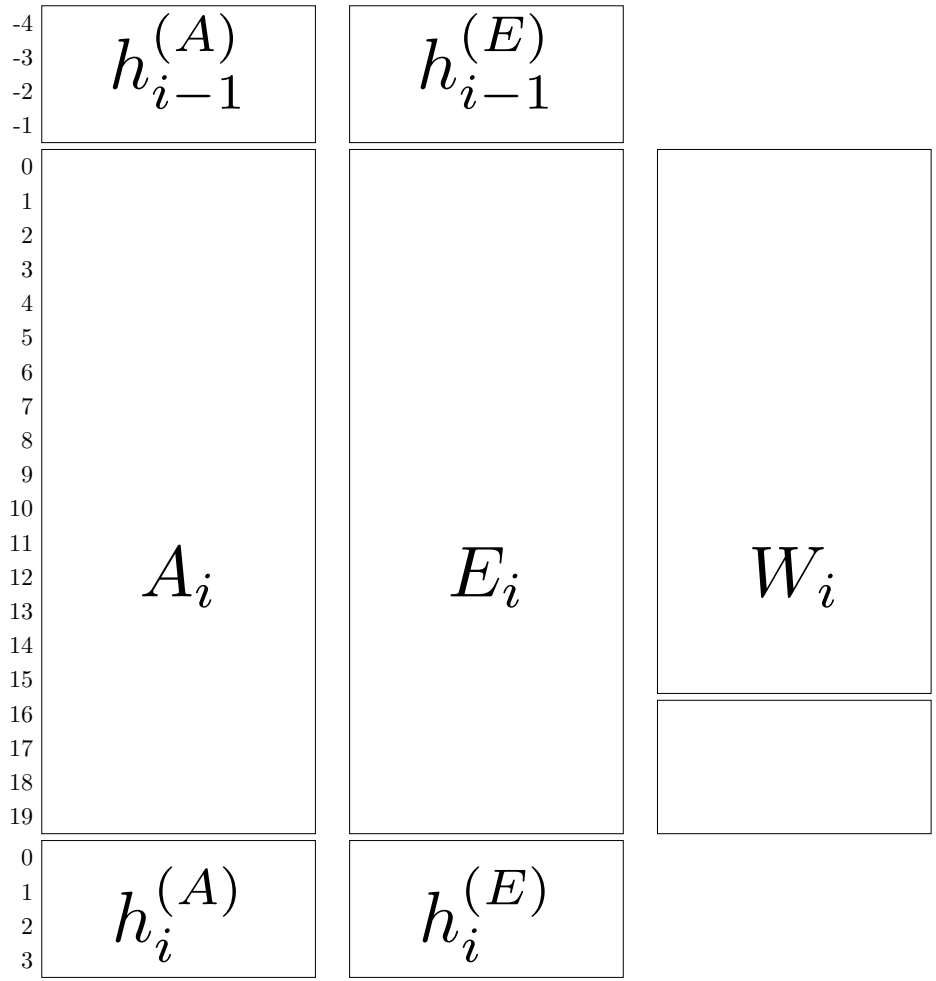
simplified equations and stored variables:

$$
\begin{aligned}
S_i &= W_{i-16} \oplus W_{i-9} \oplus W_{i-3} \lll 15 \\
P_i &= S_i \oplus S_i \lll 15 \oplus S_i \lll 23 \\
W_i &= P_i \oplus W_{i-13} \lll 7 \oplus W_{i-6} \\
W'_i &= W_i \oplus W_{i+4} \\
L_i &= A_{i-1} \lll 12 + E_{i-1} + K_i \lll 7 \\
F_i &= f_0(A_{i-1}, A_{i-2}, A_{i-3} \lll 9) \\
A_i &= F_i + W'_i + A_{i-4} \lll 9 + (L_i \lll 7 \oplus A_{i-1} \lll 12) \\
G_i &= f_1(E_{i-1}, E_{i-2}, E_{i-3} \lll 19) \\
R_i &= E_{i-4} \lll 19 + L_i \lll 7 + W_i + G_i \\
E_i &= R_i \oplus R_i \lll 9 \oplus R_i \lll 17
\end{aligned}
$$

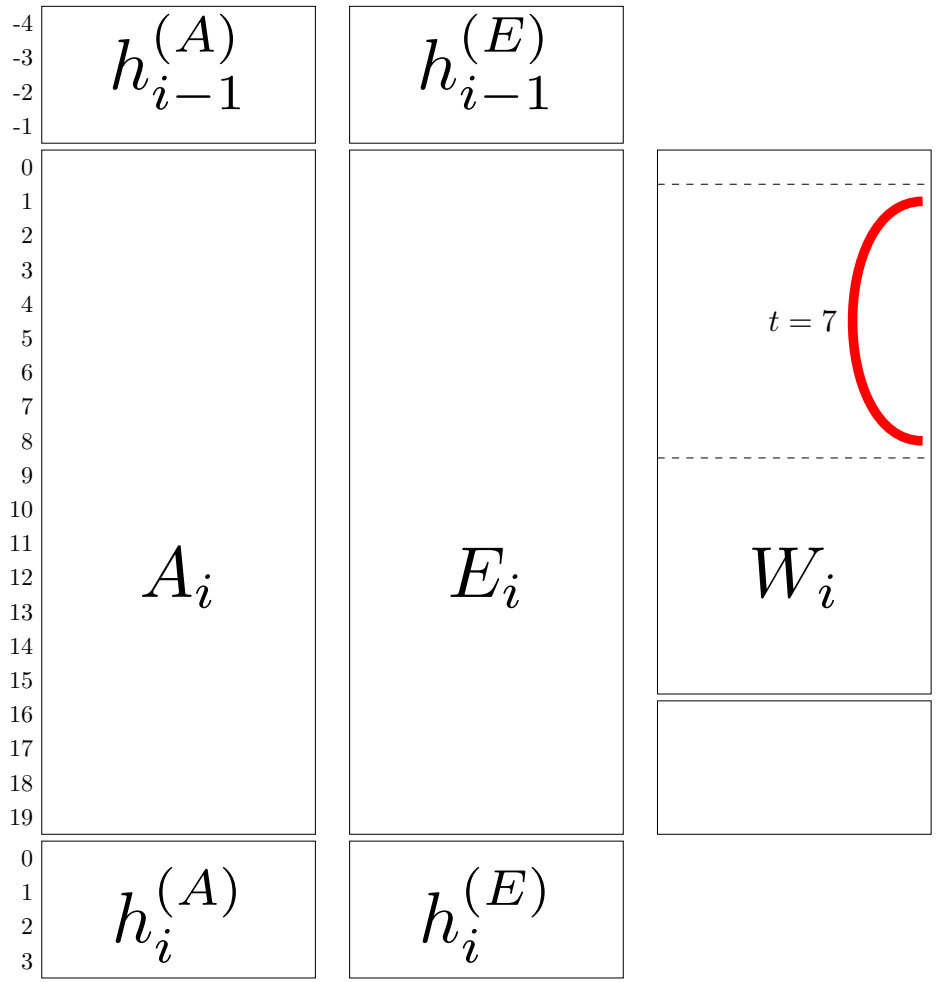
equations of small complexity: easier to update using automatic tool

Outline

Constructing Hash Collisions for 20 Steps

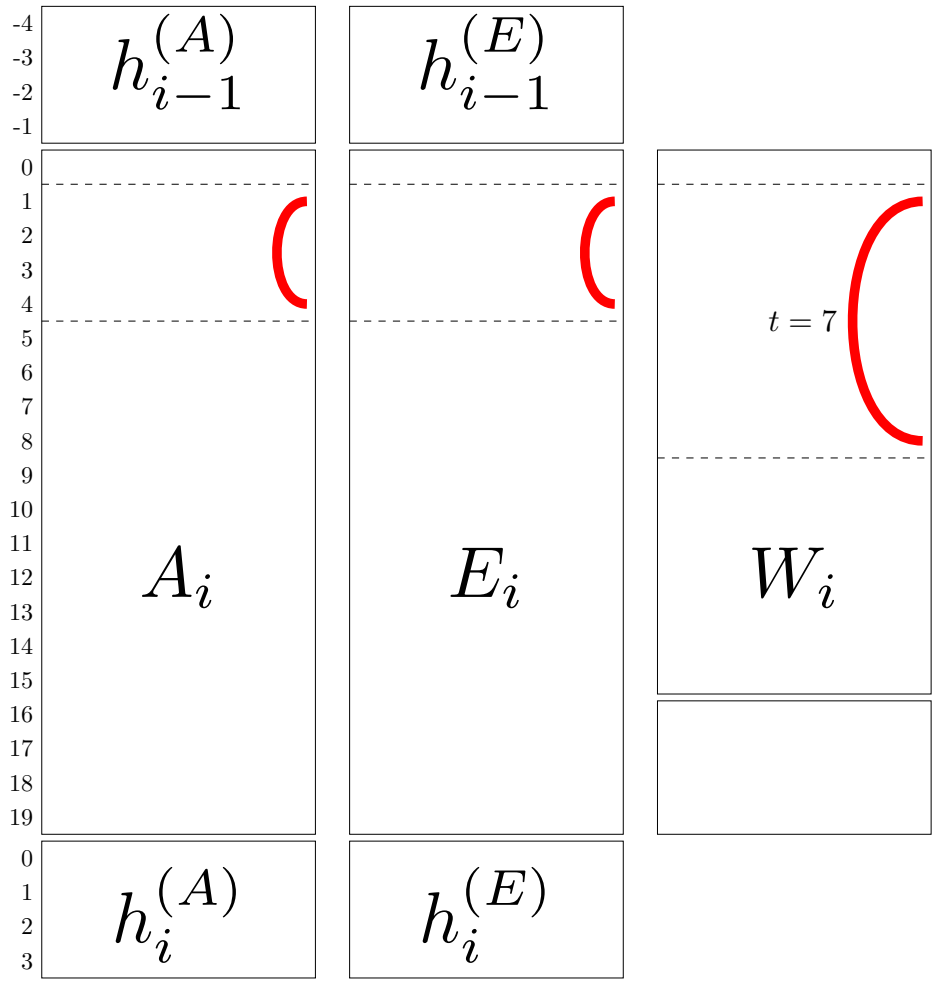


Constructing Hash Collisions for 20 Steps



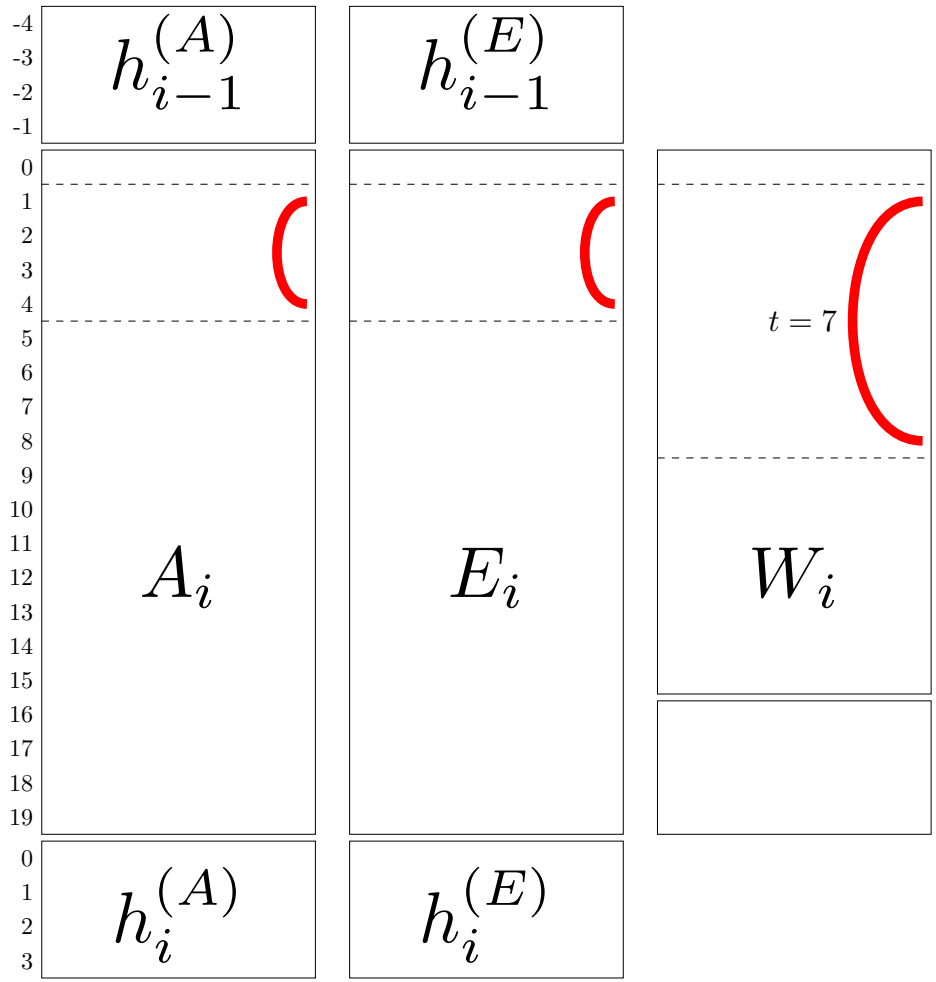
1. message word difference?
   - linear low-weight code search
   - 7-step local collision
   - 20-step hash collision

Constructing Hash Collisions for 20 Steps



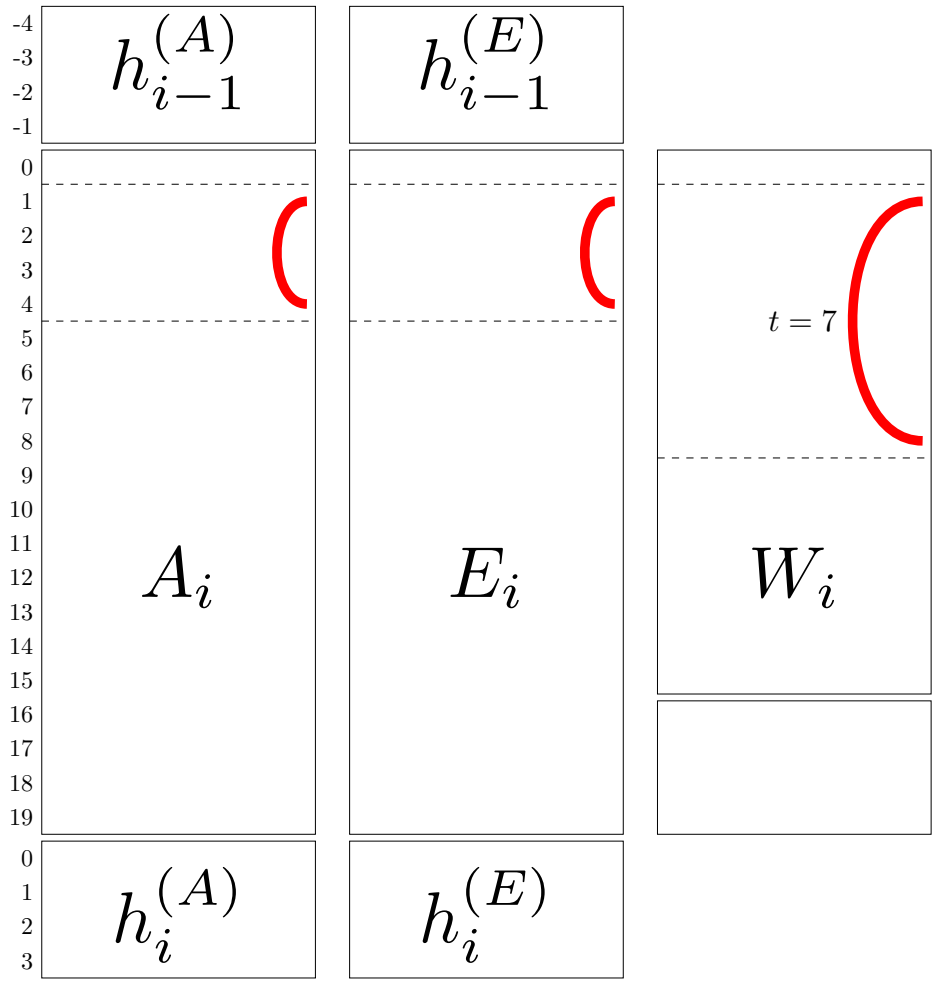
1. message word difference?
   - linear low-weight code search
   - 7-step local collision
   - 20-step hash collision
2. search for differential path
   - using semi-automatic tool
   - optimize search by hand

Constructing Hash Collisions for 20 Steps



1. message word difference?
   - linear low-weight code search
   - 7-step local collision
   - 20-step hash collision
2. search for differential path
   - using semi-automatic tool
   - optimize search by hand
3. determine message pair
   - using semi-automatic tool
   - combine with path search

Constructing Hash Collisions for 20 Steps



1. message word difference?
   - linear low-weight code search
   - 7-step local collision
   - 20-step hash collision
2. search for differential path
   - using semi-automatic tool
   - optimize search by hand
3. determine message pair
   - using semi-automatic tool
   - combine with path search

$\Rightarrow$ iterations between phases

Differences and Conditions

Generalized Conditions [DR06]

- take all 16 possible conditions on a pair of bits into account

(X_i, X_i^*)	$(0,0)$	$(1,0)$	$(0,1)$	$(1,1)$
?	✓	✓	✓	✓
-	✓	-	-	✓
x	-	✓	✓	-
0	✓	-	-	-
u	-	✓	-	-
n	-	-	✓	-
1	-	-	-	✓
#	-	-	-	-

(X_i, X_i^*)	$(0,0)$	$(1,0)$	$(0,1)$	$(1,1)$
3	✓	✓	-	-
5	✓	-	✓	-
7	✓	✓	✓	-
A	-	✓	-	✓
B	✓	✓	-	✓
C	-	-	✓	✓
D	✓	-	✓	✓
E	-	✓	✓	✓

2-bit Conditions [MNS11]

- linear relation between closely related bits: $X_i \oplus X_j = 0/1$
- 2-bit conditions on any generalized condition (-,x,?,...)
- used to determine critical bits (those with many relations)

Starting Point for 20 Steps

i	∇A_i	∇E_i	∇W_i
-4	----------	----------	
-3	----------	----------	
-2	----------	----------	
-1	----------	----------	
0	----------	----------	----------
1	??????????	??????????	---------x-x----------x----
2	??????????	??????????	----------------------x----
3	??????????	??????????	----------
4	??????????	??????????	----------
5	----------	----------	--x-------x------x----------
6	----------	----------	----------
7	----------	----------	----------
8	----------	----------	---------x-x----------x----
9	----------	----------	----------
10	----------	----------	----------
11	----------	----------	----------
12	----------	----------	----------
13	----------	----------	----------
14	----------	----------	----------
15	----------	----------	----------
16	----------	----------	----------
17	----------	----------	----------
18	----------	----------	----------
19	----------	----------	----------

Search Strategy

Search Algorithm [DR06, MNS11, MNS13]

(1) Start with an unrestricted characteristic ('?' and '-')

(2) Successively impose new conditions on the characteristic
- path search: replace '?' by '-' and 'x' by 'n' or 'u'
- message search: replace '-' by '1' or '0'

(3) Propagate the conditions and check for consistency
- if a contradiction occurs: backtrack
- else proceed with step 2

(4) Repeat steps 2 and 3 until all bits are determined

Search Strategy

The difficulties are in the details...

- Which information to propagate (and when)?
  - path search: generalized conditions
  - message search: generalized conditions and 2-bit conditions
- Which bits (which area) to guess?
  - dedicated to hash function
  - bits with many 2-bit conditions (in message search)
  - lots of trial and error needed to find best strategy
- How to backtrack?
  - if a contradiction occurs on a bit, backtrack until bit can be set
  - keep and check a list of previous critical bits

Search Strategy

The difficulties are in the details...

- Which information to propagate (and when)?
  - path search: generalized conditions
  - message search: generalized conditions and 2-bit conditions
- Which bits (which area) to guess?
  - dedicated to hash function
  - bits with many 2-bit conditions (in message search)
  - lots of trial and error needed to find best strategy
- How to backtrack?
  - if a contradiction occurs on a bit, backtrack until bit can be set
  - keep and check a list of previous critical bits

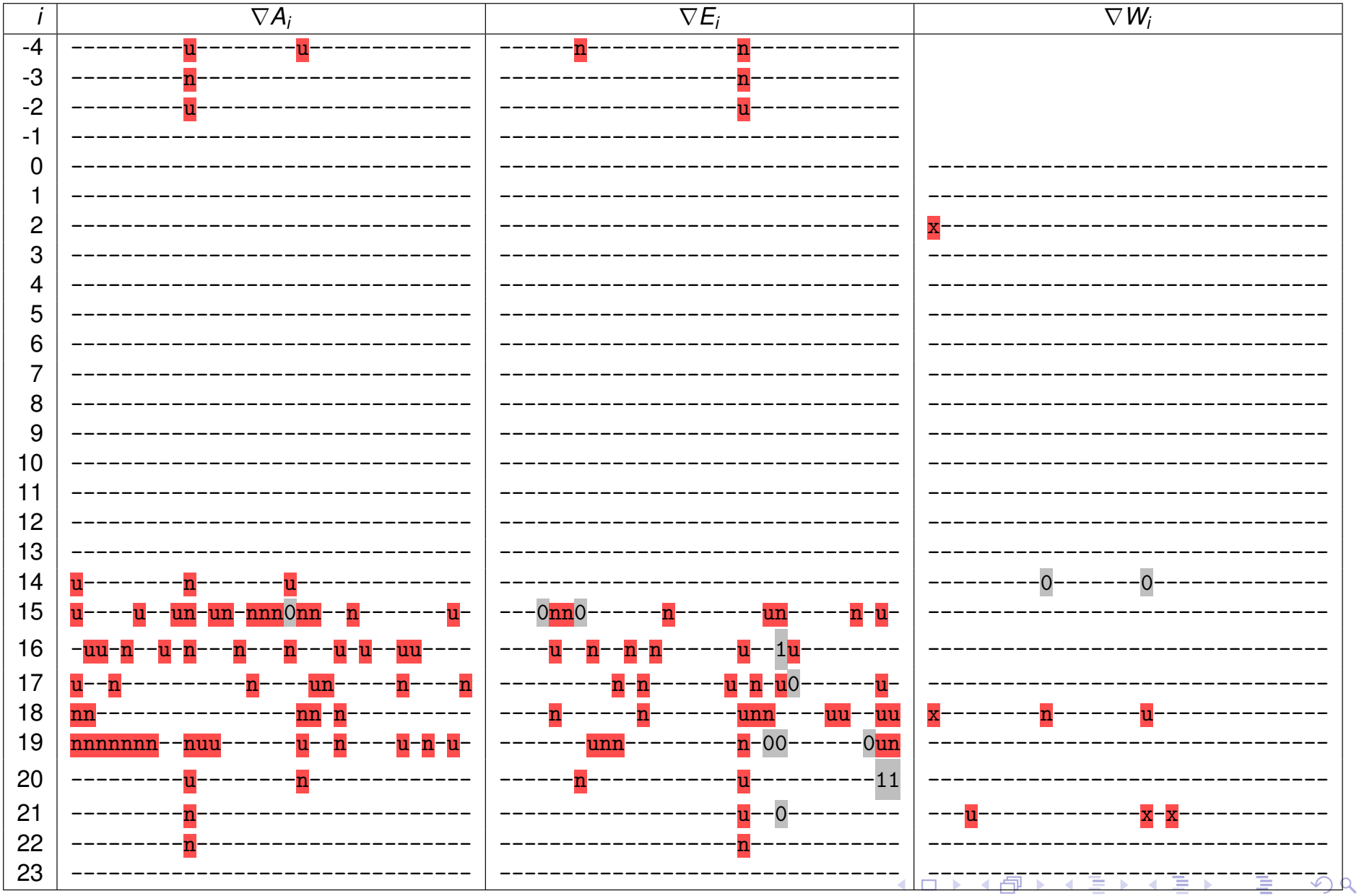
$\Rightarrow$ Dedicated for every hash function (unfortunately)

Characteristic for 20-step Hash Collision

search: first guess outputs of modular additions (L_i, A_i, R_i)

i	∇A_i	∇E_i	∇W_i
-4	----------	----------	
-3	----------	----------	
-2	----------	----------	
-1	----------	----------	
0	-------1----------	----------	----------
1	unn-----un----------uuuuuuuu--1-	u-u----unuuu------n-10nnu-nu--uu	--0------u-n-------1-------u----
2	-uu-n---un-uu--uun--n--------uun	---u0nunu----uu-u-unu-u--n----uu	----------n----
3	n-n-u--------unnnnnnn-------n-u-	nu-u--n1n-nuu-u----------nuu---	----------
4	uuuuu-------------nn--------nuu	uuuuuuuuuuuuuuu-------------u-un	----------
5	----------	----------	--u------0-u-------n-------1----
6	----------	----------	----------
7	----------	----------	----------
8	----------	----------	---------n-n-------------u----
9	----------	----------	----------
10	----------	----------	----------
11	----------	----------	----------
12	----------	----------	---------1-0-------------0----
13	----------	----------	----------
14	----------	----------	----------
15	----------	----------	----------
16	----------	----------	----------
17	----------	----------	----------
18	----------	----------	----------
19	----------	----------	----------

Characteristic for 24-step Free-Start Collision



Outline

Summary

- Results:

component	attack	steps	complexity
hash function	collision	20	example
compression function	free-start collision	24	example

- First collision attacks on reduced SM3
  - automatic path search and automatic message modification
  - time consuming to find the right settings
  - once settings are found, collisions can be found in minutes

References I


Chinese Commercial Cryptography Administration Office.
Specification of SM3 cryptographic hash function (In Chinese).
http://www.oscca.gov.cn/UpFile/20101222141857786.pdf.


Ronald Cramer, editor.
Advances in Cryptology - EUROCRYPT 2005, 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22-26, 2005, Proceedings, volume 3494 of *LNCS*. Springer, 2005.


Christophe De Cannière and Christian Rechberger.
Finding SHA-1 Characteristics: General Results and Applications.
In Xuejia Lai and Kefei Chen, editors, *ASIACRYPT*, volume 4284 of *LNCS*, pages 1–20. Springer, 2006.


Aleksandar Kircanski, Yanzhao Shen, Gaoli Wang, and Amr Youssef.
Boomerang and Slide-Rotational Analysis of the SM3 Hash Function.
In Lars R. Knudsen and Huapeng Wu, editors, *Selected Areas in Cryptography*, LNCS. Springer, 2012.
to appear.


Florian Mendel, Tomislav Nad, and Martin Schläffer.
Finding SHA-2 Characteristics: Searching through a Minefield of Contradictions.
In Dong Hoon Lee and Xiaoyun Wang, editors, *ASIACRYPT*, volume 7073 of *LNCS*, pages 288–307. Springer, 2011.

References II


Florian Mendel, Tomislav Nad, and Martin Schläffer.
Improving Local Collisions: New Attacks on Reduced SHA-256.
In *EUROCRYPT*, 2013.
to appear.


Xiaoyun Wang, Xuejia Lai, Dengguo Feng, Hui Chen, and Xiuyuan Yu.
Cryptanalysis of the Hash Functions MD4 and RIPEMD.
In Cramer [Cra05], pages 1–18.


Xiaoyun Wang and Hongbo Yu.
How to Break MD5 and Other Hash Functions.
In Cramer [Cra05], pages 19–35.


Xiaoyun Wang, Yiqun Lisa Yin, and Hongbo Yu.
Finding Collisions in the Full SHA-1.
In Victor Shoup, editor, *CRYPTO*, volume 3621 of *LNCS*, pages 17–36. Springer, 2005.


Jian Zou, Wenling Wu, Shuang Wu, Bozhan Su, and Le Dong.
Preimage Attacks on Step-Reduced SM3 Hash Function.
In Howon Kim, editor, *ICISC*, volume 7259 of *LNCS*, pages 375–390. Springer, 2011.

Security in knowledge

Many Weak Keys for PRINTcipher: Fast Key Recovery and Countermeasures

Michael Walter

UC San Diego

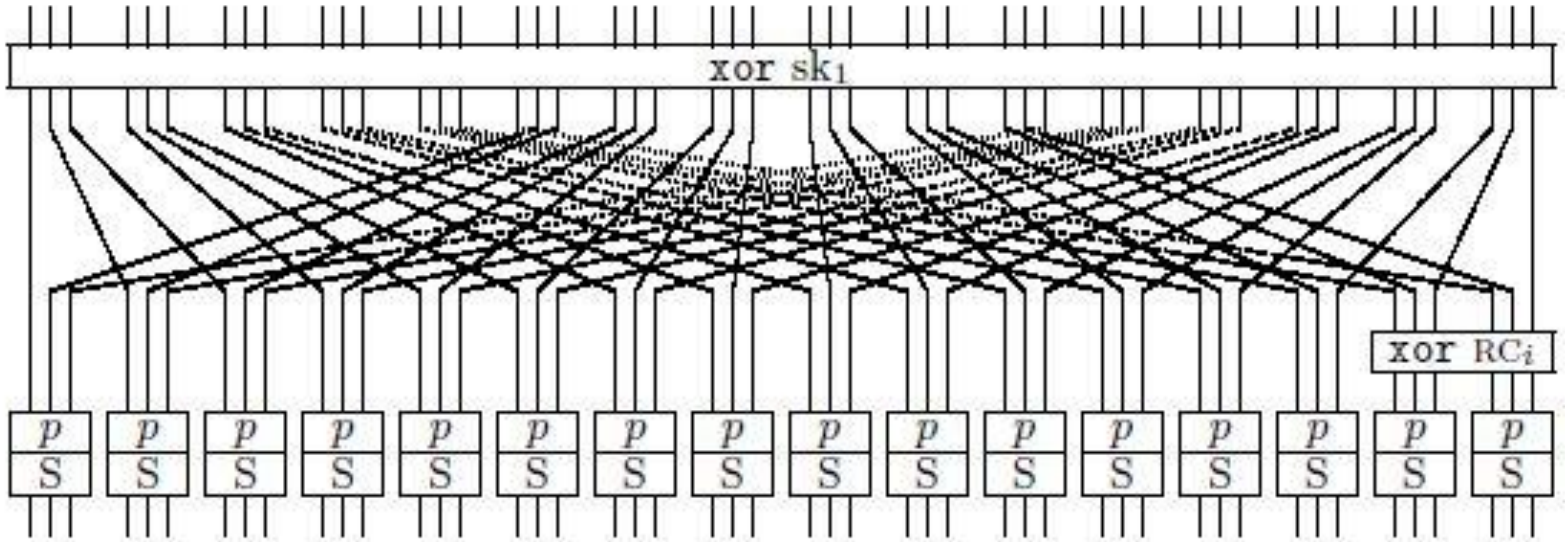
RSA CONFERENCE 2013

Session ID: CRYP-R31

Session Classification: Advanced

PRINTcipher

- Iterative block cipher, substitution-permutation network
- 48-bit blocks and 80-bit key, 48 rounds
- 96-bit blocks and 160-bit key, 96 rounds

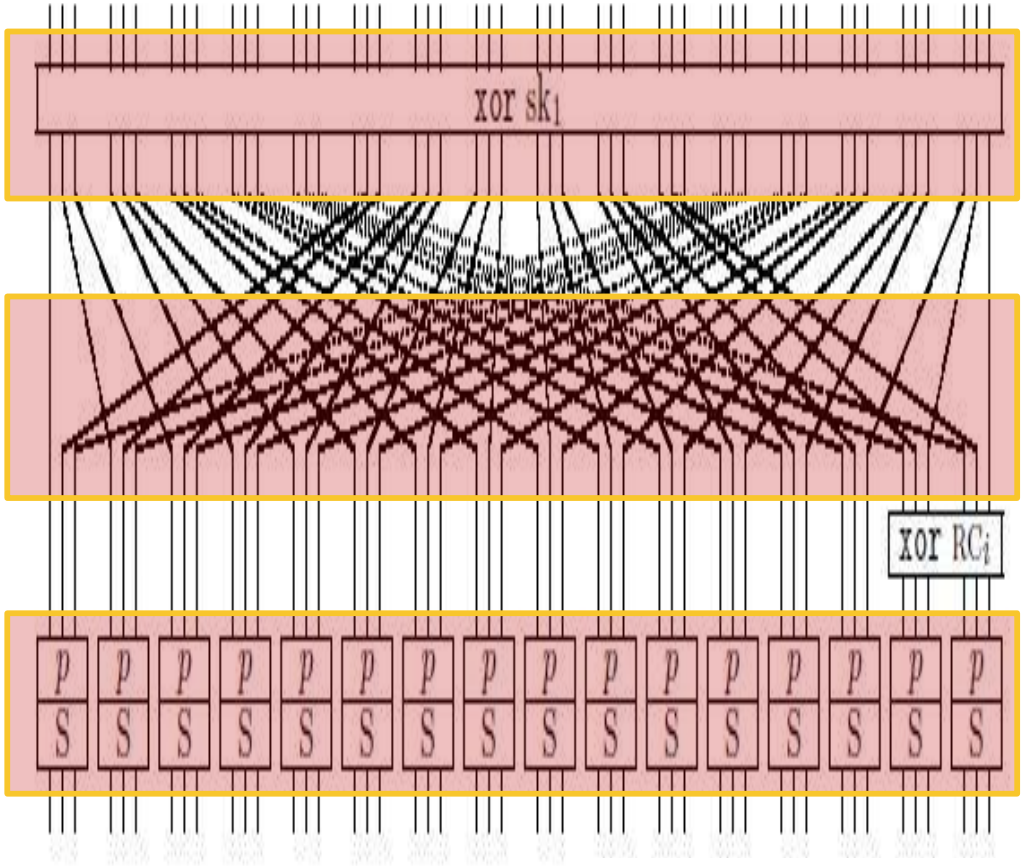


Round function

- Plaintext $\mathbf{p} = \mathbf{y}_0$
- $i = 0, \ldots, 47$
  - KeyAddition
    $\mathbf{s}_i = \mathbf{y}_i \oplus \mathbf{sk}_1$
  - BitPermutation

$j = 0, \ldots, 47 : \; x_{i,Perm(j)} = s_{i,j}$

  - S-Box
    $\mathbf{y}_{i+1} = S(\mathbf{x}_i, \mathbf{sk}_2)$
- Ciphertext $\mathbf{c} = \mathbf{y}_{48}$

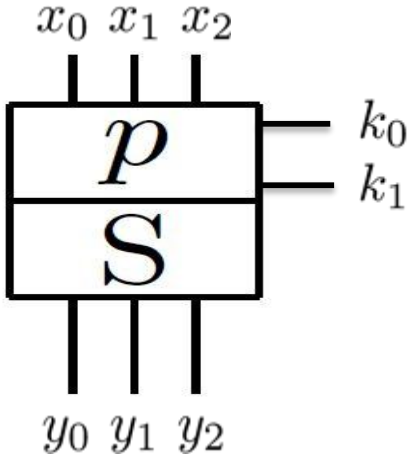


Invariant coset attack

- First studied by Leander et al. CRYPTO 2011
- $k \in WK, p \in P_{WK} \Rightarrow c := PRINTcipher_k(p) \in P_{WK}$
- Attack:
  - Distinguish k by checking $c \in P_{WK}$ if $p \in P_{WK}$
  - Knowing $k \in WK$ recover k faster than brute force
- $P_{WK} = U + d, U = \{x \in F_2^n | x_{i_1} = 0, \ldots, x_{i_t} = 0\}$
- Therefore: *invariant* (linear) *coset* attack

Masks for S-Boxes

- Values of some input bits $\Rightarrow$ values of some output bits

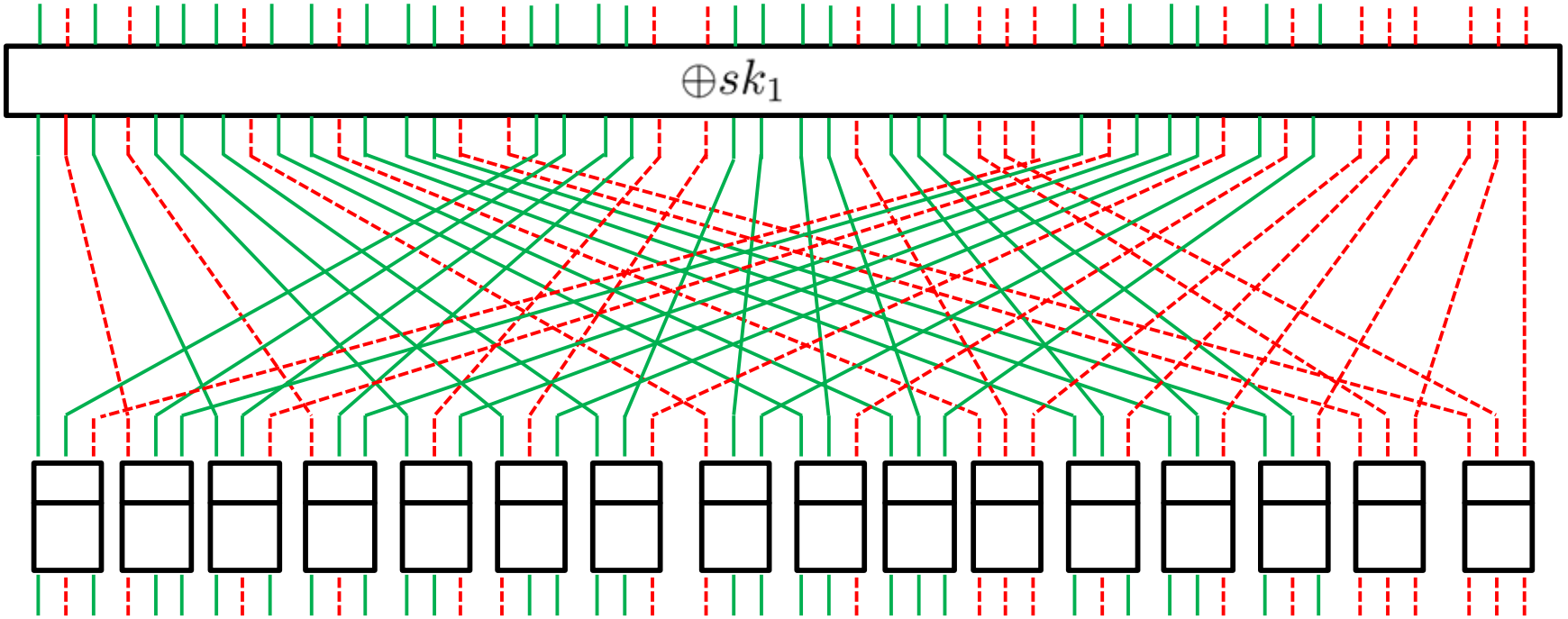


Example:

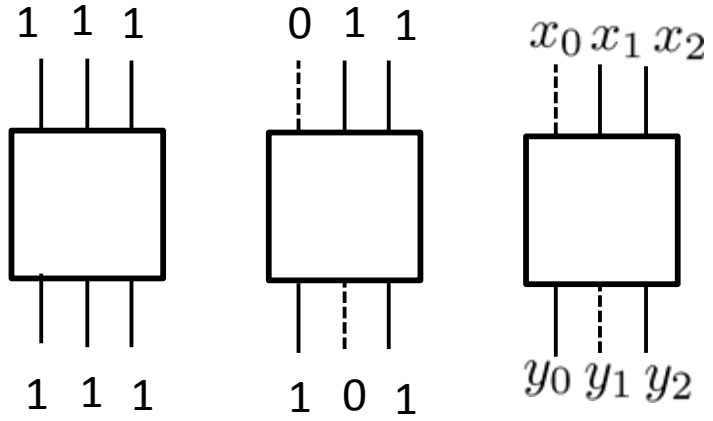
$(k_0, k_1) = (1, 0) \Rightarrow 10* \rightarrow 1*1$

- 3 – 3 and 0 – 0 masks
- Any of 3*3=9 possible arrangements yields a 2 – 2 mask for some (k_0, k_1)
- No 1 – 1 masks
- a – a masks for a=0,2,3 yield an invariant property

Example of invariant coset via masks

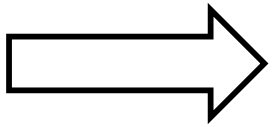


Getting invariant cosets systematically



$$y_0 + y_1 + y_2 = x_0 + x_1 + x_2,$$
$$y_0 + y_1 \geq y_2,$$
$$y_0 + y_2 \geq y_1,$$
$$y_1 + y_2 \geq y_0$$

S-Box conditions + bit permutation



$$T \subset \mathbf{Z}^{48} \cap \{0,1\}^{48}$$

Theorem 1 (Invariant property) *For $v \in T$ define $T_v := \{i | v_i = 1\} \subset \{0,1\}^{48}$. For each $v \in T : \exists U \subset \mathbf{F}_2^{48}$ coset projected at positions from T_v such that*

$$PRINTcipher_k(U) = U$$

for all $k \in WK_U \subset \mathbf{F}_2^{80}$.

Using the technique

- Key recovery:
  - For $p \in U$ observe whether $c = PRINTCipher_k(p) \in U$ thus we have $k \in WK_U \Rightarrow$ many bits of k are known
  - Recover the remaining bits of k
    - Two-step process, faster than vanilla brute force
- How to find all elements of T?
  - Since $T \subset \mathbf{Z}^{48} \cap \{0,1\}^{48}$ and is defined by linear (in)equalities, we can apply Integer Linear Programming techniques
  - Constraints: S-Box conditions + bit permutation as above
  - Objective function: constant on T
  - Practically:
    - all 64 elements of T (for 48-bit version), and
    - 115,669 elements (96-bit)

  in a fraction of a second using the IBM ILOG CPLEX solver

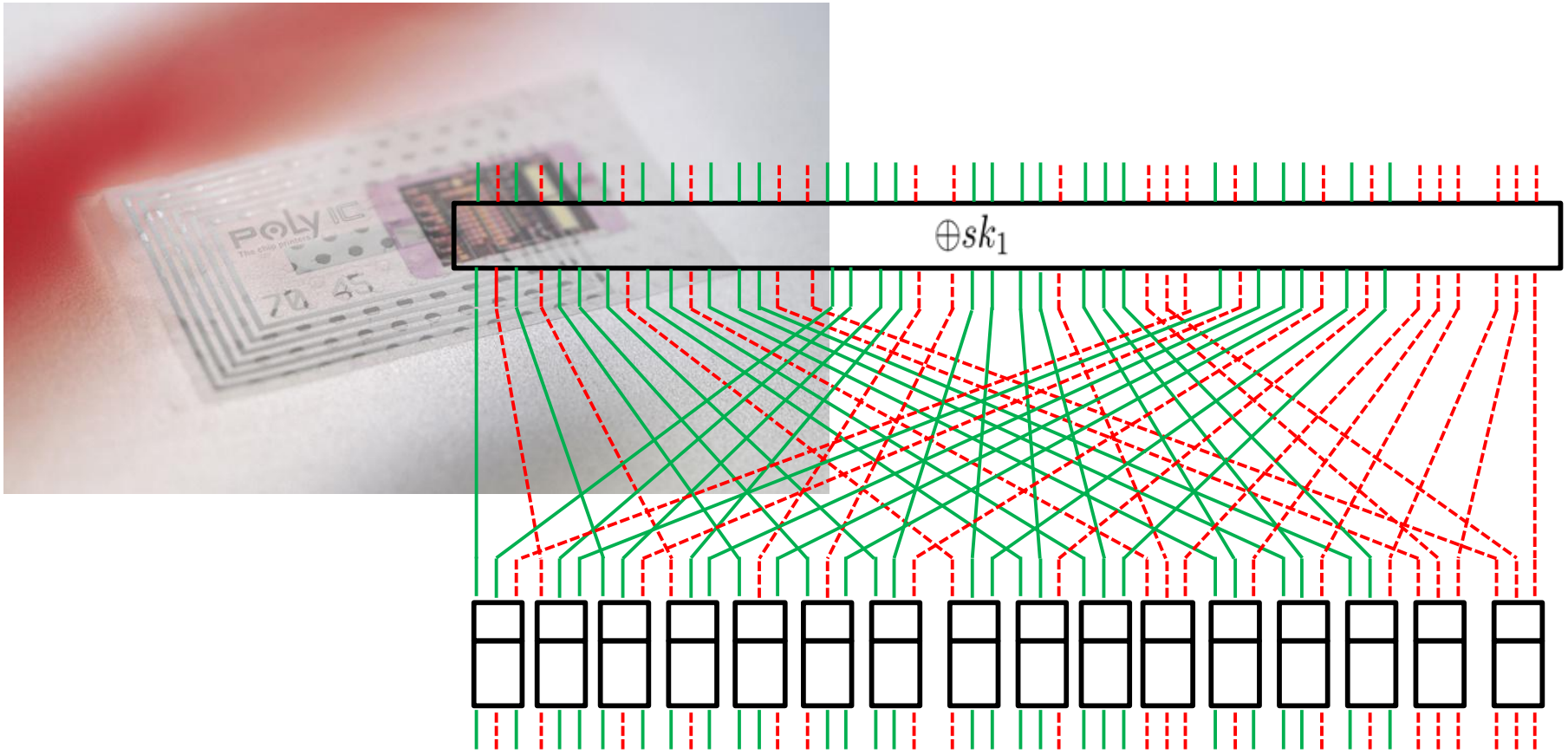
Results for PRINTcipher - 48

	Our analysis	Leander et al.
# weak key families / all found?	64 / Yes	2 / No
Log (upper bound on # weak keys)	52.5	52
Log (fastest time for key recovery in CP/KP scenario)	24	38

Results for PRINTcipher - 96

	Our analysis	Leander et al.
# weak key families / all found?	115,669 / Yes	2 / No
Log (upper bound on # weak keys)	117.7	102
Log (fastest time for key recovery in CP/KP scenario)	30	76

- Countermeasures exist, but not obvious!

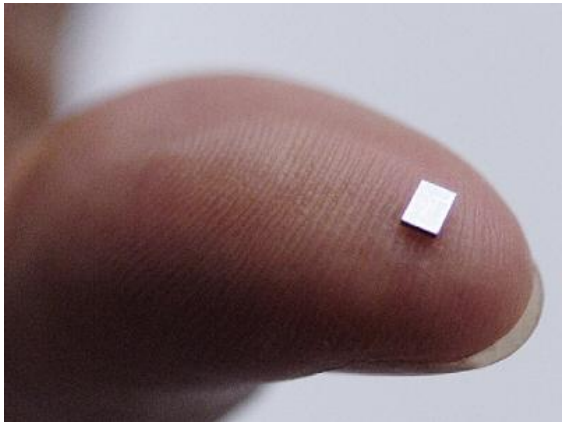


http://eprint.iacr.org/2012/085

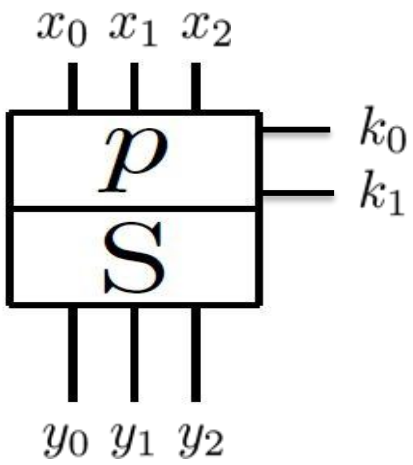
Security in knowledge

Motivation: Lightweight

- Vast growth in using resource constraint devices
- Ergent need for crypto for these devices
  - In particular block ciphers
- Standard primitives (e.g. AES) are too expensive to implement



Masks in detail



Input mask	Output mask	k0,k1	x0,x1,x2	y0,y1,y2
+ + -	+ + -	0*	00*	00*
+ + -	+ - +	10	10*	1*1
+ + -	- + +	11	11*	*10
+ - +	+ + -	10	0*0	00*
+ - +	+ - +	00/11	0*1/1*0	1*1
+ - +	- + +	01	1*1	*10
- + +	+ + -	11	*00	00*
- + +	+ - +	01	*10	1*1
- + +	- + +	*0	*11	*10

Characterization

1. $X \cap \{n-6, \ldots, n-1\} = \oslash$;
2. X is a $\bar{1}-$subset of $\mathbf{Z}_n$.
3. $P(X)$ is a $\bar{1}-$subset of $\mathbf{Z}_n$.
4. $\forall\ 0 \leq j < n/3$:

$$|\{3j, 3j+1, 3j+2\} \cap X| = |\{3j, 3j+1, 3j+2\} \cap P(X)|.$$

Example invariant propagation

in	1*1	*10	1*1	1*1	00*	*10	00*	*10	00*	XYZ	***	1*1	00*	1*1	***	***
sk_1	0*0	*11	1*1	K*0	01*	*11	00*	*11	0B*	XYZ	***	0*0	01*	0*L	***	***
XOR	1*1	*01	0*0	A*1	01*	*01	00*	*01	0B*	000	***	1*1	01*	1*C	***	***
P	10*	*11	10*	*01	0*0	1*1	00*	*11	00*	ABC	***	10*	00*	10*	***	***
sk_2	10	∀0	10	01	10	01	0∀	∀0	0∀	EF	**	10	0∀	10	**	**
out	1*1	*10	1*1	1*1	00*	*10	00*	*10	00*	XYZ	***	1*1	00*	1*1	***	***